Índice de contenido

1 Análisis....2
1.2 Análisis de requerimientos....2
1.2.1 Requisitos funcionales....2
1.2.2 Requisitos no funcionales....2
1.3 Actores del sistema....3
1.4 Modelado de casos de uso....3
1.5 Diagramas de secuencia....5
1.5.1 Comprobar estado del pestillo....5
1.5.2 Parear cuenta....7
1.5.3 Desparear cuenta....9
2 Diseño....11
2.1 Modelado de la base de datos....11
2.2 Modelado estático....15
3 Implementación....18
3.1 Diagrama de despliegue....18
3.2 Mediawiki....19
3.2.1 Hooking....19
3.2.2 Wrapper de la base de datos....23
3.2.3 Internacionalización (i18n) y Localización (L10n)....26
3.3 Latch....28
3.3.1 API de Latch....29
El encabezado Authorization....29
El encabezado X-11Paths-Date....31
3.3.2 SDK de Latch para PHP....32
4 Pruebas....32

Índice de ilustraciones

Ilustración 1: Diagrama de casos de uso general del sistema....4
Ilustración 2: Diagrama de secuencia: comprobar estado....5
Ilustración 3: Diagrama de secuencia: parear cuenta....7
Ilustración 4: Diagrama de secuencia: desparear cuenta....9
Ilustración 5: Diagrama entidad relación....12
Ilustración 6: Paso a tablas del diagrama entidad relación....13
Ilustración 7: Diagrama de clases....15
Ilustración 8: Diagrama de despliegue....19

Índice de tablas

Tabla 1: Flujo de eventos del diagrama de secuencia: comprobar estado....6
Tabla 2: Flujo de eventos del diagrama de secuencia: parear cuenta....8
Tabla 3: Flujo de eventos del diagrama de secuencia: desparear cuenta....10
Tabla 4: Prueba pareo de cuenta....32
Tabla 5: Prueba de despareo de cuenta....32
Tabla 6: Prueba intento de acceso con pestillo abierto....33
Tabla 7: Prueba intento de acceso con pestillo cerrado....33
Tabla 8: Prueba envío de código OTP incorrecto....33
Tabla 9: Prueba intento de despareo erróneo....33

1 Análisis

Este documento describe los aspectos técnicos del proyecto detallando el análisis, el diseño, la implementación y las pruebas de la extensión de autenticación basada en 2FA con Latch para la plataforma Mediawiki.
El documento comienza con el análisis de requisitos del sistema donde se capturan los principales casos de uso. A continuación, se incluye el diseño y finalmente, se citan detalles de la implementación y se incluyen las pruebas realizadas.
El siguiente apartado especifica el proceso de análisis de requisitos, donde se presentan las funciones y restricciones que el proyecto debe respetar.

1.2 Análisis de requerimientos

En esta sección se especifican los requisitos y funcionalidades que debe cumplir la herramienta. Se listan las funcionalidades básicas clasificadas según el entorno que debe ofrecerlas y posteriormente se detallan cada una de ellas presentando sus correspondientes casos de uso y diagramas de secuencia de la etapa de análisis.

1.2.1 Requisitos funcionales

En primer lugar se presenta un listado con los requisitos para el plugin. En el listado se describen las funciones que debe cumplir para satisfacer los objetivos planteados.

1. Un usuario con una cuenta en la wiki puede parear su cuenta con Latch.
2. Un usuario con una cuenta en la wiki puede desparear su cuenta de Latch.
3. Un usuario con una cuenta en la wiki pareada con Latch puede poner un pestillo digital al servicio.
4. Un usuario con una cuenta en la wiki pareada con Latch puede programar autobloqueo por tiempo.
5. Un usuario con una cuenta en la wiki pareada con Latch puede programar autobloqueo por uso.
6. Un usuario con una cuenta en la wiki pareada con Latch puede programar autobloqueo por rango horario.

1.2.2 Requisitos no funcionales

Además de los requisitos funcionales descritos anteriormente, se considera importante que el sistema cumpla otros aspectos como los mostrados a continuación:

- Se seguirán los estándares y buenas prácticas de programación de la comunidad Wikimedia a la hora de desarrollar el plugin.
- El entorno web de la wiki presentará un interfaz internacionalizada.

1.3 Actores del sistema

Se ha identificado un único actor del sistema que es el usuario final que interactúa con el sistema a través de Mediwiki para parear y desparear el servicio y que a través de la aplicación para el móvil de Latch gestiona el pestillo digital para la identidad de su usuario de Mediawiki.

1.4 Modelado de casos de uso

La siguiente figura representa el diagrama general de casos de uso que modela las funcionalidades del sistema.
Se presentan en color amarillo las opciones que se gestionan desde la wiki una vez el usuario se ha logeado en la misma y en color azul verdoso las funcionalidades que el usuario puede gestionar desde la aplicación para el teléfono de Latch.

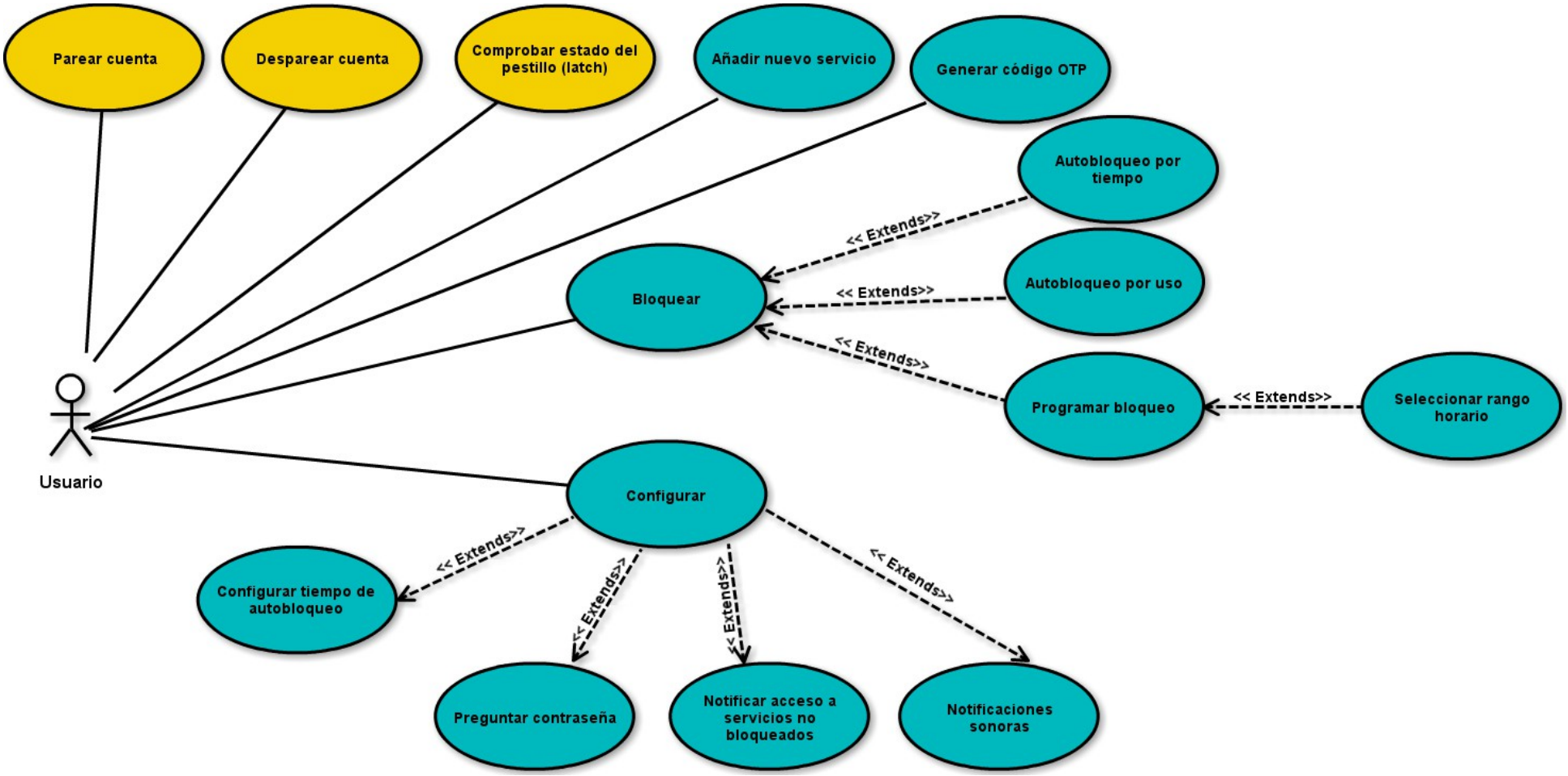


Ilustración 1: Diagrama de casos de uso general del sistema

1.5 Diagramas de secuencia

A continuación se describe con detalle cada uno de los diagramas de secuencia que modelan el sistema junto con una tabla que describe las precondiciones, postcondiciones y los posibles flujos del programa.

Los diagramas de secuencia que se muestran a continuación se corresponden con los casos de uso representados en color amarillo en el diagrama general de casos de uso del sistema.

1.5.1 Comprobar estado del pestillo

El diagrama de la ilustración 2 que se muestra a continuación modela el proceso de comprobación de un pestillo digital cuando el usuario hace login en Mediawiki.

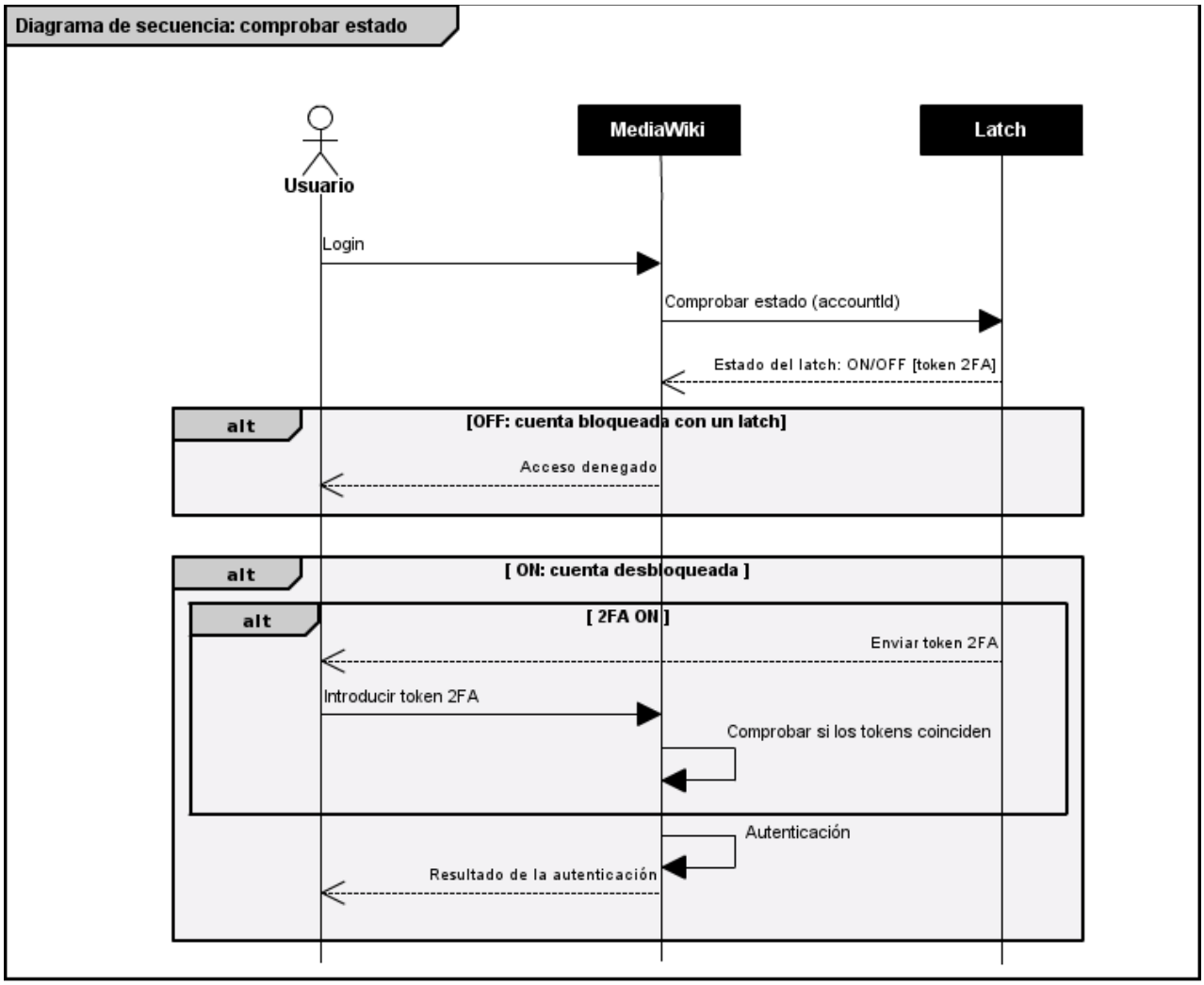


Ilustración 2: Diagrama de secuencia: comprobar estado

<table>
<tr><th colspan="2">Comprobar estado del latch</th></tr>
<tr><td>Actor</td><td>Usuario final</td></tr>
<tr><td>Descripción</td><td>El usuario final se logea con sus datos de acceso en mediawiki y el sistema comprueba si existe un pestillo digital que impida el acceso a la cuenta.
Si hay un pestillo y está cerrado el usuario recibirá una notificación de acceso fraudulento en su teléfono móvil.
Si hay un pestillo y está abierto el usuario accederá con normalidad a su cuenta de mediawiki.</td></tr>
<tr><td>Precondiciones</td><td>El usuario se logea con éxito en su cuenta de mediawiki.</td></tr>
<tr><td>Post-condiciones</td><td>Si hay un pestillo digital y está cerrado se muestra una notificación en la aplicación del teléfono del usuario avisando de un intento de acceso fraudulento.</td></tr>
<tr><td>Flujo de eventos</td><td>
<table>
<tr><th></th><th>Entrada usuario/a</th><th>Respuesta del sistema</th></tr>
<tr><td>1</td><td>El usuario introduce sus datos de acceso para logearse en mediawiki.</td><td></td></tr>
<tr><td>2</td><td></td><td>El sistema comprueba que el usuario y la contraseña sean correctos.</td></tr>
<tr><td>3</td><td></td><td>El sistema envía una petición de comprobación del estado del pestillo digital al servidor de Latch, éste le devuelve si está ON/OFF.</td></tr>
<tr><td>3.1.2</td><td></td><td>El sistema comprueba si existe un 2FA en el latch.</td></tr>
<tr><td>4</td><td></td><td>El sitema envía un token 2FA.</td></tr>
<tr><td>4.1</td><td>El usuario introduce el token 2FA.</td><td></td></tr>
<tr><td>4.2</td><td></td><td>El sistema comprueba que los tokens coinciden.</td></tr>
<tr><td>4.3</td><td>El usuario se logea con éxito.</td><td></td></tr>
</table>
</td></tr>
<tr><td>Flujo alternativo [A3.1.1]</td><td>El usuario no puede acceder porque el latch está cerrado y recibe una notificación de intento de acceso en la aplicación del teléfono móvil.</td></tr>
<tr><td>Flujo alternativo [A3.1.3]</td><td>El usuario se logea con éxito porque el latch está abierto y no existe un 2FA.</td></tr>
<tr><td>Flujo alternativo [A3.2.5]</td><td>Se muestra un mensaje al usuario diciendo que el token 2FA no es válido y le pide que vuelva a generar un token válido y lo introduzca en el formulario.</td></tr>
</table>

Tabla 1: Flujo de eventos del diagrama de secuencia: comprobar estado

1.5.2 Parear cuenta

El diagrama de secuencia de la ilustración 3 muestra los pasos necesarios para que el usuario paree su cuenta de Mediawiki con Latch, se puede ver la comunicación que se produce desde el navegador del usuario con el sistema Mediawiki, la interacción que se lleva a cabo desde la aplicación del teléfono móvil del usuario hacia el servidor de latches y su respuesta correspondiente, así como la petición desde Mediawiki hacia Latch con su correpondiente respuesta.

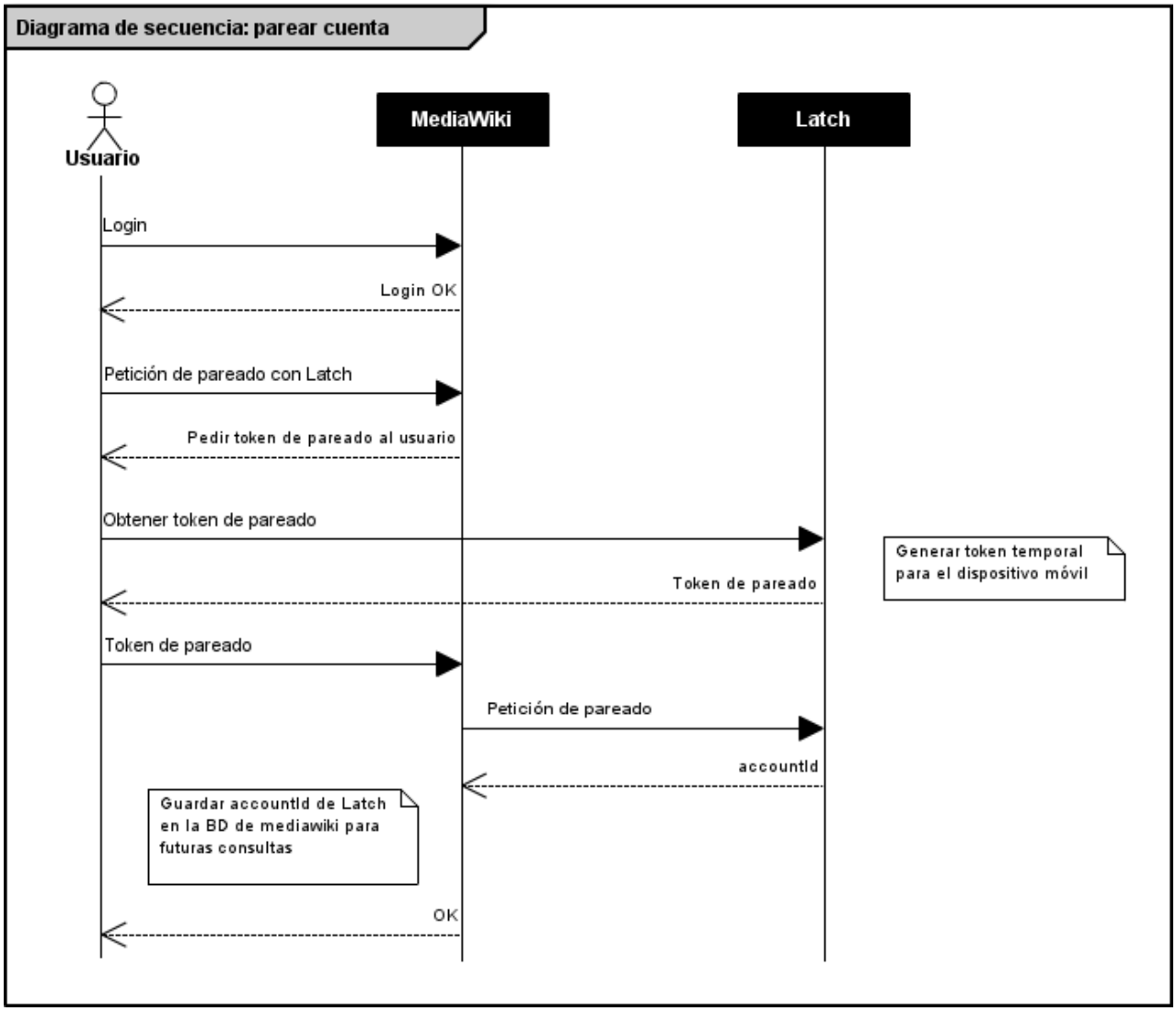


Ilustración 3: Diagrama de secuencia: parear cuenta

<table>
<tr><th colspan="3">Parear cuenta</th></tr>
<tr><td>Actor</td><td colspan="2">Usuario final</td></tr>
<tr><td>Descripción</td><td colspan="2">El usuario final tiene acceso a la opción de parear su cuenta a través del menú de preferencias de usuario dentro de su cuenta de mediawiki.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">El usuario debe estar logeado con su cuenta de mediawiki.
El usuario debe tener instalada en su teléfono smartphone la aplicación de Latch para poder generar un código OTP y enviarlo desde su cuenta de mediawiki para parear el servicio mediawiki.</td></tr>
<tr><td>Post-condiciones</td><td colspan="2">Se guarda el accountId de Latch en la base de datos de mediawiki.</td></tr>
<tr><td>Flujo de eventos</td><td colspan="2"><table>
<tr><th></th><th>Entrada usuario/a</th><th>Respuesta del sistema</th></tr>
<tr><td>1</td><td>El usuario se logea en mediawiki.</td><td></td></tr>
<tr><td>2</td><td></td><td>Se comprueba que las credenciales son correctas.</td></tr>
<tr><td>3</td><td>El usuario accede a la sección de "preferencias de usuario".</td><td></td></tr>
<tr><td>4</td><td></td><td>El sistema comprueba que el usuario de mediawiki no tiene asignado un accountId de Latch y por tanto su cuenta no está pareada, y se muestra el formulario con las opciones de pareado.</td></tr>
<tr><td>5</td><td>El usuario pide un OTP a través de la app de su smartphone.
Recibe el OTP en su teléfono y lo introduce en el formulario de pareado de mediawiki y pulsa el botón de pareado.</td><td></td></tr>
<tr><td>6</td><td></td><td>El sistema valida si el OTP es correcto, y si no está caducado (time out:60 seg.) contra el servidor de Latch. [A6]</td></tr>
<tr><td>7</td><td>Se muestra un mensaje al usuario diciendo que su cuenta ha sido pareada.
Y se actualiza el formulario que se muestra al formulario para desparear la cuenta.</td><td></td></tr>
</table></td></tr>
<tr><td>Flujo alternativo [A6]</td><td colspan="2">Si el OTP no es válido porque se ha introducido un OTP incorrecto, un OTP en blanco o ha caducado se muestra un mensaje informando al usuario de que pida un nuevo OTP a través de su teléfono y lo vuelva a introducir en el formulario.</td></tr>
</table>

Tabla 2: Flujo de eventos del diagrama de secuencia: parear cuenta

1.5.3 Desparear cuenta

El diagrama de secuencia de la ilustración 4 muestra los pasos necesarios para que el usuario desparee su cuenta de Mediawiki de Latch, se puede ver la comunicación que se produce desde el navegador del usuario con el sistema Mediawiki, la petición que se lleva a cabo desde la aplicación del teléfono móvil del usuario hacia el servidor de Latch y su respuesta correspondiente, así como la petición desde Mediawiki hacia Latch y la respuesta que este envía al sistema.

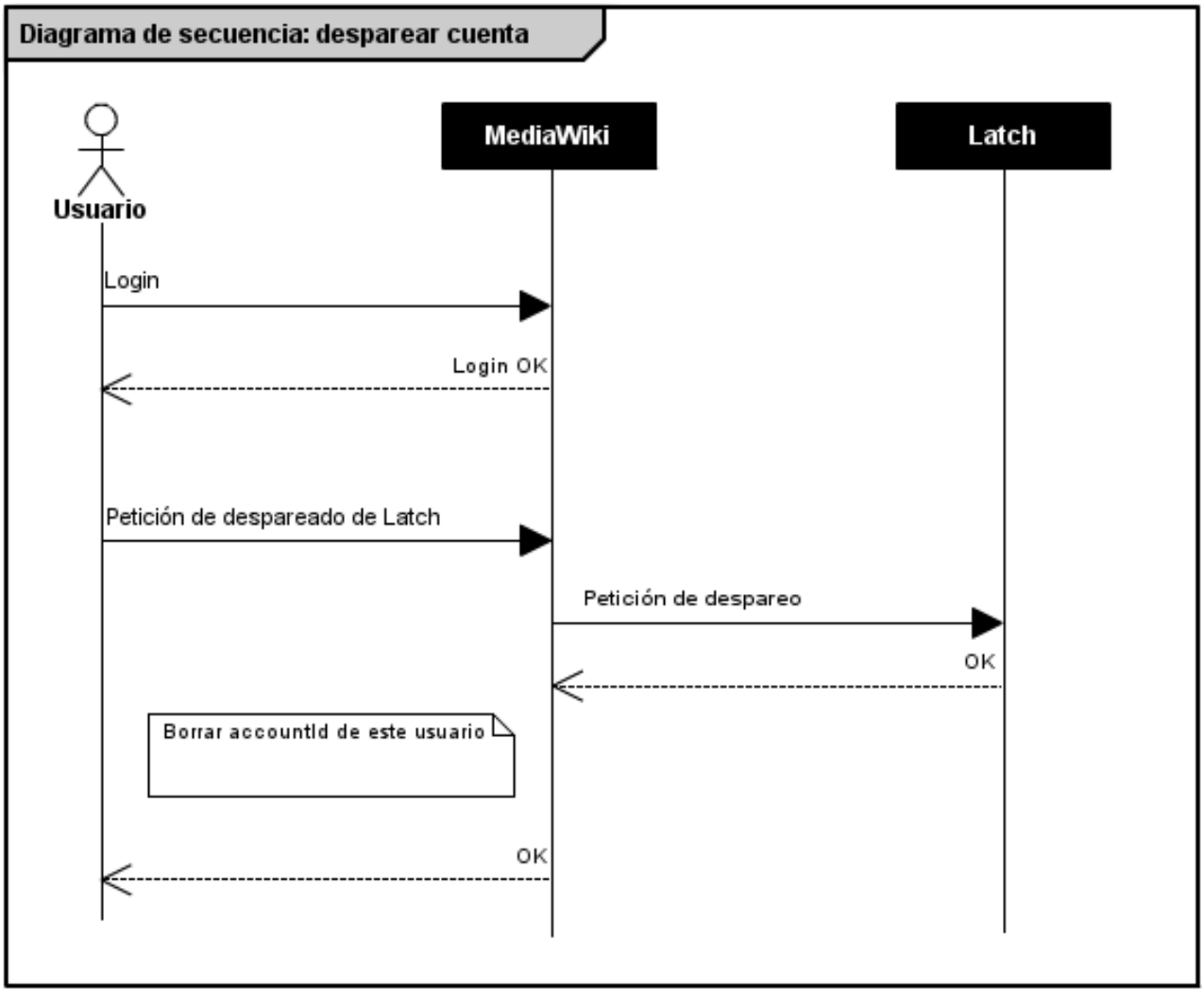


Ilustración 4: Diagrama de secuencia: desparear cuenta

<table>
<tr><th colspan="4">Desparear cuenta</th></tr>
<tr><td>Actor</td><td colspan="3">Usuario final</td></tr>
<tr><td>Descripción</td><td colspan="3">El usuario se logea con éxito en mediawiki y accede al menú de preferencias de usuario.
El usuario pulsa el botón para desparear su cuenta de Latch.</td></tr>
<tr><td>Precondiciones</td><td colspan="3">El usuario está logeado con su cuenta en mediawiki.</td></tr>
<tr><td>Post-condiciones</td><td colspan="3">Se borra el accountId de Latch de la base de datos de mediawiki.</td></tr>
<tr><td>Flujo de eventos</td><td colspan="3"><table>
<tr><th colspan="2">Entrada usuario/a</th><th>Respuesta del sistema</th></tr>
<tr><td>1</td><td>El usuario se logea en mediawiki.</td><td></td></tr>
<tr><td>2</td><td></td><td>Se comprueba que las credenciales son correctas.</td></tr>
<tr><td>3</td><td>El usuario accede a la sección de "preferencias de usuario".</td><td></td></tr>
<tr><td>4</td><td></td><td>El sistema comprueba que el usuario de mediawiki tiene asignado un accountId de Latch y por tanto su cuenta está pareada, y muestra el formulario con las opciones de despareado.</td></tr>
<tr><td>5</td><td>El usuario pulsa el botón de desparear cuenta.</td><td></td></tr>
<tr><td>6</td><td></td><td>El sistema hace una petición de despareo de la cuenta del usuario logeado en mediawiki al servidor de Latch.</td></tr>
<tr><td>7</td><td>Se muestra un mensaje al usuario diciendo que su cuenta ha sido despareada con éxito.
Se actualiza el formulario y se muestra el formulario de pareado.</td><td></td></tr>
</table></td></tr>
<tr><td>Flujo alternativo [A4]</td><td colspan="3"></td></tr>
</table>

Tabla 3: Flujo de eventos del diagrama de secuencia: desparear cuenta

2 Diseño

En esta sección se presenta el diseño del sistema, que da respuesta a las preguntas cuestionadas durante en análisis de requirimientos funcionales y no funcionales en la etapa de análisis.

A continuación veremos los diagramas que representan el modelo de la base de datos, las entidades y relaciones así como su paso a tablas y las estructura de las mismas ya que se trata de un modelo de base de datos relacional, también se incluye el diagrama de clases del sistema y una pequeña explicación de cual es la responsabilidad de cada una de clases en la extensión de autenticación.

2.1 Modelado de la base de datos

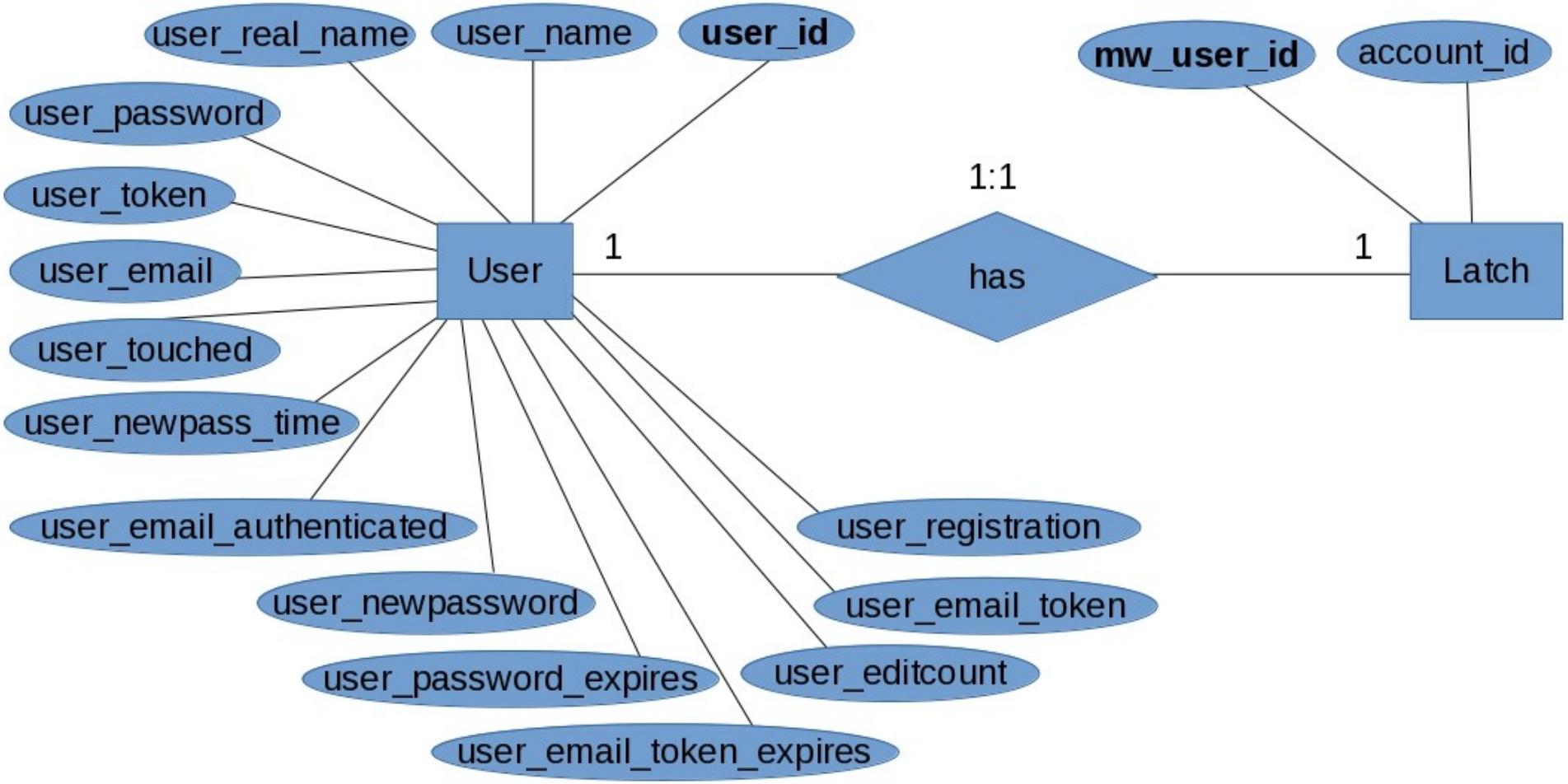
En esta sección se recoge el análisis llevado a cabo en el momento inicial de diseñar el sistema. A continuación se hace referencia a la estructura necesaria para almacenar los datos en la base de datos de Mediawiki , utilizando una base de datos de tipo MySQL.

Mediawiki es una plataforma multibase de datos, esto quiere decir que es posible elegir entre varios tipos de bases de datos diferentes a la hora de instalar la wiki, en el caso de este proyecto se ha elegido MySQL.

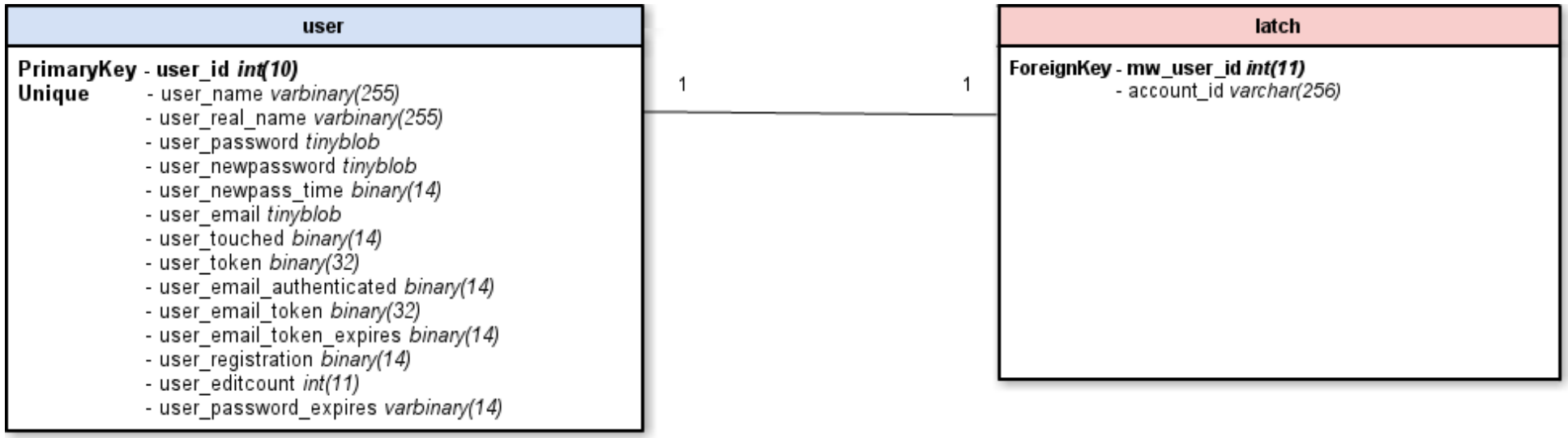
En la sección 3.2.2 se explica con granularidad más fina a nivel técnico el sistema que utiliza Mediawiki para poder gestionar consultas hacia bases de datos de distintos tipos usando una única sintaxis, haciendo uso de un wrapper para la base de datos.

En cuanto al modelo entidad relación, la tabla `latch` se usa para almacenar el identificador del usuario de Mediawiki y su identificador de pestillo digital asociado a esa cuenta.
Las entidades tienen un atributo id que se emplea como clave primaria, en el caso de la entidad latch el atributo es además clave foránea hacia la tabla user.
El campo `mw_user_id` hace referencia al id de usuario de Mediawiki.
El campo `mw_user_id` de la tabla latch es clave primaria de dicha tabla.
La relación existente entre ambas tablas es de tipo 1:1. Por tanto por cada usuario registrado en la wiki que tenga asociada una cuenta y un identificador de usuario existe una entrada con el identificador de ese usuario en la tabla `latch` junto con el identificador del pestillo digital (latch) del usuario en cuestión.
Los diagramas siguientes, correspondientes a la ilustración 5 e ilustración 6, muestran el diagrama entidad relación extendido y el paso a tablas del diagrama respectivamente.

A continuación se muestra el diagrama entidad relación extendido de la base de datos.



El siguiente diagrama muestra el paso a tablas del diagrama E/R del modelado de la base de datos.



En el siguiente gráfico se puede ver la estructura interna de las tablas contra las que actúa el plugin así como la definición de sus tipos de datos, campos y valores por defecto. La tabla `user` contiene los usuarios de Mediawiki y sus atributos.

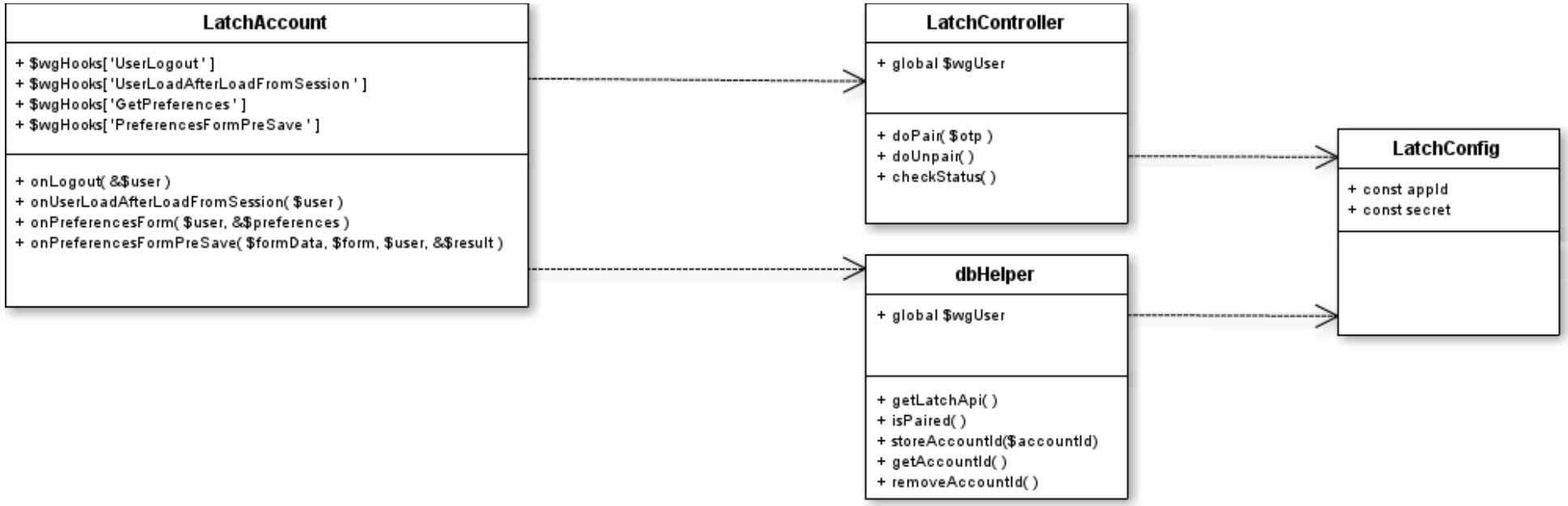
Field	Type	Null	Key	Default	Extra
user_id	int(10) unsigned	NO	PRI	NULL	auto_increment
user_name	varbinary(255)	NO	UNI		
user_real_name	varbinary(255)	NO			
user_password	tinyblob	NO		NULL	
user_newpassword	tinyblob	NO		NULL	
user_newpass_time	binary(14)	YES		NULL	
user_email	tinyblob	NO	MUL	NULL	
user_touched	binary(14)	NO			
user_token	binary(32)	NO			
user_email_authenticated	binary(14)	YES		NULL	
user_email_token	binary(32)	YES	MUL	NULL	
user_email_token_expires	binary(14)	YES		NULL	
user_registration	binary(14)	YES		NULL	
user_editcount	int(11)	YES		NULL	
user_password_expires	varbinary(14)	YES		NULL	

La tabla `latch` es creada para almacenar cada id de usuario de Mediawiki con su id correspondiente de latch.

Field	Type	Null	Key	Default	Extra
mw_user_id	int(11)	NO	PRI	NULL	
account_id	varchar(256)	YES		NULL	

2.2 Modelado estático

El diagrama de la ilustración 7 describe la estructura del sistema mostrando las clases y sus atributos y métodos. Como indica el diagrama la clase `LatchConfig.php` es usada por `LatchController.php` y por `dbHelper.php` y éstas dos últimas son invocadas desde `LatchAccount.php`



En este apartado se explica la funcionalidad de cada una de las clases del diagrama de clases de la ilustración anterior y cual es la interacción entre ellas.
En el patrón MVC las clases controladoras se encargan de interpretar los datos de la solicitud recibida. Para ello, se llama a los modelos correctos (los controladores manejan la lógica en torno a un único modelo) y se renderiza la vista correspondiente. Se consideran un enlace entre el modelo y la vista.
A continuación se describen las clases que actúan como controlador, modelo y vista.

El ***controlador*** hace referencia al código que se encuentra en el archivo `LatchController.php` hace uso de las funciones:

- `public static function doPair( $otp )` es la encargada de hacer la petición de pareado del usuario al servicio de Latch y si esta petición devuelve un ok guarda el identificador del pestillo en la base de datos de Mediawiki.
- `public static function doUnpair( )` se encarga de realizar una petición para desparear la cuenta de Mediawiki de su latch asociado en el servidor de Latch y si la respuesta muestra que todo ha salido bien, borra los datos del usuario y su identificador de latch de la base de datos de Mediawiki.
- `public static function checkLatchStatus( )`se encarga de comunicarse con el servidor de Latch para consultar cual es el estado del latch para el usuario que está actualmente logeado en Mediawiki.

Estas funciones son invocadas desde el código core de Mediawiki durante los eventos `UserLogout`,`UserLoadAfterLoadFromSession`,`GetPreferences`, `onPreferencesForm`, `PreferencesFormPreSave` y se hookean con las funciones `onLogout`,`onUserLoadAfterLoadFromSession`,`onPreferentecesForm`, `onPreferencesFormPreSave` respestivamente.

La ***vista*** está contenida en la clase `LatchAccount.php` que contiene las definiciones de las funciones que son hookeadas con los eventos del sistema nombrados en el párrafo anterior:

- La función `onLogout( &$user )` se invoca cuando el usuario hace log off de su cuenta de Mediawiki se encarga de poner a null la variable `latchStatus`, esta variable se usa para comprobar una única vez durante el login si el usuario tiene un latch activo que no permite el login.
- La función `onUserLoadAfterLoadFromSession( $user )` es invocada por el código principal de Mediawiki justo cuando la sesión del usuario actual se carga después de hacer login y se encarga de verificar si existe un latch para ese usuario y si este latch está bloqueado no permite el login del usuario a la wiki.
- La función `onPreferencesForm( $user, &$preferences )` se invoca al renderizar la vista del formulario de preferencias de usuario, se encarga de gestionar qué vista se le muestra al usuario dependiendo de si el usuario tiene su

cuenta de Mediawiki pareada con Latch o no.

- La función `onPreferencesFormPreSave` es llamada antes de guardar las preferencias del formulario de usuario y se encarga de realizar la petición de pareado o despareado según el formulario en que se encuentre el usuario y las opciones que seleccione.

El ***modelo*** se encarga de las tareas que requieren interactúar con la base de datos y su código se encuentra en el archivo `dbHelper.php`. Mediante el código de la clase `dbHelper` se almacena, consulta o elimina el identificador del usuario junto con el identificador de su pestillo digital correspondiente en la base de datos de Mediawiki haciendo uso de las funciones siguientes:

- `public static function getLatchApi( )` esta función devuelve un objeto de tipo Latch para interactúar con el API de Latch, este objeto contiene el id de aplicación y el secreto asociados al pestillo digital.
- `public static function isPaired( )` consulta en la base de datos de Mediawiki si el usuario que está actualmente logeado en la wiki está pareado o no mediante una consulta SQL a la tabla de latch realizando una búsqueda por el campo mw_user_id, esta función devuelve un valor booleano true/false en función del resultado de la consulta SQL.
- `public static function storeAccountId($accountId)` esta función almacena una entrada en la tabla latch con el identificador del usuario actualmente logeado en la wiki y su identificador de pestillo digital asociado. Para realizar el pareado de la cuenta, se invoca a esta función desde la clase `LatchAccount.php`, concretamente en la función hookeada que se llama `onPreferencesFormPreSave.`
- `public static function getAccountId( )` devuelve el identificador del latch correspondiente al usuario actualmente logeado en Mediawiki, si el usuario no ha pareado el servicio con Latch esta función devuelve una cadena vacía.
- `public static function removeAccountId( )` esta función elimina de la base de datos el registro que contiene los valores correspondientes al usuario que está logeado en ese momento en la wiki.

La clase `LatchConfig.php` es un *archivo de configuración* donde se almacenan el identificador del pestillo digital y el secreto asociados al mismo. Estos valores son usados en el método `getLatchApi( )` desde la clase `dbHelper.php`, así como desde la clase `LatchController.php` que requiere estos valores en las funciones `doPair( )` y `checkLatchStatus( ).`

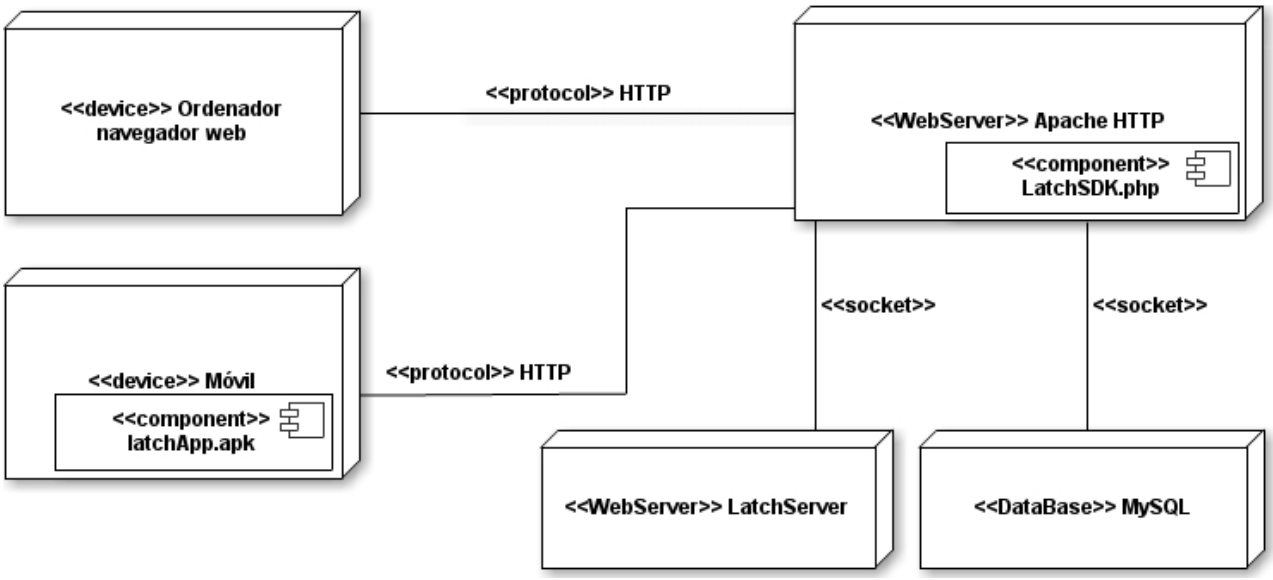
3 Implementación

En los siguientes apartados de la presente sección de implementación se muestran detalles referentes al desarrollo del código, se incluye el diagrama de despliegue de la arquitectura del sistema, se explica como funciona la técnica de hooking para interceptar llamadas del sistema de Mediawiki en base a eventos, también se explica cómo gestiona Mediawiki el hecho de ser un sistema multi base de datos proveyendo de un wrapper para la base de datos para los desarrolladores, se explica el funcionamiento de los sistemas de internacionalización `(i18n)` y localización `(L10n)` y por último los dos puntos finales de este apartado se refieren al funcionamiento de la API y el SDK de Latch.

3.1 Diagrama de despliegue

El funcionamiento del sistema es sencillo. La vista del usuario final es accesible a través de un navegador web donde se muestra la wiki a la que el usuario de conecta por el puerto 80 a través del protocolo HTTP, la wiki corre sobre un servidor Apache y haciendo uso del SDK de Latch para PHP se comunica mediante la API de Latch con servidor de Latchpara comprobar el estado del pestillo digital. El almacenamiento y gestión de acceder a los datos se realiza con MySQL. El usuario gestiona su pestillo digital de Mediawiki desde la aplicación de Latch instalada en su teléfono móvil.

En el diagrama de la ilustración 8 se puede ver una vista detallada del sistema y las capas que componen la extensión de autenticación.



3.2 Mediawiki

En el presente apartado se detallan las llamadas a funciones de la extensión de autenticación desde el código core de Mediawiki mediante el sistema de hooks. El siguiente subapartado explica con un nivel de detalle más técnico cómo funciona el wrapper de la base de datos usado por Mediawiki para permitir el acceso a distintos motores de base de datos utilizando el mismo código fuente, y por último se detalla como funcionan los sitemas de internacionalización y localización que permiten que la traducción a los diferentes idiomas se realice de una forma óptima.

3.2.1 Hooking

Los hooks de Mediawiki permiten que se ejecute código propio cuando ocurre un evento definido (como por ejemplo guardar una página de la wiki, o que un usuario haga login). Por ejemplo, el siguiente fragmento de código lanzará una llamada a la función `MyExtensionHooks:pageContentSaveComplete`

siempre que se ejecute el hook `PageContentSaveComplete`, pasándole argumentos específicos de PageContentSaveComplete:

```
$wgHooks['PageContentSaveComplete'][] =
MyExtensionHooks::pageContentSaveComplete';
```

Los hooks permiten desacoplar código que se ejecuta opcionalmente de código que se ejecuta obligatoriamente. Esto permite a los desarrolladores de Mediawiki definir código que se ejecutará en ciertos puntos del código principal de Mediawiki. Los hooks permiten mantener las líneas principales del código del core de Mediawiki limpio, claro y fácil de leer y entender, de este modo es más sencillo escribir extensiones para la plataforma.

Consideremos, por ejemplo, dos funciones en MediaWiki. Una invierte el orden de un título antes de mostrar el artículo, la otra convierte el título a letras mayúsculas.

El siguiente fragmento de código representa la situación que se acaba de detallar:

```
function showAnArticle($article) {
            global $wgReverseTitle, $wgCapitalizeTitle;
            if ($wgReverseTitle) {
                  wfReverseTitle($article);
            }
            if ($wgCapitalizeTitle) {
                  wfCapitalizeTitle($article);
            }
            # code to actually show the article goes here
}
```

Un programador que escribe una extensión para Mediawiki, es probable que encuentre la necesidad de añadir código propio a la función (con o sin usar variables globales de Mediawiki). Por ejemplo, alguien que necesite mostrar una notificación en el email cuando se muestra un artículo podría añadir el siguiente código:

```
function showAnArticle($article)
{
        global $wgReverseTitle, $wgCapitalizeTitle, $wgNotifyArticle;
            if ($wgReverseTitle) {
                  wfReverseTitle($article);
            }
```

```
        if ($wgCapitalizeTitle) {
              wfCapitalizeTitle($article);
        }
        # El código para mostrar el artículo va aquí
        if ($wgNotifyArticle) {
              wfNotifyArticleShow($article));
        }
    }
```

Utilizar una estrategia de hooks, permite eliminar todas estas opciones específicas en el código principal. Usando hooks, la función sería como se muestra a continuación:

```
function showAnArticle($article)
{
        if (wfRunHooks('ArticleShow', array(&$article) ) ) {
              # El código para mostrar el artículo va aquí
              wfRunHooks('ArticleShowComplete', array(&$article));
        }
}
```

De este modo, se mantiene el código principal limpio, eliminando fragmentos de código que no se usan de forma habitual y moviendo estos fragmentos de código a otros archivos. Así, es más fácil, para alguien que esté trabajando sobre este código realizar cambios o encontrar bugs y solucionarlos.

Siguiendo con el código del ejemplo inicial, la estrategia de hooks tiene la ventaja de permitir juntar todo el código que tiene que ver con las opciones de invertir el orden del título de un artículo y ponerlo todo en el mismo lugar. En lugar de tener `bloques if` para invertir el orden del título del artículo que estén dispersos por el código principal en `showAnArticle`, `deleteAnArticle`, `exportArticle`, etc.

De este modo se puede concentrar todo el código en un archivo para hacer una extesión que provea de la funcionalidad invertir orden del título del artículo:

```
function reverseArticleTitle($article)
{       # ...     }
```

```
function reverseForExport($article)
{          # ...       }
```

La función de configuración de la extensión debe añadir sus funciones hook a los eventos apropiados:

```
setupTitleReversingExtension() {
            global $wgHooks;
            $wgHooks['ArticleShow'][] = 'reverseArticleTitle';
            $wgHooks['ArticleDelete'][] = 'reverseArticleTitle';
            $wgHooks['ArticleExport'][] = 'reverseForExport';
}
```

Tener todo el código relacionado con las opciones de inversión de título en un único lugar implica que es más sencillo de leer y de entender, por ejemplo no sería necesario usar el comando grep-find para ver donde se utiliza la variable `$wgReverseTitle`.
Si el código está lo suficientemente bien aislado, incluso puede ser excluído cuando no está siendo utilizado, de este modo, se puede hacer un pequeño ahorro de memoria y mejorar el tiempo de carga y de ejecución del propio sistema de Mediawiki.
En el ejemplo que nos ocupa, un administrador que quisiera tener activas todas las funcionalidades de inversión de título solo debería añadir al archivo `LocalSettings.php` la siguiente línea:

```
require_once 'extensions/ReverseTitle.php';
```

Del mismo modo, los usuarios que no deseen tener disponibles estas funcionalidades, pueden no añadir la línea en cuestión al archivo `LocalSettings.php` con el consecuente ahorro de recursos y memoria del sistema.

Los hooks pueden devolver tres valores posibles:

- `true`: el hook ha funcionado correctamente.
- `string`: ha ocurrido un error, se detiene el proceso y se muestra el error al usuario.

- `false`: el hook ha hecho su trabajo correctamente y la función que lo llama debe finalizar.

El último caso es usado en situaciones donde la funcionalidad del hook reemplaza a la función principal. Por ejemplo, si se quieren autenticar usuarios en un sistema propio, como podría ser un sistema de autenticación con LDAP, se podría hacer lo siguiente:

```
$wgHooks['UserLogin'][] = array('ldapLogin', $ldapServer);
function ldapLogin($username, $password) {
            # log user into LDAP
            return false;
}
```

3.2.2 Wrapper de la base de datos

Mediawiki provee de una capa de abstracción hacia la base de datos, de modo que, a no ser que se esté trabajando en la capa de abstracción, no es buena práctica llamar directamente a funciones en PHP de la base de datos (como `mysql_query()` o `pg_send_query()`). El modo de acceder a la capa de abstracción es usar la función `wfGetDB()`. La función `wfGetDB()` recibe como parámetro una constante, que hace referencia al modo de acceso a la base de datos, que puede ser `DB_SLAVE` para consultas de lectura o `DB_MASTER` para consultas de lectura y escritura que necesitan información nueva. La diferencia entre master y slave es importante en un entorno multi base de datos como Mediawiki. Esta función devuelve un objeto `DataBase` que se usa para acceder a la base de datos. La razón por la cual es importante usar métodos de alto nivel en lugar de construir consultas propias es asegurarse de que el código funcionará correctamente sin importar el tipo de base de datos.

Actualmente Mediawiki soporta bases de datos `MySQL`, `SQLite` y `PostgreSQL`, y también pero de forma más limitada `Oracle` y `DB2`, pero puede soportar otros tipos de bases de datos como `MSSQL` o `Firebird` en el futuro.

El siguiente fragmento de código representa un ejemplo de interface que provee

Mediawiki para una consulta **SELECT**:

```
$dbr = wfGetDB( DB_SLAVE );
$res = $dbr->select(
	'category', // $table
	array( 'cat_title', 'cat_pages' ), //$vars (columnas de la tabla)
	'cat_pages > 0', // $conds
	__METHOD__, // $fname = 'Database::select',
	array( 'ORDER BY' => 'cat_title ASC' ) // $options = array()
);
```

Este ejemplo se corresponde con la consulta siguiente:

```
SELECT cat_title, cat_pages FROM category WHERE cat_pages > 0 ORDER BY
cat_title ASC;
```

También son posibles los JOINS, por ejemplo:

```
$res = $dbw->select(
	array( 'watchlist', 'user_properties' ),
	array( 'wl_user' ),
	array(
		'wl_user != 1',
		'wl_namespace' => '0',
		'wl_title' => 'Main page',
		'up_property' => 'enotifwatchlistpages',
		),
		__METHOD__,
		array(),
		array( 'user_properties' => array( 'INNER JOIN', array(
			'wl_user = up_user' ) ) )
);
```

Este ejemplo se corresponde con la consulta:

```
SELECT wl_user FROM 'watchlist' INNER JOIN 'user_properties' ON ((wl_user=up_user)) WHERE (wl_user != 1) AND wl_namespace = '0' AND wl_title = 'Main_page' AND up_property = 'enotifwatchlistpages'
```

Los parámetros son valores simples (como `'category'` y `'cat_pages > 0'`) o arrays, si se pasa más de un valor como parámetro.

Al usar `strings` como parámetro es necesario añadir `DataBase::addQuotes()` en los valores para construir la cadena ya que el wrapper no lo hace automáticamente.

La construcción del array de condiciones `$conds` está limitado, solo puede realizar relaciones de igualdad, del tipo `WHERE key = 'value'`.

Se puede acceder a filas individuales del resultado con un bucle `foreach`. Una vez se tiene el objeto fila, se puede usar el operador `->` para acceder a un campo específico.

Un ejemplo de este tipo de acceso se puede ver en el siguiente ejemplo:

```
$dbr = wfGetDB( DB_SLAVE );
$res = $dbr->select(
	'category',// $table
	array( 'cat_title', 'cat_pages' ), // $vars (columnas de la tabla)
	'cat_pages > 0', // $conds
	__METHOD__,//$fname = 'Database::select',
	array( 'ORDER BY' => 'cat_title ASC' )  // $options = array()
);
$output = '';
foreach( $res as $row )
{
	$output .= 'Category ' . $row->cat_title . ' contains ' . $row->cat_pages . " entries. \n";
```

3.2.3 Internacionalización (i18n) y Localización (L10n)

Los sistemas `i18n` (internacionalización) y `L10n` (localización) son componentes esenciales desde las primeras fases de construcción e integración del software y uno de los cores principales de Mediawiki.

La internacionalización y la localización son medios para adaptar software a los diferentes idiomas, diferencias regionales y requerimientos técnicos.

El core de Mediawiki y las extensiones utilizan un sistema de mensajes para cualquier texto mostrado en la interfaz de usuario de Mediawiki.

Existe un objeto Language en `Language.php`.

Este objeto contiene todos los mensajes de tipo `String`, así como otras opciones específicas para idiomas y comportamientos personalizados (mayúsculas, minúsculas, imprimir formatos de fechas, formatos numéricos, reglas de gramática etc.)

Este objeto se construye desde dos fuentes: instancias de la propia clase y ficheros de mensajes. Para manejar la entrada de texto a través del namespace de Mediawiki se usa la clase `MessageCache`.

La mayor parte de la internacionalización se hace a través de objetos `Message`, y usando la función `wfMessage()`, que está definida en `includes/GlobalFunctions.php`.

Para el paso mensajes en `PHP` se usa la función global `wfMessage()` que actúa como un `wrapper` para la clase `Message`, creando un objeto `Message`.

A continuación se muestra un ejemplo que invoca el método `text()` de `Message`, este método recupera el texto del mensaje `'enviar'` en el lenguaje en el que esté configurada la wiki, realiza ciertas transformaciones (género y plural) y devuelve el texto del mensaje sin escapar:

```
$out = Xml::submitButton( wfMessage( 'submit' )->text() );
```

A continuación se muestra un ejemplo más complejo mediante un mensaje que cuenta el

número de páginas y soporta el plural lingüístico:

```
$out = Xml::label( wfMessage( 'numberofpages' )->numParams( $count )->text() );
```

Existen dos formas de obtener un objeto `Language`. Se pueden usar las variables globales `$wgLang` y `$wgContLang` para la interface de usuario y el idioma del contenido respectivamente.
Para un idioma arbitrario se construye un objeto usando `Language::factory('en')`, y reemplazando en con el código del idioma.
También se puede usar `wfGetLangObj($code)` si la variable `$code` es un objeto de Language. La lista de todos los códigos se encuentra en `languages/Names.php`.

Mediawiki usa un repositorio central de mensajes que son referenciados por claves (keys) en el código fuente. El sistema basado en claves (keys) hace que algunas tareas como refinar los textos originales y tracear cambios en los mensajes, sea sencillo.
El inconveniente es, que la lista de mensajes usados y la lista de los textos de origen de estas llaves (keys) pueden no estar sincronizadas.
En la práctica, esto no es un gran problema, y el único problema importante es que los mensajes que ya no se usan (`deprecated`) se mantienen en el sistema para traducciones.
La convención global para nombrar llaves (keys) para los mensajes es utilizar un prefijo estándar, preferentemente el nombre de la extensión en letras minúsculas, seguido de un guión.
Ejemplo del fichero `extensions/Latch/i18n/en.json`:

```
"prefs-2FA-label" : "Pair your account",
"prefs-2FA-button-pair" : "Send",
"prefs-2FA-button-unpair" : "Unpair account"
```

Los mensajes que forman parte del core de Mediawiki se encuentran en `languages/i18n/en.json`.

Los mensajes que forman parte de una extensión (plugin) se encuentran en `i18n/en.json` en el subdirectorio de la propia extensión.
Si la extensión tiene un número muy elevado de mensajes se pueden crear subdirectorios dentro de `i18n` y listarlos en la variable `$wgMessagesDirs`.
La documentación de los mensajes incluye una descripción de estos mensajes para que el equipo de traductores de Mediawiki tenga contexto al traducir las extensiones a los diferentes idiomas en los que Mediwiki está disponible.

Para documentar los mensajes, se utiliza un pseudocódigo llamado `qqq` y contenido en el fichero `qqq.json`.
Este pseudocódigo es uno de los códigos ISO 639 reservados para uso privado.
El archivo `qqq.json` reune frases en inglés sobre cada una de las llaves (keys), estas frases explican dónde se usa el mensaje, dan pistas sobre cómo traducir el mensaje, enumeran y describen sus parámetros y contienen links a mensajes relacionados.
Los desarrolladores deben documentar cada mensaje añadiendo la entrada `qqq` correspondiente cuando se añade un nuevo mensaje al software o cuando se modifica un mensaje que ya existe en idioma inglés, de este modo el equipo de traducción y en equipo encargado de mantener el módulo tienen acceso a la documentación de los mensajes.

En el caso de esta extensión de autenticación se han creado ficheros de idioma que permiten que el plugin para Latch esté disponible además de en inglés, en castellano, gallego, euskera, catalán, turco y polaco.

3.3 Latch

Para poner un segundo factor de autenticación en el sistema libre de wikis Mediawiki se eligió utilizar la plataforma Latch por su sencillez de uso a través del teléfono móvil.

En este apartado se explica el funcionamiento de la API de Latch y el SDK de Latch para

PHP con un nivel de detalle técnico.

3.3.1 API de Latch

A continuación se describe cómo funciona la autenticación en el servicio Latch. Todas las solicitudes a la API de Latch deben estar firmadas. El proceso de firma es una versión simplificada del protocolo Oauth de dos vías. Cada solicitud HTTP a la API debe ir acompañada de dos encabezados de autenticación: Authorization y Date.

El encabezado Authorization

El encabezado Authorization debe tener el formato siguiente:

```
11PATHS requestId requestSignature
```

requestId puede ser o bien un 'applicationId' o un 'userId', dependiendo de la API a la que acceda la petición.

11PATHS es una constante que determina el método de autenticación.

applicationId es un identificador alfanumérico que se obtiene al registrar una aplicación nueva.

requestSignature es una firma derivada de la url, parámetros, encabezados personalizados y fecha de la solicitud actual, todos ellos con hash utilizando un Secreto que también se obtiene mediante el applicationId al registrar la aplicación.

La firma de solicitud es una cadena codificada en Base64 y con firma HMAC-SHA1.

La cadena se debe crear siguiendo este proceso:

1) Empezar por una cadena vacía.

2) Anexar el nombre de método en mayúsculas. Actualmente, solo se admiten los valores **GET**, **POST**, **PUT** y **DELETE**.

3) Anexar un separador de línea, " " (Punto Unicode U+000A).

4) Anexar una cadena con la fecha actual en este formato exacto **aaaa-MM-dd HH:mm:ss**. Todo en este formato debe explicarse por sí solo, ten en cuenta que todo es numérico y se debe completar con ceros en caso necesario para que todos los números tengan dos dígitos, excepto el año, que tiene cuatro. Este valor se debe mantener para utilizarlo en el encabezado **Date**. El proceso de comprobación de firma fallará si no coinciden ambos.

4.1) Anexar un separador de línea, " " (Punto Unicode U+000A).

4.2) Serializar todos los encabezados específicos de esta aplicación (no cada encabezado HTTP de la solicitud). Todos los nombres de estos encabezados empiezan por **X-11paths-**.

4.2.1) Convertir todos los nombres de encabezados a minúsculas.
Ordenar los encabezados por nombre de encabezado en orden alfabético ascendente.
4.2.2) Para cada valor de encabezado, convierte los encabezados de varias líneas en una única línea sustituyendo los caracteres de línea nueva " " por espacios sencillos " "
4.2.3) Crear una cadena vacía. A continuación, para cada encabezado después de la ordenación y transformaciones anteriores, añadir a la cadena recién creada el nombre de encabezado seguido de dos puntos ":" y del valor de encabezado. Cada nombre:valor debe estar separado del siguiente por un espacio sencillo " ".
4.2.4) Recortar la cadena para asegurarse de que no contenga ningún carácter de espacio al inicio o al final.

- Anexar un separador de línea, " " (Punto Unicode U+000A).
- Anexa la cadena de consulta con codificación URL que consta de la ruta (empezando por la primera barra inclinada) y los parámetros de consulta. La cadena de consulta no debe contener el nombre de host o el puerto y no debe contener ningún carácter de espacio como prefijo o sufijo.
- Solamente para peticiones POST o PUT, anexar un separador de línea, " " (Punto Unicode U+000A).

- Solamente para peticiones POST o PUT, serializar los parámetros de la petición de la siguiente forma, siendo el nombre del parámetro y su valor, la representación en UTF-8 de su codificación URL.
- Ordenar los parámetros por nombre de parámetro en orden alfabético ascendente y a continuación por valor de parámetro.
- Crear una cadena vacía. A continuación, para cada parámetro después de la ordenación, añadir a la cadena recién creada el nombre de parámetro seguido de un signo igual "=" y del valor del parámetro. Cada nombre=valor debe estar separado del siguiente por un ampersand "&".
- Recortar la cadena para asegurarse de que no contenga ningún carácter de espacio al inicio o al final.

Una vez que se haya creado la cadena siguiendo el proceso descrito más arriba, se debe firmar mediante el algoritmo **HMAC-SHA1** y el **secreto** obtenido al registrar la aplicación. Después de la firma, los datos binarios sin procesar se deben codificar en **base64**. La cadena resultante es la cadena **requestSignature** que añadir al encabezado Authorization.

El encabezado X-11Paths-Date

El encabezado **X-11Paths-Date** contiene el valor de la fecha UTC actual y debe tener el siguiente formato:

```
yyyy-MM-dd HH:mm:ss
```

donde yyyy es el año, MM es el número del mes, dd es el número del día, HH es la hora en formato de 24 horas, mm son los minutos y ss los segundos. Todos los valores se deben completar con ceros para que tengan valores de dos dígitos, excepto el año, que tiene cuatro. Es muy importante que el valor y el formato de este encabezado sean exactamente los mismos que los utilizados en el proceso de creación de **requestSignature** para el encabezado de autorización como se explica más arriba.

Ten en cuenta que puedes seguir utilizando el encabezado HTTP **Date** estándar en el formato que desees, por ejemplo RFC 1123. Solo asegúrate de no confundir ambos y de utilizar siempre el valor que utilices en **X-11Paths-Date** en el proceso de firma. La API ignorará el encabezado **Date** estándar.

3.3.2 SDK de Latch para PHP

El SDK de Latch para PHP está compuesto por los archivos Error.php Latch.php y LatchResponse.php. Este conjunto de clases permiten a los desarrolladores integrar Latch en sus aplicaciones. El SDK proporcionado por Eleven Paths, es software libre, que puede ser redistribuído y modificado bajo los términos de la GNU Lesser General Public License.

4 Pruebas

A continuación se detallan las pruebas realizadas una vez finalizado el desarrollo de la extensión de autenticación para comprobar que los mecanismos de gestión de errores funcionan correctamente y todas las funcionalidades descritas en la etapa de análisis son accesibles al usuario.

Título	***parear cuenta de usuario de Mediawiki con Latch***
Descripción	El usuario envía el token OTP generado desde la app móvil a través del formulario de preferencias de usuario de Mediawiki para parear su cuenta.
Resultado obtenido	cuenta pareada (correcto).

Tabla 4: Prueba pareo de cuenta

Título	***desparear cuenta de usuario de Mediawiki con Latch***
Descripción	El usuario realiza una petición de despareado de su cuenta a través del formulario de la web de Mediawiki.
Resultado obtenido	cuenta despareada (correcto).

Tabla 5: Prueba de despareo de cuenta

Título	***intento de login con latch abierto***
Descripción	El usuario intenta acceder a su cuenta de Mediawiki estando el latch asociado a esta abierto.
Resultado obtenido	usuario logeado (correcto).

Tabla 6: Prueba intento de acceso con pestillo abierto

Título	***intento de login con latch cerrado***
Descripción	El usuario intenta acceder a su cuenta de Mediawiki estando el latch asociado a esta cerrado.
Resultado obtenido	usuario no logeado y notificación en la app del teléfono (correcto).

Tabla 7: Prueba intento de acceso con pestillo cerrado

Título	***envío de código OTP incorrecto***
Descripción	El usuario envía un código OTP con formato incorrecto a través del formulario de la web de Mediawiki para parear su cuenta.
Resultado obtenido	no se realiza el pareado y la web muestra un mensaje de error (correcto).

Tabla 8: Prueba envío de código OTP incorrecto

Título	***respuesta de despareo de servicio desde Latch no afirmativa***
Descripción	El usuario envía una petición de despareo de su cuenta de Mediawiki desde el formulario de la web pero el servidor no puede procesar la petición por un fallo técnico o un corte en la conexión a internet.
Resultado obtenido	no se realiza el despareado de la cuenta y la web de Mediawiki muestra un mensaje de error informando al usuario (correcto).

Tabla 9: Prueba intento de despareo erróneo