UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

GESTOR WEB Y APP MÓVIL PERSONALIZABLE PARA PROFESIONALES DE MEDIANAS Y PEQUEÑAS EMPRESAS

Trabajo de fin de Grado

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

por

Xavier Vilella Muñoz

En cumplimiento de los requisitos para el grado de INGENIERÍA TELEMÁTICA

Tutor: Anna Calveras Augé

Barcelona, junio 2016

Resumen

El trabajo de fin de grado se basa en el desarrollo de una herramienta gestora para el negocio de pequeñas y medianas empresas, las cuales requieren de gran interactividad con su cliente.

La herramienta consta de una aplicación web para el profesional y una app móvil para el profesional y su cliente. La aplicación permitirá al profesional llevar a cabo todas las tareas de gestión de visitas, consultas, creación de promociones e historial de fichas de forma automática, así como también personalizar la app móvil que se descargará su cliente, de tal forma que dé la impresión a su cliente que la app que está utilizando no es genérica, sino que es propia de la empresa.

Todo ello debe desarrollarse en un bajo coste.

Resum

El treball de fi de grau es basa en el desenvolupament d'una eina per la gestió del negoci de petites i mitjanes empreses, les quals necessiten tenir una gran interactivitat amb el seu client.

L' eina consta d'una aplicació web destinada al professional i una app mòbil tant pel professional com pel seu client. L'aplicació permetrà al professional portar totes les tasques de gestió de visites, consultes, creació de promocions i un historial de les visites de forma automàtica, així com també podrà personalitzar l'app mòbil que es descarregarà el seu client. De tal forma que tingui la impressió de que l'app que està utilitzant és de la seva empresa i no una app genèrica.

Tot això s'ha de desenvolupar amb un baix cost.

Histórico de Revisiones

Revisión	Fecha	Razón
0	10/06/2016	Creación del documento
1	13/06/2016	Redacción punto 1, descripción del proyecto
2	17/06/2016	Redacción punto 3, tecnologías usadas
3	19/06/2016	Redacción punto 4 y 5, metodología de desarrollo y resultados
4	20/06/2016	Redacción punto 2, 6 y 7, aplicaciones similares, presupuesto y conclusiones
5	21/06/2016	Bibliografía, índices y resumen
6	22/06/2016	Corrección ortográfica
7	27/06/2016	Revisión final

LISTA DE DISTRIBUCIÓN DEL DOCUMENTO

Nombre	e-mail
Xavier Vilella Muñoz	vilellamunoz@gmail.com
Anna Calveras Augé	anna.calveras@entel.upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	27/06/2016	Fecha	27/06/2016
Nombre	Xavier Vilella Muñoz	Nombre	Anna Calveras Augé
Posición	Autor del proyecto	Posición	Tutor del proyecto

Índice

Listado de imágenes

Listado de tablas

1. Descripción del proyecto

1.1. Motivación

La motivación del proyecto viene dada por la dificultad de autónomos o pequeñas empresas, los cuales tienen un horario de consulta o visita, de organizar sus agendas y mantener un contacto fácil con sus pacientes.

Con el proyecto se pretende automatizar y solucionar las problemáticas siguientes:

- Evitar la necesidad de que el profesional deba estar 24h pendiente del teléfono para atender a sus pacientes, así como evitar llamadas cuando está atendiendo a otro paciente.
- Agenda automatizada y visible siempre tanto para el profesional como sus pacientes.
- Que el profesional pueda tener un registro de notas sobre sus pacientes siempre disponible.
- Canal de comunicación sencillo en el que profesional pueda crear campañas de descuento y mantener conversaciones con sus pacientes separando su línea personal de la profesional, es decir que no tenga que utilizar “whatsapp” o llamadas.
- Reducir el ausentismo de clientes recordándoles su visita.

1.2. Requisitos y Especificaciones

Portal web

En el portal web los profesionales deberán poder gestionar todos los puntos de su negocio de forma completa, cubriendo todos los posibles casos que se puedan dar.

El profesional podrá gestionar sus datos y personalizar la aplicación que posteriormente descargarán los clientes mediante su código único.

El profesional podrá gestionar las visitas que le van llegando, consultar historial de visitas o el perfil del cliente.

El profesional deberá poder gestionar toda su agenda mediante un calendario en el cual verá la situación actual de su negocio.

El profesional podrá crear promociones para impulsar la actividad de su negocio, las cuales llegarán en forma de notificaciones a los Smartphone de los clientes.

En función del tamaño de la empresa o el uso que el profesional desee, se deberán poder crear diferentes agendas independientes (por servicios ofrecidos, por empleados…).

App

Al ser una aplicación móvil se simplifican las funcionalidades que ofrece para hacerla más intuitiva y simple, cubriéndose los casos más importantes del negocio.

El profesional podrá llevar a cabo las tareas relacionadas con la gestión de visitas (aceptar o anular visitas), visualización de la agenda y creación de promociones simples, pero no la introducción de sus datos del negocio para la personalización de la app o creación de agendas, eso se llevará a cabo únicamente desde el portal web.

El cliente, previamente registrado, podrá gestionar sus reservas, así como crear nuevas directamente desde la app.

Con el sistema de notificaciones se avisará al cliente de nuevas promociones, recordatorio de visitas, o si la cita ha sido confirmada por el profesional.

En el momento de pedir una nueva visita podrá existir un formulario en el que indicar los motivos como por ejemplo la causa o los síntomas por los cuales se lleva a cabo la reserva. De esta forma el profesional tendrá la información de manera más rápida y ágil.

Finalmente, es necesario remarcar que la aplicación debe ser altamente personalizable y parametrizable para que cada profesional pueda crear la suya propia.

Seguridad

Al ser una aplicación con datos sensibles, se deben cumplir los siguientes apartados de la Ley Orgánica de Protección de Datos:

- *Identificación y autenticación:*
  - Cada usuario debe tener una identificación única basada en la combinación de identificador de usuario y contraseña.
  - La contraseña debe cancelar cada 180 días.
  - La contraseña debe almacenarse cifrada.
  - La contraseña debe cumplir los mínimos de complejidad (mínimo de 8 caracteres y utilizando mayúsculas y minúsculas)
- *Control de acceso:*
  - Los diferentes tipos de usuario deben tener limitados el acceso a la aplicación mediante permisos.
  - La sesión debe cerrarse automáticamente después de un cierto tiempo de inactividad.
- *Distribución de soportes*
  - Los emails que se envíen desde la plataforma deben estar cifrados.
- *Copias de seguridad y recuperación*
  - Cada día debe realizarse una copia de seguridad de todos los datos de carácter sensible existentes en la aplicación.
  - La copia de seguridad debe guardarse en un datacenter diferente al del servidor principal.
  - La copia de seguridad se debe almacenar cifrada.
- *Registro de accesos*
  - Se debe guardar como mínimo un registro de cada intento de acceso en la aplicación, así como todas las interacciones donde se traten datos sensibles.
  - Este fichero deberá guardarse durante 2 años.
- *Telecomunicaciones*
  - El acceso a la plataforma por parte de los usuarios debe realizarse bajo https.
  - Para evitar transferencia internacional de datos, se deberán utilizar datacenters dentro de la UE.

1.3. Objetivos

El objetivo del proyecto es crear una aplicación tanto web como móvil. El nombre de la aplicación será “MiHora”, el nombre es debido que la función principal tanto para profesionales como pacientes será la gestión de sus horas de visita. Si un profesional lee mi hora entiende que es su agenda con todos sus pacientes y reservas, en cambio sí lo lee un paciente entiende que es únicamente su hora que tiene reservada con el profesional.

Pese a que la gestión de reservas es la función principal de la aplicación, MiHora tiene que ser mucho más que un gestor de agendas. Por ello incorporará otras 5 características que le darán un gran valor añadido, estas son:

- Sistema de promociones: los profesionales contaran con un sistema para promocionarse en las horas que deseen, y llegando en forma de notificación a todos sus clientes de forma rápida y sencilla.
- Mensajería: incorporar un chat para poder mantener una conversación paciente-profesional.
- Pacientes: los profesionales no solo verán quien le ha reservado, sino que desde la misma app podrán mantener un historial clínico de cada paciente.
- Recordatorios: siempre que tengamos una reserva hecha, el día anterior nos llegara un aviso recordándonoslo.
- Personalización: cada profesional introducirá su logo, así como nombre y datos claves. De esta forma cada profesional y paciente que utilice mi hora, tendrá su propio mihora.

Aparte de las características para los usuarios, un gran objetivo es que sea fácilmente escalable para poder dar servicio a todos los clientes que quieran usarla.

Para hacer una aplicación más acorde a lo que necesita el usuario final, un objetivo durante todo el proyecto será conseguir beta/testers en el ámbito de la fisioterapia para ir desarrollando y obteniendo el feedback.

1.4. Plan de trabajo

<table>
<tr><td>Definición del proyecto</td><td colspan="2">PT: #1</td></tr>
<tr><td>Búsqueda y diseño</td><td colspan="2">Paquete 1 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se definirá la estructura del proyecto, las tecnologías que se utilizarán y las funciones que tendrá.</td><td colspan="2">Fecha inicial: 15/02/2016
Fecha final: 11/07/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T5</td></tr>
<tr><td>T1: Estudio del mercado
T2: Definir Arquitectura
T3: Buscar y escoger tecnologías
T4: Análisis de casos de uso
T5: Diseño de implementación</td><td>Entregables:</td><td>Fechas:
29/02/2016</td></tr>
</table>

Tabla 1 – Paquete de trabajo 1

<table>
<tr><td>Desarrollo de Entornos</td><td colspan="2">PT: #2</td></tr>
<tr><td>Desarrollo</td><td colspan="2">Paquete 2 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se crearán tres entornos dotados con servidor web y base de datos, preparados para subir el código del portal, almacenar datos y crear funciones de interconexión con la app.</td><td colspan="2">Fecha inicial: 01/03/2016
Fecha final: 21/06/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T3</td></tr>
<tr><td>T1: Entorno de desarrollo (local)
T2: Entorno de test (red)
T3: Entorno de producción (red)</td><td>Entregables:</td><td>Fechas:</td></tr>
</table>

Tabla 2 – Paquete de trabajo 2

<table>
<tr><td>Desarrollo portal fase 1</td><td colspan="2">PT: #3</td></tr>
<tr><td>Desarrollo software</td><td colspan="2">Paquete 3 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se desarrollarán las funciones básicas del portal.</td><td colspan="2">Fecha inicial: 01/03/2016
Fecha final: 14/04/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T3</td></tr>
<tr><td>T1: Estructura básica
T2: Funciones administrador
T3: Creación y modificación de agendas
T4: Ingreso, consulta y anulación de visitas
T5: Visualización de visitas en la agenda
T6: Aceptación de visitas ingresadas por el cliente
T7: Consulta perfil cliente</td><td>Entregables:
Portal fase 1 en el entorno de test</td><td>Fechas:
14/04/16</td></tr>
</table>

Tabla 3 – Paquete de trabajo 3

<table>
<tr><td>Desarrollo app fase 1</td><td colspan="2">PT: #4</td></tr>
<tr><td>Desarrollo software</td><td colspan="2">Paquete 4 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se desarrollarán las funciones básicas de la app.</td><td colspan="2">Fecha inicial: 07/03/2016
Fecha final: 14/04/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T4</td></tr>
<tr><td>T1: Estructura básica
T2: Gestión de cliente
T3: Visualización agenda del profesional
T4: Ingreso y anulación de visitas</td><td>Entregables:
App fase 1 en Google Play</td><td>Fechas:
14/04/16</td></tr>
</table>

Tabla 4 – Paquete de trabajo 4

Desarrollo portal fase 2 y 3	PT: #5	
Desarrollo software	Paquete 5 de 9	
Descripción: Se desarrollarán las funciones avanzadas del portal	Fecha inicial: 16/04/2016 Fecha final: 12/06/2016	
	Evento inicial: T1 Evento final: T2	
T1: Ingreso de promociones T2: Visualización en el calendario y modificación de las promociones T3: Restringir clientes de la promoción T4: Comunicación cliente-profesional	Entregables: Subidas portal fase en el entorno de test	Fechas: Al final de cada tarea

Tabla 5 – Paquete de trabajo 5

Desarrollo app fase 2 y 3	PT: #6	
Desarrollo software	Paquete 6 de 9	
Descripción: Se desarrollarán las funciones avanzadas de la app.	Fecha inicial: 23/04/2016 Fecha final: 16/06/2016	
	Evento inicial: T1 Evento final: T6	
T1: Ingreso de promociones T2: Visualización en el calendario y modificación de las promociones T3: Visualización promociones T4: Aceptación de visitas T5: Consulta de visitas T6: Ingreso, modificación y eliminación de promociones	Entregables: Subidas app al Google Play	Fechas: Al final de cada tarea

Tabla 6 – Paquete de trabajo 6

<table>
<tr><td>Feedback con Fisioterapeutas</td><td colspan="2">PT: #7</td></tr>
<tr><td>Recogida de información y correcciones</td><td colspan="2">Paquete 7 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se recogerá la opinión de los usuarios sobre el sistema y se modificarán funcionalidades y/o corregirán bugs</td><td colspan="2">Fecha inicial:
primera subida a test
Fecha final: 27/06/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T3</td></tr>
<tr><td>T1: Subidas al entorno de test
T2: Subidas al entorno de producción
T3: Corrección de errores</td><td>Entregables:</td><td>Fechas:</td></tr>
</table>

Tabla 7 – Paquete de trabajo 7

<table>
<tr><td>Sistema de notificaciones</td><td colspan="2">PT: #8</td></tr>
<tr><td>Integración api externa</td><td colspan="2">Paquete 8 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se adaptará una API externa para crear las notificaciones deseadas en el sistema.</td><td colspan="2">Fecha inicial: 28/04/2016
Fecha final: 20/05/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T2</td></tr>
<tr><td>T1: Notificaciones globales
T2: Notificaciones a un grupo</td><td>Entregables:</td><td>Fechas:</td></tr>
</table>

Tabla 8 – Paquete de trabajo 8

<table>
<tr><td>Segurizacion de la información</td><td colspan="2">PT: #9</td></tr>
<tr><td>Seguridad</td><td colspan="2">Paquete 9 de 9</td></tr>
<tr><td rowspan="2">Descripción:
Se aplicarán las configuraciones para adaptar la base de datos y la aplicación a cifrados para la información sensible.</td><td colspan="2">Fecha inicial: 09/05/2016
Fecha final: 25/05/2016</td></tr>
<tr><td colspan="2">Evento inicial: T1
Evento final: T3</td></tr>
<tr><td>T1: Backups
T2: Cifrado de datos sensibles
T3: HTTPS
T4: Registro de acciones</td><td>Entregables:</td><td>Fechas:</td></tr>
</table>

Tabla 9 – Paquete de trabajo 9

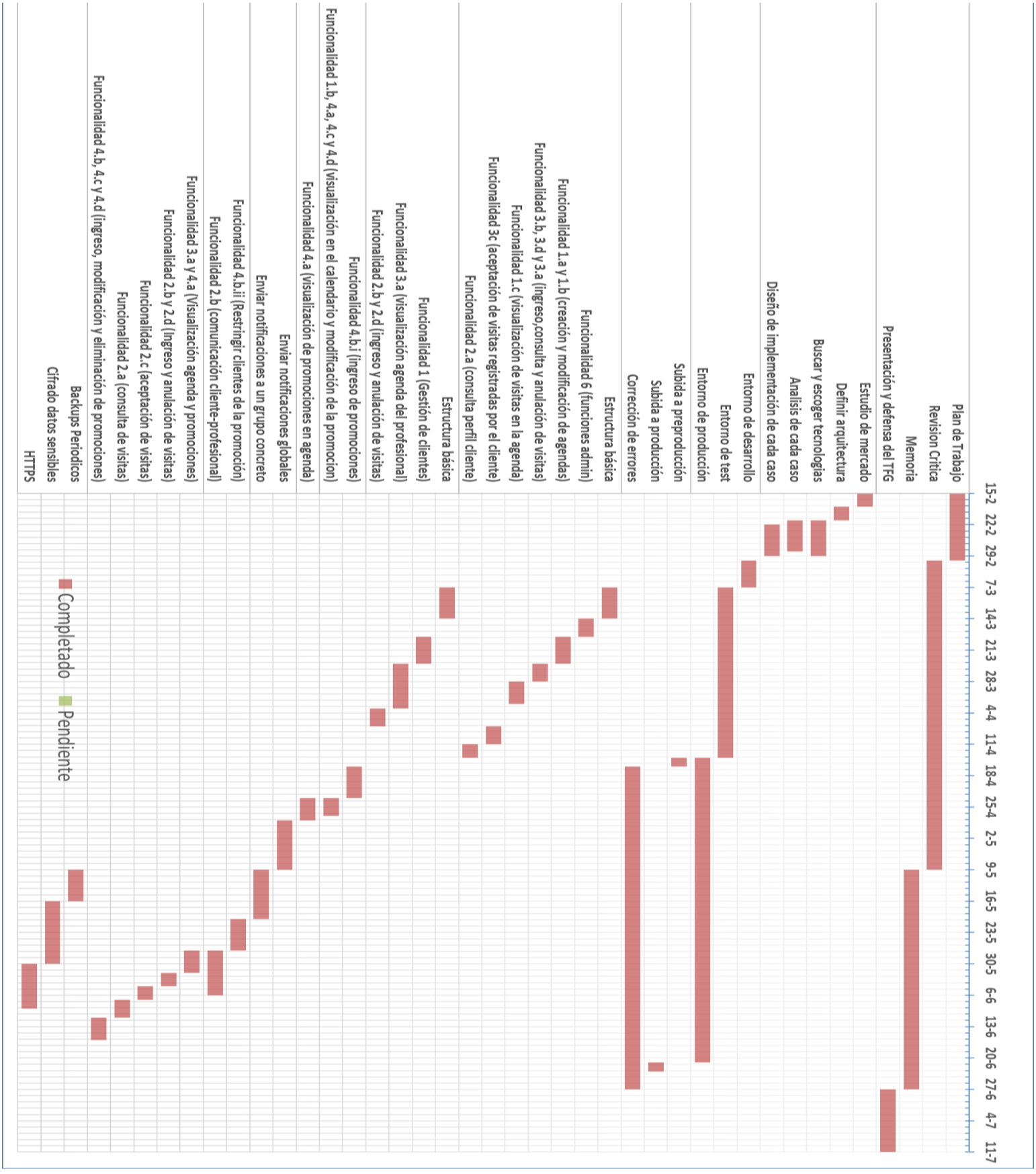


Ilustración 1 - Gantt

1.5. Desviaciones del plan de trabajo

Inicialmente no analicé bien el tema de la seguridad de la información, por ello se tuvo que actualizar el plan de trabajo pensado inicialmente donde no estaba incluido.

Al ser un tema sumamente importante para poder cumplir las leyes de protección de datos, lo prioricé por delante de la tarea de sincronización del calendario con Google Calendar, pasando ésta a una función adicional una vez terminado el proyecto.

En este tema de seguridad de datos, se incluyen las siguientes tareas:

- Creación de backups periódicos y cifrados, como mínimo de toda la información sensible.
- Cifrado en Base de datos de toda la información sensible.
- Comunicación cliente-servidor en HTTPS.
- Creación de un registro de acceso y acciones realizadas por los usuarios.

2. Aplicaciones similares:

Existen varias aplicaciones que ofrecen características similares, quizás la más conocida es Bucmi (https://www.bucmi.com). La cual ha conseguido más de 50 mil descargas y financiación externa. Pero no es competencia directa ya que, pese a que en el fondo su finalidad es la reserva de citas para centros de belleza, su negocio se basa en un buscador como podría ser “Atrapalo”.

Por lo que a quien les compran la suscripción para aparecer en Bucmi pretenden conseguir aumentar las reservas y obtener más beneficios a partir de los usuarios que tiene Bucmi.

La finalidad de MiHora no es esa, sino que lo que se pretende es darles a las pequeñas y medianas empresas su propia aplicación. Una aplicación con la cual puedan organizar mejor su negocio y dar facilidades a sus clientes para fidelizarlos. Dar el prestigio de poder decir “resérvame desde mi app” y todo ello por un precio muy inferior al que significa desarrollar la app del negocio.

Por eso haremos la comparativa con las aplicaciones Bookitit (http://www.bookitit.com/es) y Bookeo que son más parecidas a MiHora.

Tanto Bookitit como Bookeo son un sistema que permite implementar la creación de reservas desde la web o Facebook del negocio, mediante la instalación de la libreria.

A continuación, tenemos la tabla comparativa entre las tres aplicaciones:

	MIHORA	BOOKITIT	BOOKEO
WEB DE GESTIÓN DEL PROFESIONAL	Si	Si	Si
APP DE GESTIÓN DEL PROFESIONAL	Si	Si	No
WEB DE RESERVA PARA EL CLIENTE	No	Si	Si
APP DE RESERVA PARA EL CLIENTE	Si (con su logo)	No	No
INTEGRACIÓN EN LA WEB DE LA EMPRESA	No	Si	Si
INTEGRACIÓN EN FACEBOOK	No	Si	SI
RECORDATORIOS	Móvil Push (con su logo)	Email,SMS	Email,SMS
INTRODUCCIÓN DEL MOTIVO DE LA RESERVA	Si	No	Si
PAGO ONLINE	No	Si	Si
MÚLTIPLES AGENDAS	Si	Si	Si
VALIDACIÓN DE LA RESERVA	Si	Si	No
COMUNICACIÓN CON EL CLIENTE DESDE LA APP	Si	No	No
CREACIÓN DE PROMOCIONES	Si	No	Si
LISTADO DE PACIENTES	Si	No	No
HISTÓRICO CLÍNICO	Si	No	No

Tabla 10 – Comparativa Aplicaciones

Después de la comparación podemos ver como los puntos fuertes donde se está por encima de la competencia directa son sobretodo dos:

- App móvil, es el punto más fuerte de MiHora. La gran personalización de MiHora al introducir el código del profesional la convierte en la app del profesional, viendo siempre su logo y la personalización que haya introducido el centro.
- El otro punto fuerte frente a la competencia es la comunicación con el paciente, mucho más ágil con las nuevas tecnologías, y la posibilidad de tener un historial clínico del paciente.

Por el contrario, se sale perdiendo en:

- Por el momento no se puede integrar un widget de MiHora en el Facebook o página web de los centros.
- No permite pagos online.

Analizando las fortalezas y debilidades el lema de MiHora será:

¡Obtén tu propia App!

Ya que es de alta importancia resaltar que es la más personalizable y con la que consigues la experiencia más cercana a tener una aplicación propia.

3. Tecnologías utilizadas:

Para la realización del proyecto, se han integrado multitud de tecnologías para crear la aplicación final. Entre ellas se encuentran diversos lenguajes de programación, Frameworks, con los que dotamos de una estructura las aplicaciones y librerías de terceros para añadir algunas funcionalidades extras.

En el cuadro resumen siguiente podemos ver en qué parte del proyecto se ha utilizado cada una de ellas, y posteriormente tenemos una breve explicación de su utilidad.

Tabla 11 – tecnologias utilizadas

3.1. Lenguajes de programación

3.1.1. PHP

Lenguaje de código abierto, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. El código es ejecutado en el servidor, generando HTML y enviándolo al cliente.

Básicamente utilizado para dotar de inteligencia a la app, dentro del modelo MVC es el encargado del controlador y obtener los datos de la base de datos.

3.1.2. HTML

Lenguaje de marcas para insertar el contenido de una página web, como texto, imágenes, videos, juegos, entre otros.

Básicamente utilizado para las vistas.

3.1.3. CSS

Lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML. Con ello se consigue separar el estilo y definirlo en un documento separado del HTML.

Básicamente utilizado para dotar de un buen aspecto visual a la app.

3.1.4. JavaScript

Lenguaje utilizado en el lado del cliente, permitiendo una mejor interfaz de usuario y que la página web sea dinámica.

3.1.5. MySQL

MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL).

3.2. Entornos y herramientas de desarrollo

3.2.1. Netbeans

Entorno de desarrollo gratuito de código abierto que permite el uso de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o dispositivos móviles.

3.2.2. MySQL Workbench

Herramienta visual para la administración, creación y mantenimiento de bases de dato MySQL.

3.2.3. Yii2

Yii es un framework orientado a objetos, software libre, de alto rendimiento basado en componentes, para desarrollar aplicaciones Web de gran escala. Es un framework MVC (modelo-vista-controlador). Permite el acceso a base de datos mediante “Database Access Objects” (DAO)

3.2.4. Apache Cordova / Phonegap

Es un framework para el desarrollo de aplicaciones para sistemas operativos móviles, haciendo uso de tecnologías web como HTML, CSS y JavaScript. Entre otros permite desarrollar aplicaciones para Android y iOS.

3.3. Código de terceros

3.3.1. Jquery

Librería JavaScript la cual permite emplear código transversalmente entre los distintos navegadores.

3.3.2. Bootstrap

Librería que permite dotar de un diseño adaptable a las diferentes resoluciones de pantalla.

3.3.3. Fullcalendar

Plugin JavaScript que permite crear un calendario. Código libre y personalizable.

3.3.4. Google Cloud Messaging + OneSignal

Google Cloud Messaging (GCM) es un servicio gratuito de google que permite enviar y recibir mensajes entre servidores y clientes.

Además, con OneSignal conseguimos una interfaz visual para enviar mensajes a grupos o segmentos determinados.

4. Metodología y desarrollo:

4.1. Entornos

El proyecto consta de una aplicación web como una móvil, por lo tanto, se ha tenido que crear una parte común de la inteligencia y datos, y otra propia para cada sistema.

Y al estar constantemente en contacto con beta/testers que han probado los avances de la aplicación, se necesitaban tanto un entorno de desarrollo como otro más estable. A continuación, se describen cada uno de ellos, así como las herramientas utilizadas para programar en cada uno de ellos.

4.1.1. WEB

Para llevar a cabo las tareas de programación, lo primero fue crear dos entornos uno de desarrollo y otro de producción.

Ambos entornos están hospedados en el mismo hosting web, pero bajo una ruta distinta.

https://mihora.es/app-development/web

https://mihora.es/app/web

Ambos entornos tienen un esquema asociado a la base de datos MySQL, con sus respectivos usuarios aplicativos con los permisos restringidos.

Para las subidas de código al entorno de desarrollo, el proyecto está sincronizado mediante FTP y netbeans, es decir los cambios guardados en local se suben automáticamente para poder testearlos desde la URL de desarrollo.

Para las subidas de código al entorno de producción, una vez validados los cambios en desarrollo se transfieren los ficheros mediante FTP con FileZilla.

En cuanto los cambios en Base de Datos, utilizando un usuario de administración y MySQL Workbench, se tiene acceso a los dos esquemas de desarrollo y producción. De esta forma se puede ir desarrollando en el esquema de desarrollo y transferirlo a producción una vez validado.

4.1.2. APP

Para las tareas de programación de la app móvil, al utilizar Apache Cordova, para testear los cambios he utilizado tanto la aplicación de escritorio de PhoneGap de escritorio como el móvil en modo desarrollador.

Al utilizar la aplicación de PhoneGap ganas en velocidad a la hora de desarrollar ya que lo puedes testear directamente desde el navegador web, teniendo así disponibles todas las herramientas de desarrolladores por ejemplo de Mozilla Firefox. Por contra si necesitas testear alguna función que necesita algún api del teléfono con el navegador no es suficiente.

Esto sería el equivalente al entorno de desarrollo de la web. En cuanto los cambios están bien testeados en local, el siguiente paso ha sido compilar la aplicación para subirla en fase alfa o beta en Google Play y posteriormente de testearla promocionarla a producción.

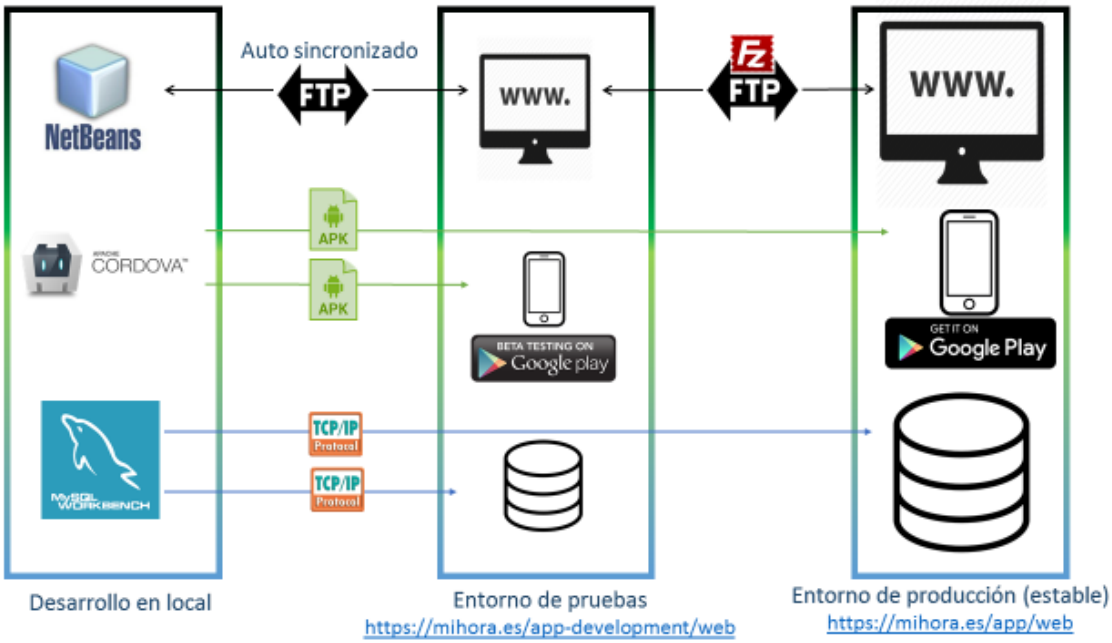


Ilustración 2 – metodologia de desarrollo

4.2. Elementos

Al ser un proyecto que se utilizan muchos elementos distintos interconectados entre ellos, a continuación, se describe con un esquema visual la arquitectura de la aplicación, para posteriormente describir en detalle cada uno de los elementos.

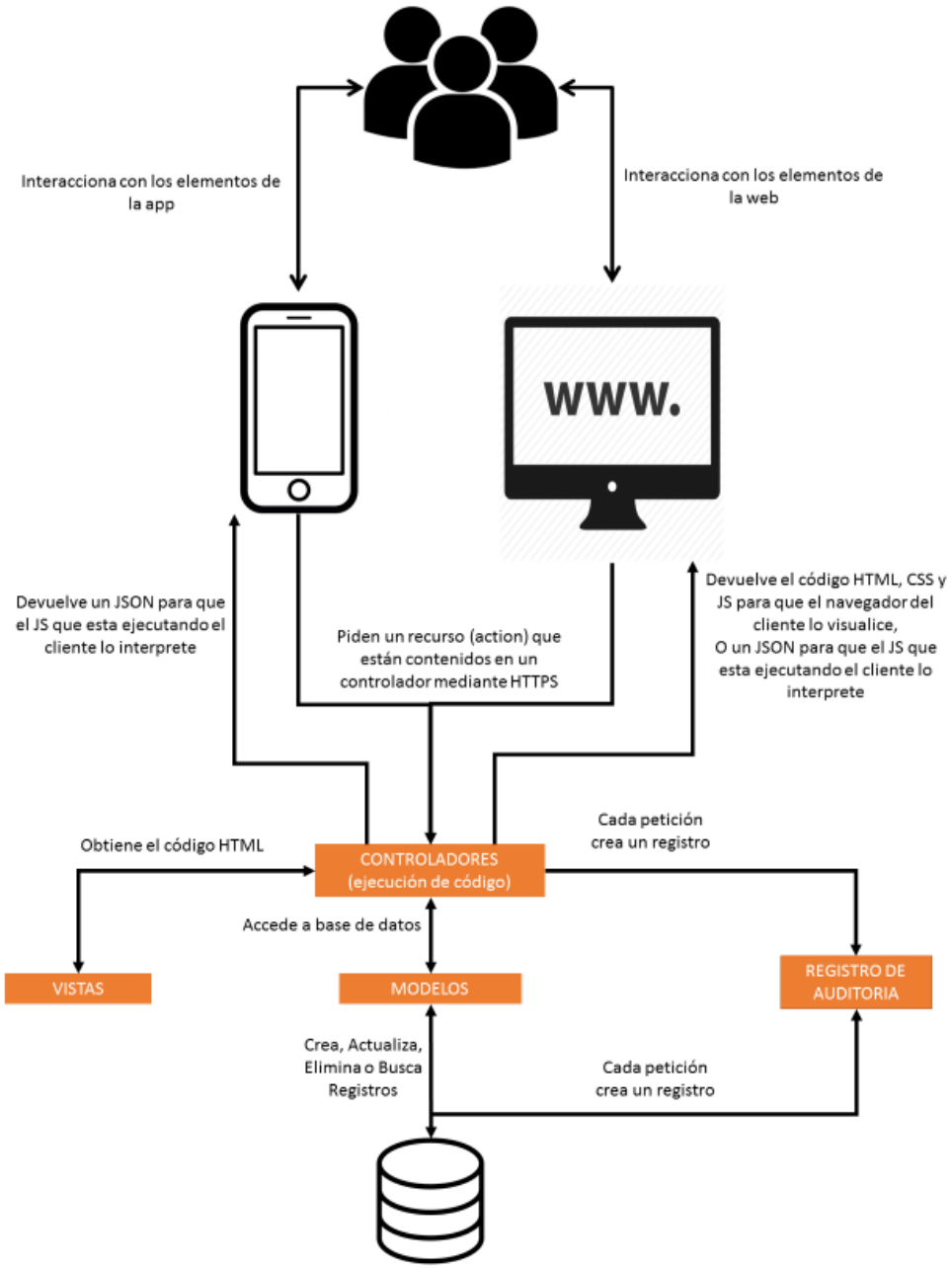


Ilustración 3 – esquema de elementos

4.2.1. Modelos

Cada tabla que he creado dentro de la base de datos (usuarios, mensajes, permisos, promociones…) tiene su modelo de datos asociado en el proyecto web.

Cada modelo es un fichero PHP, situado bajo la carpeta *models*, y desde el cual se hacen las interacciones con una tabla de la base de datos. Selects, inserts, updates y deletes, son las acciones SQL que se pueden realizar, ya que el usuario que utiliza para conectarse le he eliminado todos los demás permisos, como por ejemplo los de crear tablas.

Cada modelo es una clase que extiende el componente de \yii\db\ActiveRecord para obtener las funciones auxiliares que implementa el sistema ActiveRecord. Las ventajas de este sistema las veremos a continuación.

4.2.1.1. Búsqueda de registros (find)

Para realizar una búsqueda sencilla (select) en MySQL como podría ser:

```
select * from usuarios where username like ‘Xavier Vilella Muñoz’ and role in (
‘centro’)
```

Con el formato de ActiveRecord no escribimos el código MySQL, sino que su equivalente sería:

```
self::find()
→ where([ 'username' => $username ])
→ andWhere ([ 'in' , ' rol_id ' , [‘centro’] ] )
→ one();
```

Haciendo todas las querys de esta forma aparte de ahorrarse escribir código MySQL, el propio framework se encarga de evitar algunos posibles ataques como SQL injection.

Y en cuanto a la consulta, nos permite acceder a los campos de manera más sencilla, por ejemplo, si quisiéramos ver el nombre del usuario que hemos buscado bastaría con:

```
$resultado→nombre
```

4.2.1.2. Creación y actualización de registros (save)

Para crear un registro (insert) en MySQL la estructura es:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Y para realizar una actualización (update):

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Como vemos en MySQL tenemos una estructura distinta para hacer un insert o un update, por lo que debemos ser nosotros quien se encargue de averiguar si el campo ya existe o no y hacer una función u otra.

Con el formato ActiveRecord, para guardar un registro en la base de datos se hace de la siguiente forma:

Asignamos sus atributos:

```
$model→nombre = 'Xavier Vilella Muñoz'
```

Una vez asignados lo guardamos:

```
$modelo→save();
```

Tanto el insert como el update se hace con la función save, la diferencia está en cómo hemos creado el objeto modelo.

Si lo creamos de nuevo:

```
$modelo = new Usuarios();
```

Se hará un insert. En cambio si el modelo lo hemos obtenido de hacer un find() la acción que se llevará a cabo será un update.

Pero la función más importante que nos da el ActiveRecord, es que siempre que se llama la función save, antes de realizar un insert o un update se pasan unas reglas de validación. Estas reglas se definen en el mismo modelo de datos con la función rules. En ella se puede definir el formato de cada campo, si es requerido, el tamaño máximo… y si alguna de ellas no pasa la comprobación no se realizará el cambio.

```
public function rules()
    {
        return [
            [['username'], 'unique'],
            [['rol_id', 'status'], 'integer'],
            [['username'], 'string', 'max' => 255],
            [['password'], 'string', 'max' => 60],
            [['expire_session'], 'string', 'max' => 45],
            [['mail'], 'string', 'max' => 60]
        ];
    }
```

4.2.1.3. Eliminar un registro (delete)

La forma de eliminar un registro se hace de la misma forma que para actualizar. Se consigue el modelo del registro que queremos eliminar mediante un find(), y

posteriormente sobre el modelo, para actualizar utilizábamos save(), para eliminarlo se utiliza delete().

4.2.2. Registro de acciones (Base de Datos)

El registro de acciones, es un log de todas las acciones que se han hecho en la base de datos. Con este registro en caso de existir un ataque o acciones indebidas dentro de la aplicación, quedará registrado.

Para implementarlo he creado un componente propio, BaseActiveRecord, que extiende \yii\db\ActiveRecord y de esta forma todos los modelos pasan a extender mi componente sin perder las funcionales que teníamos de ActiveRecord.

En el componente BaseActiveRecord he implementado las funciones *afterFind, afterSave y afterDelete*. Cada una de ellas se llama en función de la acción que va a realizar el usuario, afertFind antes de un find, afterSave antes de un save y afterDelete antes de un delete. Lo que hacen todas ellas es crear un nuevo registro con la información de id aplicativo del usuario, dirección ip, modelo que se está accediendo, acción que se está realizando, atributos consultados, fecha de la acción y resultado (permitido y denegado).

En la siguiente imagen se puede ver la tabla del registro de acciones.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	id_user	ip	model	action	attributes	fecha	resultado
957	16	88.20.48.117	app\models\Mensajes	SAVE	á(□hcgzÇA+aÑ_Eac2a6bd6fa5d2f3c3e107c6d...	2016-06-16 17:33:09	PERMITIDO
958	16	88.20.48.117	app\models\Mensajes	SAVE	·Û1ëzc]¨çÐ,□zqï-22887d18033958937d30353...	2016-06-16 17:33:09	PERMITIDO
959	16	88.20.48.117	app\models\Mensajes	SAVE	ª:Íq□□â^y□□□\eÚ17f503e74487c9d60b49...	2016-06-16 17:33:09	PERMITIDO
960	16	88.20.48.117	app\models\Mensajes	SAVE	□□_□□ ïRpÙ¶©Å1e04b7cac038692a7680df...	2016-06-16 17:33:09	PERMITIDO
961	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:33:10	PERMITIDO
962	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:33:11	PERMITIDO
963	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:33:16	PERMITIDO
964	16	88.20.48.117	app\models\Reservas	UPDATE	28<>16<>17<>24<>2016-06-17 11:00:00<...	2016-06-16 17:33:16	PERMITIDO
965	16	88.20.48.117	app\models\Mensajes	SAVE	ô÷□Õê× ælhì®□12b6cf16525e9ff76fd4c884...	2016-06-16 17:33:16	PERMITIDO
966	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:33:17	PERMITIDO
967	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:37:57	PERMITIDO
968	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:37:59	PERMITIDO
969	16	88.20.48.117	app\models\Cuentas	UPDATE	16<>Hidalgo Esport i Salut<>$2y$13$K67y2Us...	2016-06-16 17:38:05	PERMITIDO

Ilustración 4 – Registro de acciones

4.2.3. Datos sensibles

Debido que en la aplicación hay datos considerados de carácter sensible, estos deben guardarse cifrados, o un hash del mismo.

4.2.3.1. Hash

Utilizado en la contraseña del usuario. Para crear el hash utilizo las funciones de seguridad que nos da el Yii.

```
$hashDelPassword = Yii::$app→getSecurity()→generatePasswordHash($password);
```

Realizar un hash significa que la única forma de averiguar cuál es su valor en claro es la fuerza bruta, probando todas las combinaciones posibles hasta dar con la buena. Como las contraseñas no necesitamos descifrarlas en claro, únicamente debemos comprobar si coinciden, la forma más segura es almacenar el hash en base de datos y cuando el usuario quiere entrar no se comprobarán las contraseñas en claro, sino que se comparan los hashes.

Para realizar la comprobación del password del usuario con el hash utilizo la siguiente función:

```
Yii::$app→getSecurity()→validatePassword($passwordIntroducido, $hashAlmacenado);
```

4.2.3.2. Cifrado

Para el resto de datos que es necesario recuperarlos en claro, utilizo las funciones encryptByKey y decryptByKey de yii2. Aparte para guardarlo en la base de datos después de encriptarlo se codifica a utf8 para evitar tener caracteres conflictivos.

```
CIFRAR: utf8_encode(Yii::$app->security->encryptByKey($valor, $clave));

DESCIFRAR: Yii::$app→security→decryptByKey(utf8_decode($valor), $clave);
```

4.2.4. Copias de seguridad

Mediante un CRON [1]que se ejecuta cada día a las 23:00, este ejecuta una llamada a la URL http://www.mihora.es/app/web/mihora/backupdb

Ejecutándose la función que primero obtiene todas las tablas del esquema, para posteriormente obtener todos los registros y crear un json. A partir del json se cifra y se guarda en un fichero de texto, para acabar enviándolo por mail a a dirección backups@mihora.es para poder tener el respaldo de la base de datos por si fuera necesario recuperar algún estado anterior.

A continuación podemos ver el código que se ejecuta para la copia de seguridad.

[1] Cron jobs o tareas programadas: ejecutan tareas repetitivas programadas por el usuario de manera automática.

```php
public static function actionBackupdb(){

    $tablas = Yii::$app->db->createCommand("SELECT table_name FROM INFORMATION_SCHEMA.tables WHERE TABLE_SCHEMA='mihora-dev'")->queryAll();

    $array = [];
    foreach($tablas as $tabla){
        $array [$tabla['table_name']] = Yii::$app->db->createCommand("SELECT * FROM ".$tabla['table_name'])->queryAll();
    }

    $json = AppHelpers::encryptString(json_encode($array));

    $fecha = date('Ymd');
    $file = fopen("../backup/backup-$fecha.txt", "w");
    fwrite($file, $json);
    fclose($file);

    $message = Yii::$app->mailer->compose()
            ->setFrom('info@mihora.es')
            ->setTo('backups@mihora.es')
            ->setSubject('Backup');

    $message->attach("../backup/backup-$fecha.txt");

    $message->send();
}
```

Ilustración 5 – Función de copias de seguridad

4.2.5. Controladores

Cada controlador tiene sus vistas asociadas dentro del directorio views, y está compuesto por un grupo de funciones (actions), y otras funciones auxiliares.

Todas estas funciones se encargan de capturar las acciones del usuario, interactuar con los modelos para obtener los datos solicitados, y finalmente delegárselos a las vistas para mostrarlos de nuevo al usuario, creando así una página web dinámica.

4.2.5.1. Resultado de la llamada a una action

En el proyecto existen solamente dos tipos de resultado al llamar a una action.

4.2.5.1.1. HTML

En este caso se renderizará una página web entera, normalmente es como resultado que el usuario ha cambiado de bloque dentro de la aplicación (calendario, mensajes, pacientes…).

4.2.5.1.2. JSON

En este caso la llamada a la action se realiza a través de javascript, obteniendo como resultado un json con el contenido a actualizar de una página que ya está visualizando el usuario.

4.2.5.2. BaseController

De la misma forma que los modelos extendían \yii\db\ActiveRecord, los controladores deben extender el componente \yii\web\Controller pero como en el caso anterior, me he creado mi propio BaseController para añadirle alguna funcionalidad al Controller por defecto de Yii2.

Dicha funcionalidad se trata del control de acceso de los usuarios a la aplicación. Para ello he implementado la función beforeAction(). Esta función se llama siempre antes de codigo en cualquier action del controlador. Y lo que realiza es mira a que URL estamos intentando acceder, acto seguido comprobamos si es una URL sobre la cual se pueda entrar sin estar logeado (como podría ser la pantalla de login).

```
$operacion = str_replace("/", "-", Yii::$app->controller->route);
if (in_array($operacion, $permitirSiempre)) {
            LogActions::insertlog('PERMITIDO SIEMPRE');
            return true;
}
```

Si se está intentando acceder a una URL que es necesario estar logeado pero no lo estamos o la sesión ha caducado, nos re-direcciona a la página de login.

```
if (Yii::$app->user->isGuest) {
            LogActions::insertlog('DENEGADO NO LOGUEADO');
            return $this->goHome();
}
if(!Cuentas::isSesionActive()){
            LogActions::insertlog('DENEGADO SESION CADUCADA');
            Yii::$app->user->logout();
            return $this->goHome();
}
```

En caso de querer entrar a una URL y cumplimos los requisitos anteriores, miramos si tenemos el permiso específico para entrar en dicha URL

```
if (!AccessHelpers::getAcceso($operacion)) {
            echo $this->render('/site/nopermitido');
            LogActions::insertlog('DENEGADO');
            return false;
}
```

La función getAcceso, básicamente lo que hace es ejecutar la siguiente query con los parámetros de la acción que requerimos, y nuestro rol de usuario. En función de eso la

query nos devolverá un resultado si tenemos permiso o no devolverá nada si carecemos de él.

```
SELECT o.nombre
FROM cuentas u
JOIN rol r ON u.rol_id = r.id
JOIN rol_operacion ro ON r.id = ro.rol_id
JOIN operacion o ON ro.operacion_id = o.id
WHERE (o.nombre =:operacion OR o.nombre =:all_controller)
AND u.rol_id =:rol_id
```

En caso de que todas las pruebas hayan sido satisfactorias se determina que se tiene acceso al módulo y por lo tanto se continua con el código de la action. En caso de que en este último paso nos devolviera que no tiene acceso se re direccionaría a una página de acceso denegado.

4.2.5.3. Roles en la base de datos

Como hemos visto en el último punto se realiza una búsqueda de si el usuario tiene permiso o no para entrar en determinadas URLs.

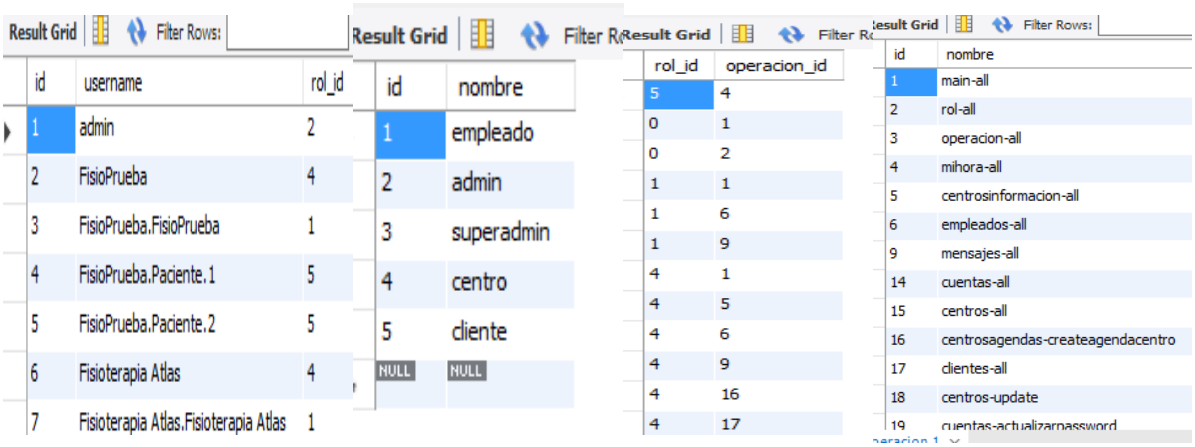
Para ello en la tabla “cuentas” cada usuario tiene asignado un rol mediante un identificador.

A modo informativo en la tabla “rol” tenemos la explicación de a que equivale cada id con un nombre.

En “rol_operacion” tenemos cada rol a que operaciones (o actions) puede entrar, esta relacion se hace mediante ids.

Para finalizar en la tabla “operaciones” tenemos todas las operaciones (o actions) que estan definidas en la aplicación, con su identificador para poder asignarselo a los roles.

De esta forma cada usuario tiene un rol asignado, donde cada rol tiene unas actions asignadas que puede realizar.



id	username	rol_id
1	admin	2
2	FisioPrueba	4
3	FisioPrueba.FisioPrueba	1
4	FisioPrueba.Paciente.1	5
5	FisioPrueba.Paciente.2	5
6	Fisioterapia Atlas	4
7	Fisioterapia Atlas.Fisioterapia Atlas	1

id	nombre
1	empleado
2	admin
3	superadmin
4	centro
5	cliente
NULL	NULL

rol_id	operacion_id
5	4
0	1
0	2
1	1
1	6
1	9
4	1
4	5
4	6
4	9
4	16
4	17

id	nombre
1	main-all
2	rol-all
3	operacion-all
4	mihora-all
5	centrosinformacion-all
6	empleados-all
9	mensajes-all
14	cuentas-all
15	centros-all
16	centrosagendas-createagendacentro
17	clientes-all
18	centros-update
19	cuentas-actualizarpassword

Ilustración 6 – Tablas de roles

4.2.6. Registro de acciones (Vistas)

Igual que en el caso de consulta y acceso a base de datos guardábamos un registro, en este caso es idéntico. Para cada llamada a una action al ejecutar el código del BaseController se guarda un registro de la URL consultada, por quien y si ha obtenido acceso o no.

4.2.7. Vistas

Las vistas son quien genera el html para la visualización en el navegador del cliente. Son ficheros php con código html incrustado y desde ellas, si fuera necesario, se añaden las rutas de css o js.

4.2.8. Mails

Para enviar mails utilizo el componente de Yii “swiftmailer”. Para poder utilizarlo primero debemos configurarlo en el fichero web.php del directorio config.

La particularidad en la configuración es el campo ‘useFileTransport’. Es utilizado para si queremos enviar mails de prueba. Si lo ponemos en true, no hará caso de toda la configuración de ‘transport’ y de esta forma cada vez que utilicemos el swiftmailer se enviará el mail a un fichero.

El fichero de configuración es el siguiente.

```
'mailer' => [
            'class' => 'yii\swiftmailer\Mailer',
'useFileTransport' => false,
            'transport' => [
                'class' => 'Swift_SmtpTransport',
                'host' => 'smtp.mihora.es',
                'username' => info@mihora.es',
                'password' => XXXXXXXX,
                'port' => '587',
                'encryption' => 'tls'
            ]
        ],
```

Una vez configurado, lo utilizamos de la siguiente forma:

```
Yii::$app→mailer
→compose( ‘LAYOUT’,[PARAMETROS])
→setTo($mailto)
```

```
->setFrom(['info@mihora.es' => 'mihora'])
->setSubject('bienvenido a mi hora')
->send();
```

Donde LAYOUT es la ruta a la vista del email, es decir el equivalente a las vistas para los controladores. El layout es el fichero que crea el HTML que se le enviará al cliente. El layout es un fichero php que no tiene por qué ser estático, de esta forma le podemos pasar los parámetros como un array para poder acceder a ellos después desde el php y crear el mail.

4.2.9. Interacción desde la app

La interacción desde la app es mucho más sencilla, ya que toda la inteligencia y acceso a base de datos, se utiliza el mismo código creado para la web.

De esta forma desde el móvil lo que tengo es un único html, esto es posible gracias a jquery. Donde cada pantalla está delimitada por un div con data-role="page" y un identificador.

Por ejemplo, la pantalla de reservar tiene el siguiente código:

```
<div data-role="page" id="reservar">
    <div  data-role="header" data-position="fixed"  data-tap-toggle="false" class="headers_top">
        <a onClick="returnToAgenda()" class="header_icons_2">Cancelar</a>
        <h2>Reservar Hora</h2>
    </div>
    <div class="logo_mihora"> <img id="img-logo-reservar" src="img/logo_lletra.png" height="80"> </div>
    <div id="promocion_info" class="info_horas_reserva">
        <h4    id="reservar_promocion" ></h4>
    </div>
    <div class="info_horas_reserva">
        <div class="info_horas_reserva_fecha" id="info_horas_reserva_fecha"> Lunes 6 de Mayo 2016</div>
        <div class="info_horas_reserva_hora" id="info_horas_reserva_hora"> de 12:00 a 12:30</div>
    </div>
    <div class="div_motivo">
        <font class="text_motivo_label">Indique el motivo</font>
        <textarea  id="reservar_motivo" class="textarea_motivo" ></textarea>
    </div>
    <div data-role="footer" onclick="confirmarReserva()" class="footer_reserva" data-position="fixed"  data-tap-toggle="false">
        <h2>CONFIRMAR RESERVA</h2>
    </div>
</div>
```

Ilustración 7 – Código reservar (app)

El contenido de las pantallas no es estático, se actualiza a partir de los resultados obtenidos del servidor mediante JavaScript.

Para obtener los resultados del servidor, me he creado la función *getValuesFromServer*.

```
function getValuesFromServer(action, parameters) {

    var respuesta = null;

    var request = new XMLHttpRequest();

    request.open("GET", "https://mihora.es/app/web/mihora/" + action + "?" + pa
rameters, false);

    request.onreadystatechange = function () {

        if (request.readyState == 4) {
```

```
            if (request.status == 200 || request.status == 0) {
                respuesta = JSON.parse(request.responseText);
            }
        }
    }
    request.send();
    return respuesta;
}
```

Su función es la de canalizar todas las peticiones que se dirigen al servidor y sus respuestas, que en este caso son siempre JSON. Siempre se llama al controlador mihora de la aplicación web, donde están las diferentes actions para las demandas de la app. Estas actions son las que interactúan con los mismos modelos que se utilizan para la versión web.

4.2.10. Plugins app

Al utilizar Apache Cordova o PhoneGap, se pueden instalar plugins de terceros para su utilización posterior en la app.

Para instalarlos, desde la línea de comandos situándonos en el directorio del proyecto, se utiliza el siguiente comando:

```
cordova plugin add nombre-del-plugin
```

4.2.10.1. Shortcut (Android)

Este plugin es el que permite crear el acceso directo en la pantalla home de Android con el nombre y logo del profesional.

Para utilizarlo, una vez instalado se llama desde JavaScript de la siguiente forma:

```
window.plugins.Shortcut.CreateShortcut({
text: NOMBRE,
icon: LOGO}, succescreate, failcreate);
```

Donde NOMBRE, es un string con el nombre del centro o profesional obtenido del servidor, y LOGO es la imagen en formato Base64.

4.2.10.2. Notificaciones (ANDROID y iOS)

Para poder utilizar las notificaciones, he creado un proyecto en Google Cloud Messaging (quien envía los mensajes) y OneSignal (el intermediario que nos permite la integración con cordova).

Ambos son servicios gratuitos, y al crear los proyectos obtenemos un identificador del proyecto de google, y otro para OneSignal.

Una vez creados e instalado el plugin en cordova ya podemos usarlo

4.2.10.2.1. Inicializar y obtener Ids

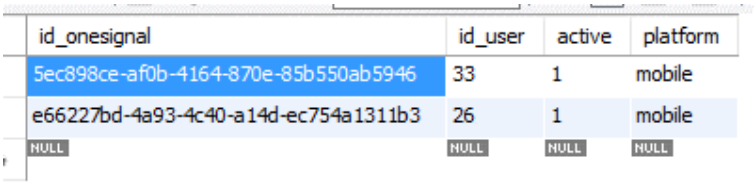
Inicializamos el plugin de OneSignal con nuestros indetificadores

```
window.plugins.OneSignal.init(ID_PROYECTO_ONESIGNAL,
            {googleProjectNumber: ID_GOOGLE_CLOUD_MESSAGING},
            notificationOpenedCallback);
```

- Una vez inicializado, OneSignal asigna un identificador único para el dispositivo. Para obtenerlo y guardarlo lo hacemos de la siguiente forma:

```
window.plugins.OneSignal.getIds(function (ids) {
        window.localStorage.setItem('opensignalid', ids.userId);
    });
```

- La id de OneSignal la obtenemos al abrir la app, por lo tanto, cuando hacemos login enviamos aparte del nombre de usuario y contraseña que se quiere loguear también la id de OneSignal. Así desde el código de la web guardamos en la base de datos el id de usuario de nuestra aplicación con que id de OneSignal está asignado para así poderle enviar posteriormente notificaciones.



id_onesignal	id_user	active	platform
5ec898ce-af0b-4164-870e-85b550ab5946	33	1	mobile
e66227bd-4a93-4c40-a14d-ec754a1311b3	26	1	mobile
NULL	NULL	NULL	NULL

Ilustración 8 – Id aplicativo & id OneSignal

4.2.10.2.2. Enviar notificaciones

Para enviar notificaciones desde el servidor a cualquier dispositivo, previamente inicializado, se hace de la siguiente forma:

- Obtenemos el id de OneSignal que tenemos almacenado en la base de datos a través del id del usuario aplicativo.
- Una vez tenemos el id del usuario o de los usuarios que queremos enviar la notificación, creamos un json con la configuración de la notificación a enviar.El ejemplo más básico es:

```
$fields = array(
            'app_id' => ID_PROYECTO_ONESIGNAL,
            'include_player_ids'=> ID_USUARIO_A_ENVIAR_NOTIFICACION,
            'large_icon' => URL_IMAGEN_LOGO_CENTRO,
            'contents' => TEXTO_NOTIFICACIÓN,
            'headings' => TITULO_NOTIFICACIÓN
```

```
        );
```

- Una vez ya creado el json, lo enviamos mediante curl a OneSignal de la siguiente forma:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "https://onesignal.com/api/v1/notifications");
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json',
    'Authorization: Basic NGM0MzJiZGYtNDcwZC00MmZjLWJhMTctNmJjMjk2ZjVmNTBj'));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch, CURLOPT_HEADER, FALSE);
curl_setopt($ch, CURLOPT_POST, TRUE);
curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
$response = curl_exec($ch);
curl_close($ch);
```

De esta forma OneSignal se encarga de transformar el json a la notificación para cada tipo de dispositivo, y enviarlo al móvil correspondiente.

4.2.10.2.3. Recibir notificaciones

Si al recibir una notificación queremos abrir una pantalla u otra en función del mensaje recibido, podemos añadir al json la variable data con un array clave-valor que se puede leer al recibir la notificación de la siguiente forma:

```
var notificationOpenedCallback = function (jsonData) {
        if (!jsonData.isActive) {
            switch (jsonData.additionalData.accion) {
                case "chat":
                        window.location.replace("#mensajes");
                    break;
                case "promocion":
                        window.location.replace("#centro");
                    break;
            }
        }
    }
```

5. Resultados

Ilustración 9 – Logo MiHora

Después de los meses de desarrollo que ha durado el proyecto, el resultado ha sido la primera versión de la app MiHora. Cubriendo las necesidades y requerimientos explicados anteriormente.

5.1. Backend

Desde el backend que se accede mediante un usuario específico desde el portal, se pueden realizar las funciones de administración de la aplicación. Dar de alta a nuevos profesionales, actualizar las subscripciones, crear roles o asignar diferentes permisos a los usuarios.

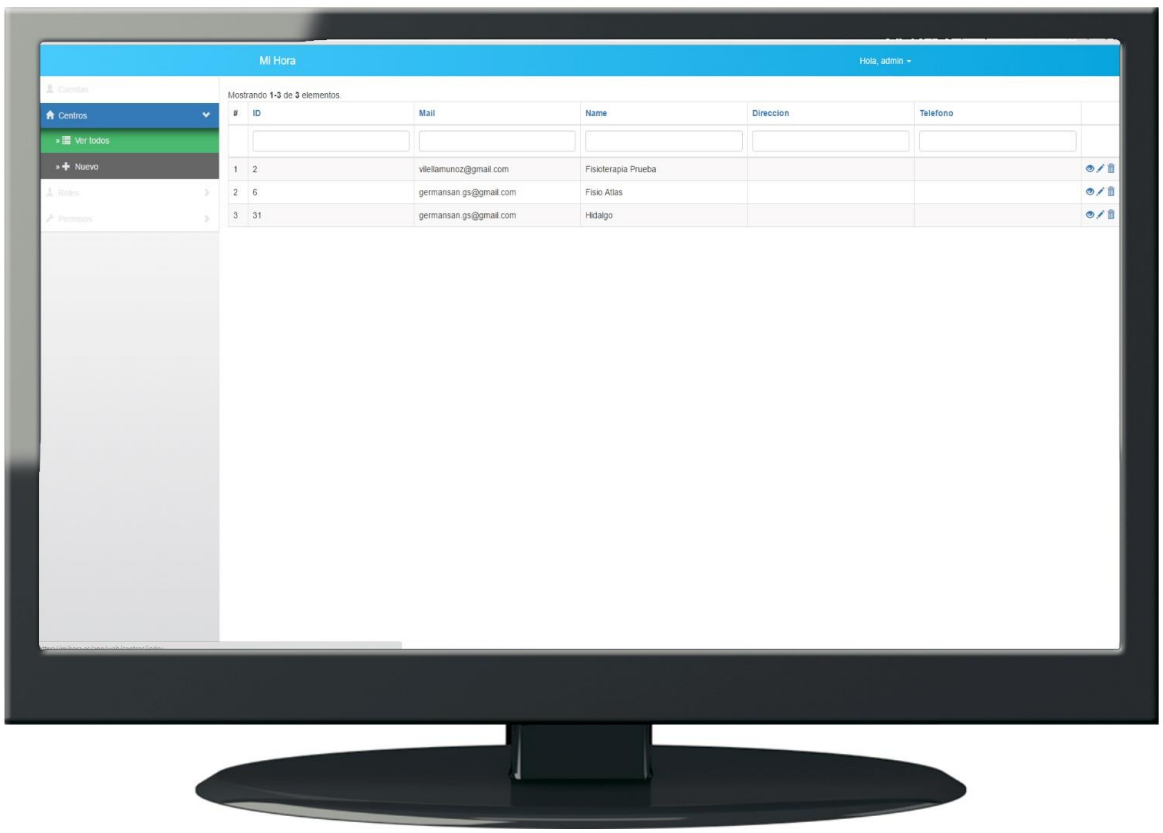


Ilustración 10 – backend (listado de centros)

5.2. Profesional

El profesional puede acceder a las funcionalidades de la app tanto desde la web como desde el móvil, anqué desde este segundo las acciones que puede llevar a cabo son más reducidas.

5.2.1. Mail de bienvenida

Una vez damos de alta a un nuevo profesional, este recibe un email de bienvenida a mihora, donde se le indican tanto su nombre de usuario y contraseña (que podrá cambiar posteriormente) y las primeras indicaciones de cómo utilizar el servicio.

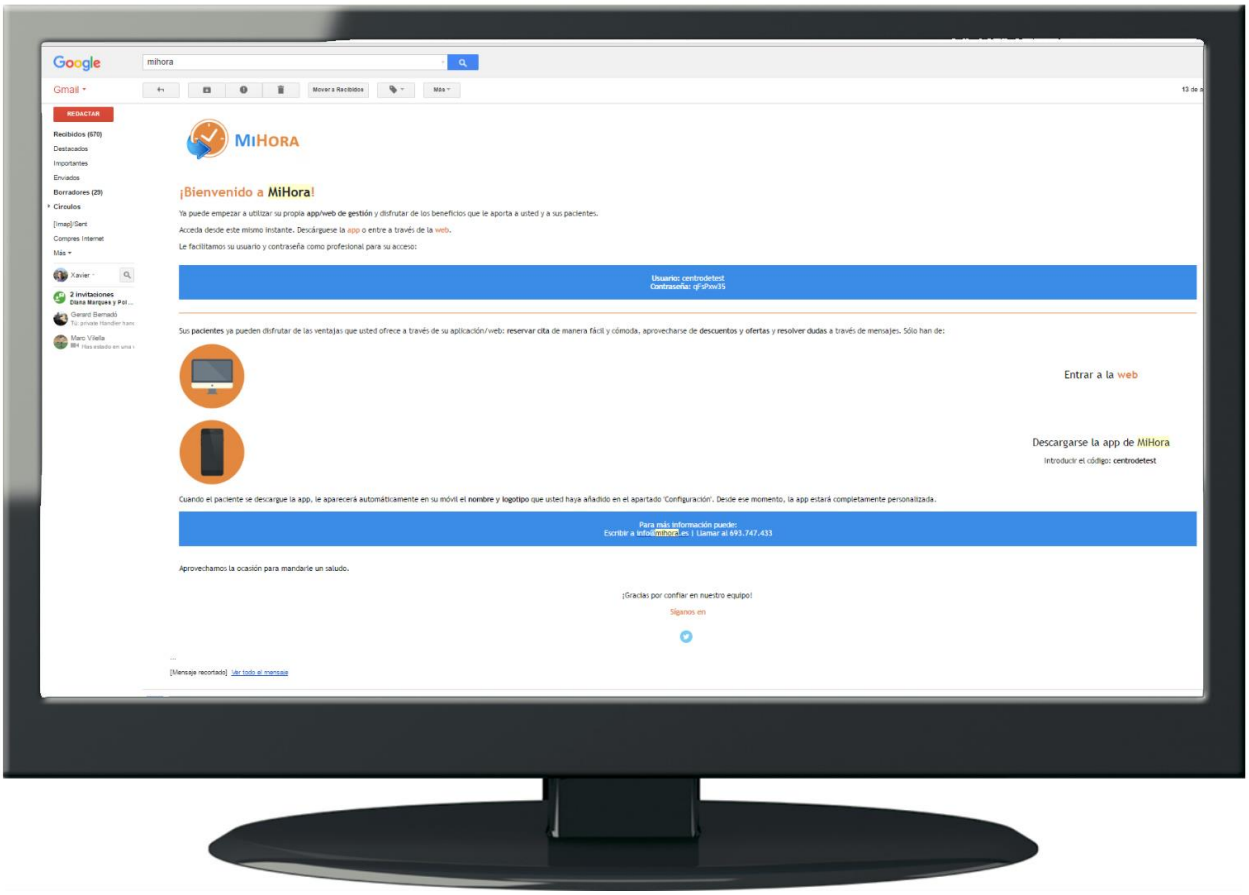


Ilustración 11 – email centro nuevo

5.2.2. Configuración de su aplicación

Una vez dentro del portal el profesional podrá configurar diversas opciones de su app.

5.2.2.1. Configuración global de la app

El profesional tiene un apartado de configuración de la app desde donde puede subir su logo e indicar los datos básicos de su empresa.

El logo que suba es el que se verá reflejado en la app móvil una vez se introduzca el código del centro, teniendo así en el móvil un acceso directo a su app con su propio logo.

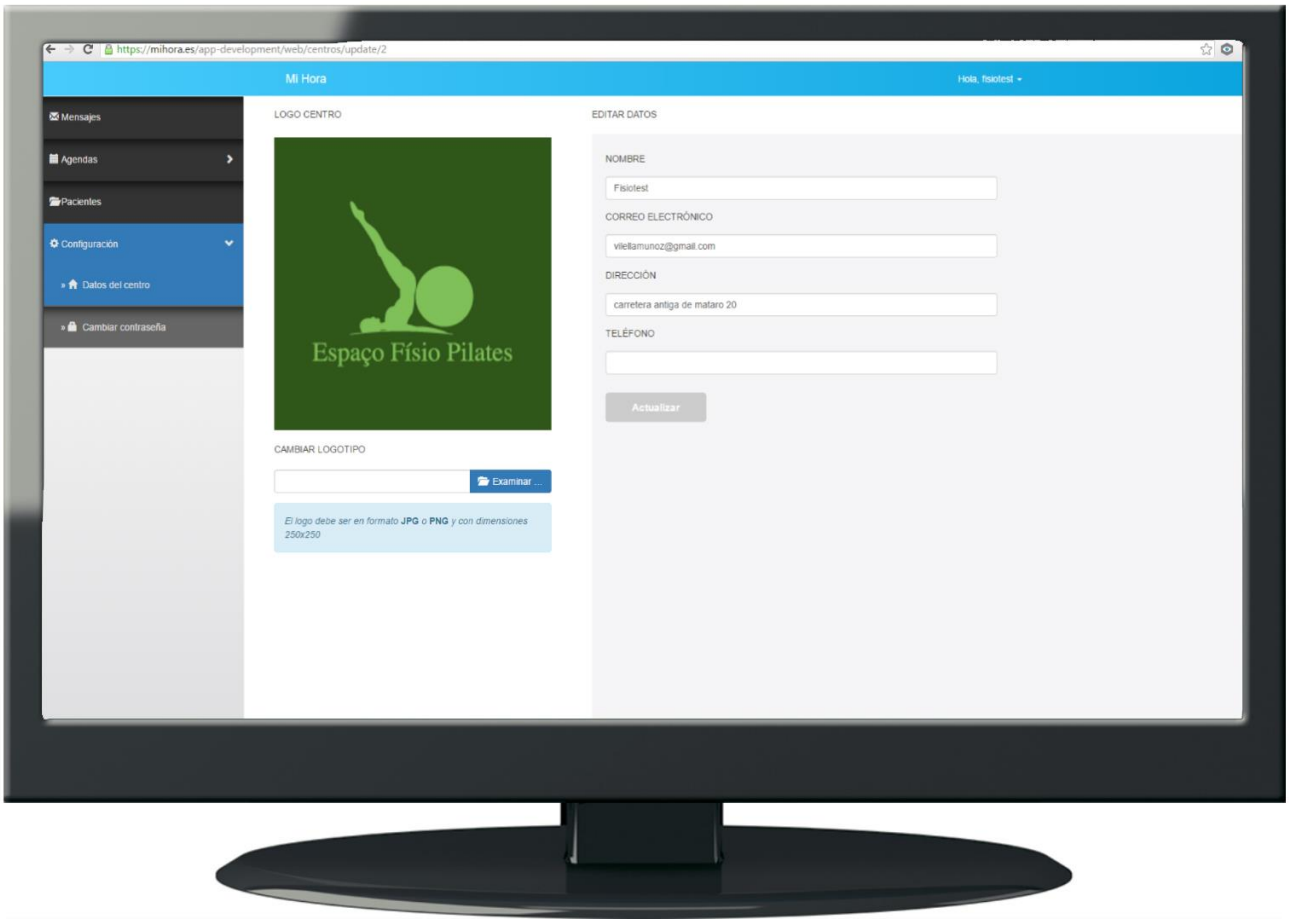


Ilustración 12 – configuración del centro

5.2.2.2. Creación agendas

Una vez dentro del portal, el profesional podrá crear tantas agendas como tenga contratadas. Si llega al máximo que tiene contratado verá un aviso de que debe actualizar su suscripción si quiere crear más agendas.

Además, con la creación de cada nueva agenda se recibe un usuario con permisos para ver solamente los datos de dicha agenda, pudiéndolas así asignar cada una a un empleado distinto y manteniendo la confidencialidad de datos de unos y otros.

5.2.2.3. Configuración de las agendas

Cada agenda se puede configurar el nombre, la duración de la sesión, si se obliga al paciente a escribir o no motivo de reserva y el horario de apertura.

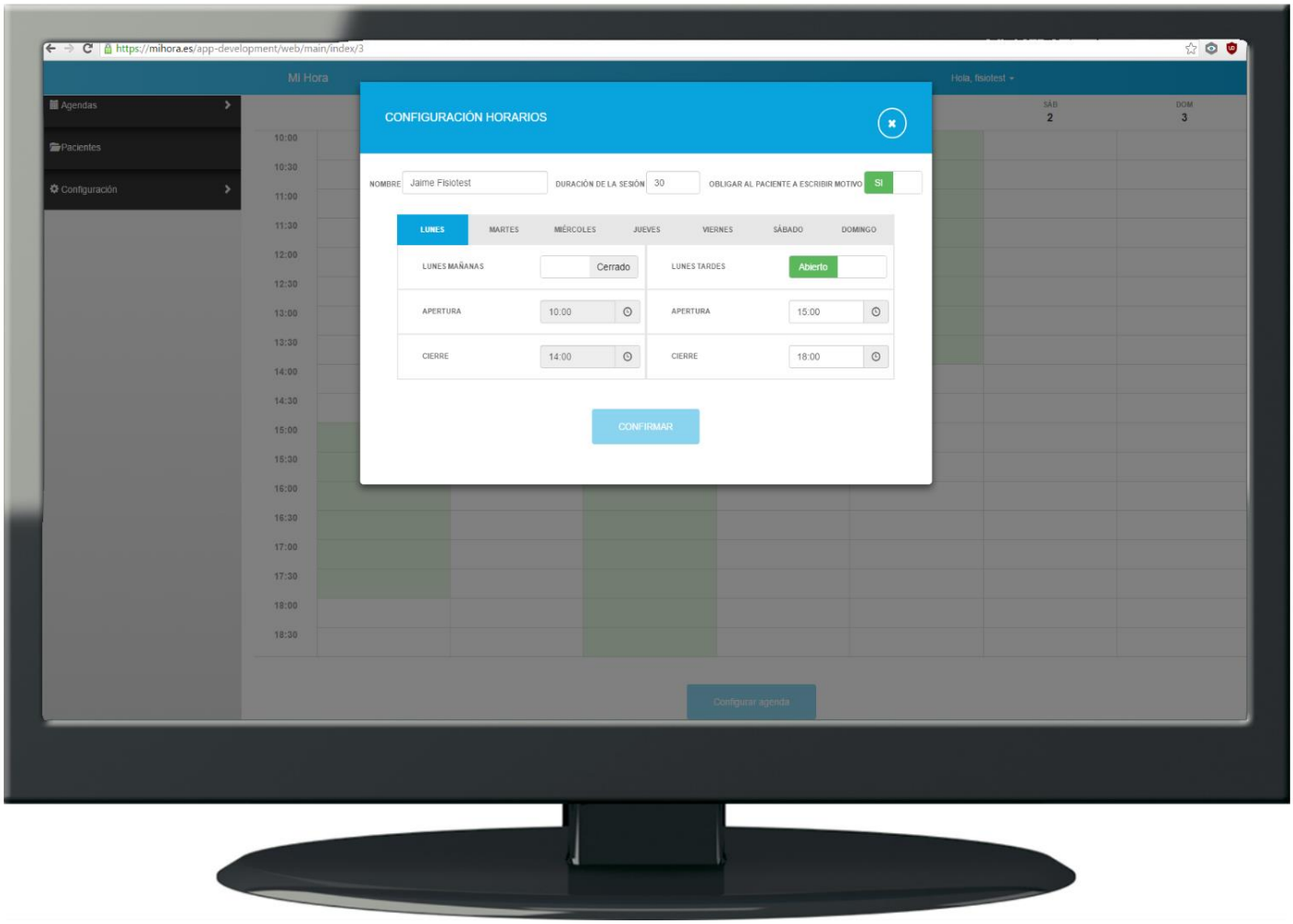


Ilustración 13 – configuración de la agenda

5.2.3. Agenda

La agenda es la función principal de la aplicación. Desde ella el profesional puede ver qué las horas que tiene reservadas y de que paciente son, el estado de dicha reserva pudiéndolas cambiar de pendiente a confirmada o cancelarlas. También puede crear las reservas manuales, cerrar horas para que no puedan ser reservadas y crear promociones, todo ello desde la misma pantalla.

Todas estas funciones están disponibles tanto desde el portal web como la app para los profesionales.

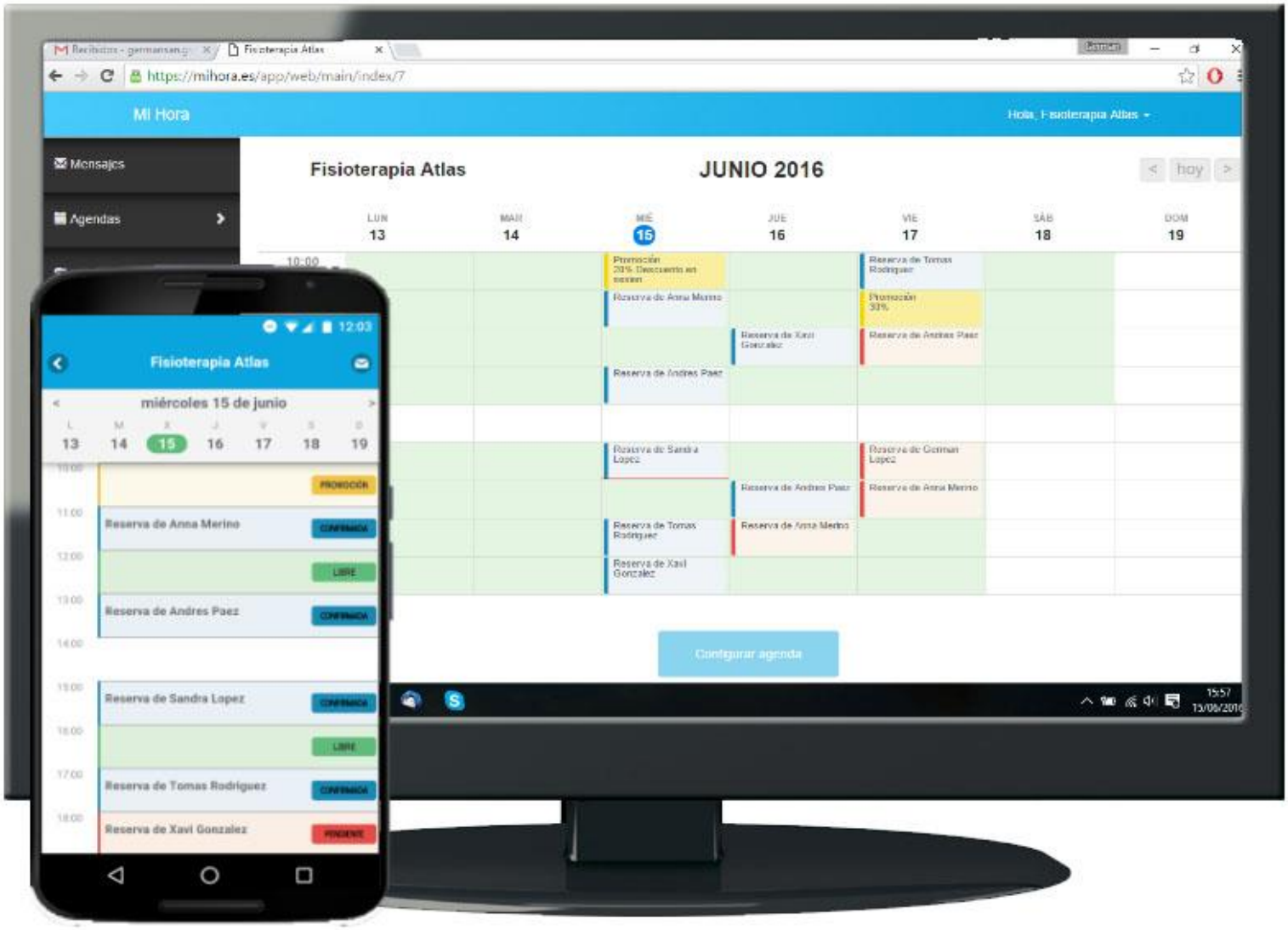


Ilustración 14 – Vista agenda como profesional

5.2.4. Pacientes

En esta primera versión de MiHora, para que un paciente pueda acceder a la aplicación de su profesional, este deberá haberle creado previamente su ficha. Cuando lo haga se le enviará un email automáticamente indicándole las credenciales para acceder y como utilizar la app.

Para ello el profesional tiene un apartado Pacientes, desde donde podrá crearles las fichas. También ve un listado de todos sus pacientes pudiendo acceder a la ficha de cualquiera de ellos para añadir notas clínicas o actualizar la información de su ficha.

De estas funcionalidades, desde el móvil están disponibles el listado de pacientes y ver la ficha y notas de los pacientes, pero no la posibilidad de crear nuevos pacientes o insertar notas.

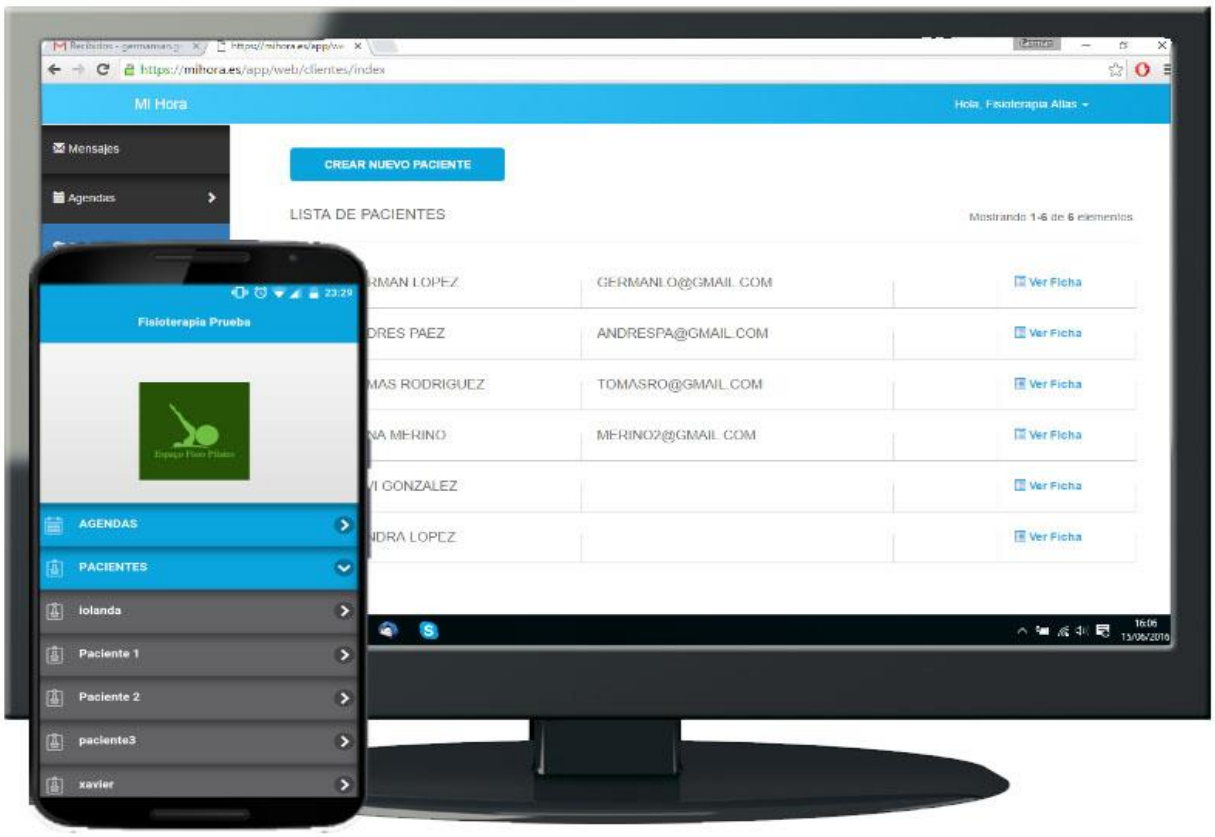


Ilustración 15 – listado de pacientes

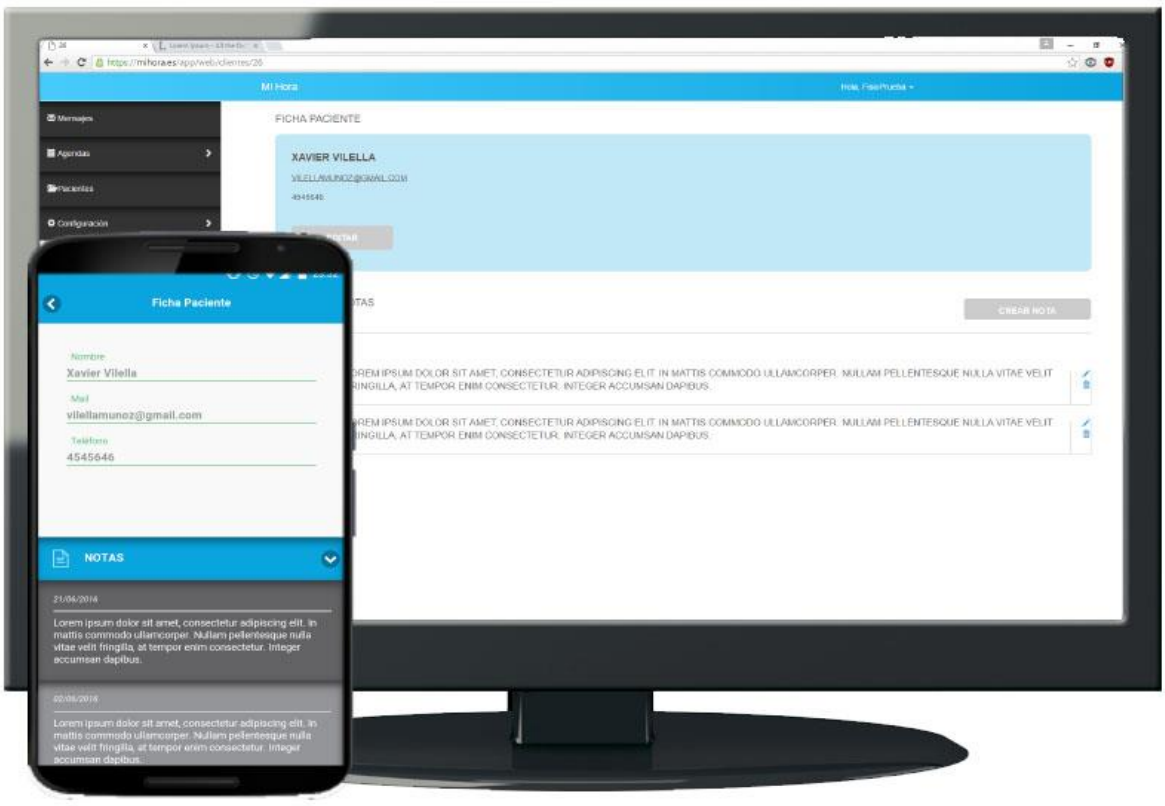


Ilustración 16 – ficha paciente

5.2.5. Mensajes

En esta pantalla tipo chat, se pueden mantener conversaciones cada agenda o trabajador con sus pacientes.

Además, hay ciertos mensajes que se envían de forma automática, por ejemplo, cuando cambia el estado de una reserva se avisa a dicho paciente con un mensaje, o cuando un profesional crea una promoción se avisa a todos los pacientes del centro (que no tengan ninguna reserva activa) mediante otro mensaje.

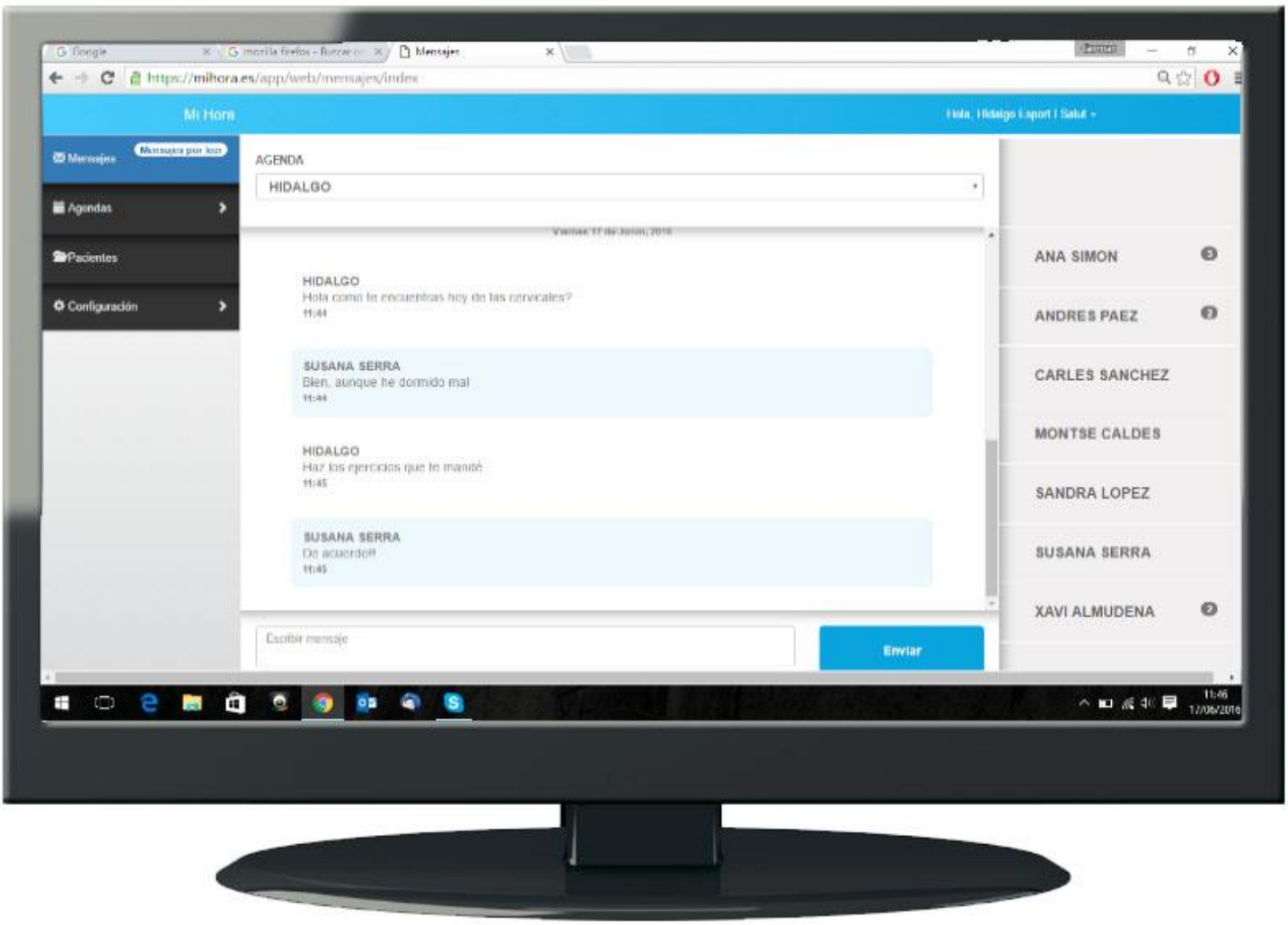


Ilustración 17 - mensajes

5.3. Paciente

Los pacientes en esta primera versión de MiHora solamente tienen acceso desde la app móvil, y obteniendo las credenciales una vez el profesional les ha creado la ficha.

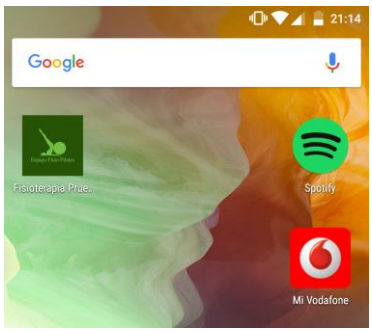


Ilustración 18 – logo profesional

5.3.1. Mail de bienvenida

Una vez el profesional les ha creado la ficha, reciben un email de bienvenida con la información de su centro, donde se le indican tanto su nombre de usuario y contraseña, y las indicaciones de cómo conseguir la app de su profesional mediante MiHora.

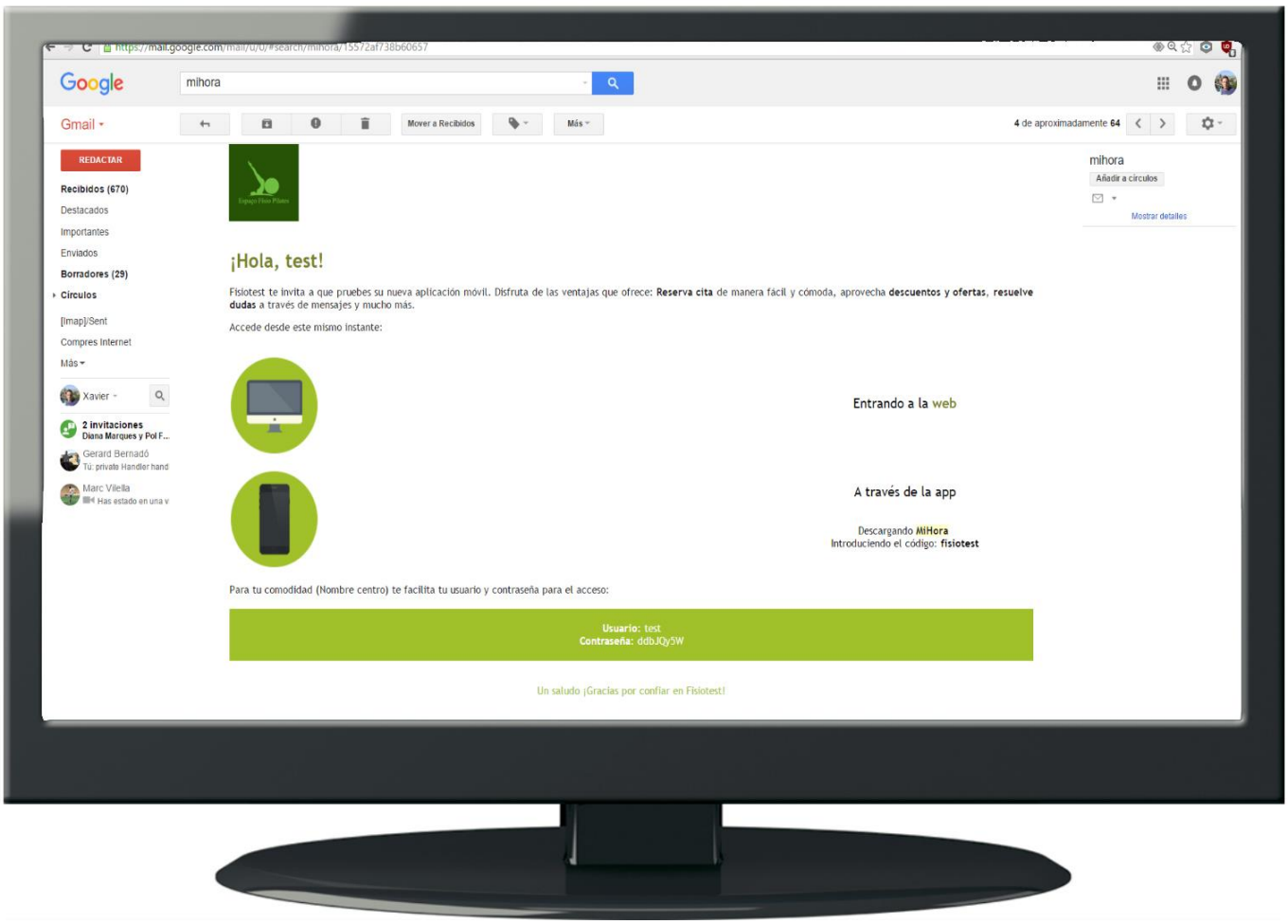


Ilustración 19 – email Nuevo paciente

5.3.2. Portada del centro

En caso de que el centro tenga más de una agenda, será la primera pantalla que verá el paciente para poder seleccionar la agenda de que profesional quiere visualizar. En caso de que solamente tenga una agenda se cargará la agenda directamente.

5.3.3. Agenda

Dentro de la agenda de un profesional, el paciente verá qué horas están disponibles para reservar y qué horas están ocupadas, pero sin ver de quien es la reserva. En caso de que hubiera promociones, si el paciente no tiene ninguna reserva activa también las vería sobre la agenda. En caso de tener una reserva además tiene disponible la opción MiHora, que le lleva hasta el día y hora que tiene la reserva remarcándole en azul.

Desde la agenda, el paciente puede crear reservas que, por defecto, se crean en estado pendiente de confirmar.

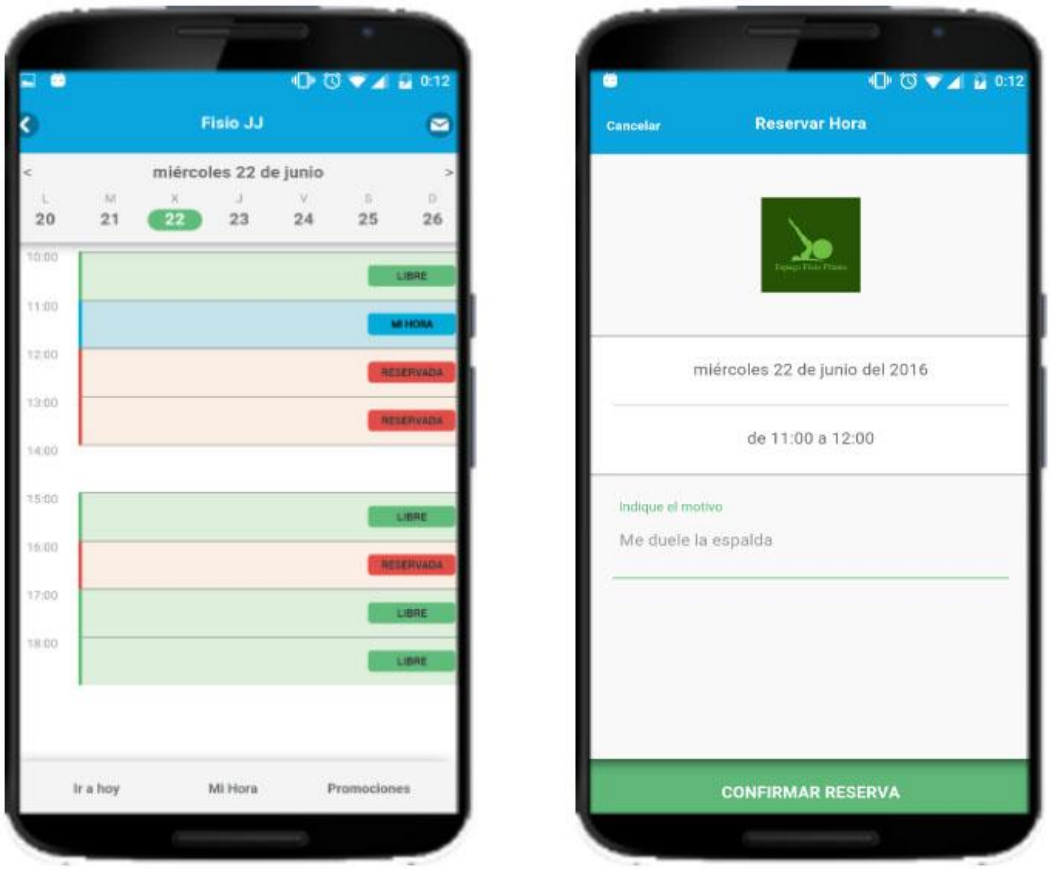


Ilustración 20 – Agenda vista paciente

5.3.4. Mensajes

De la misma forma que los profesionales tienen un chat para hablar con todos sus pacientes, estos tienen una conversación directa con su profesional desde donde pueden resolver dudas o recibir los mensajes de promociones y actualizaciones del estado de sus reservas.

Cuando se recibe un mensaje, este se visualiza en el móvil en forma de notificación directamente con el logo del centro, sin mostrar nada de MiHora, haciendo así la experiencia tanto del profesional como del paciente, más satisfactoria.

5.4. Recordatorios

Mediante un CRON que se ejecuta cada día a las 6 de la tarde, se miran todos los pacientes que tienen visita el día siguiente y se les envía un recordatorio con su visita.

De esta manera se disminuye el ausentismo a las citas concertadas.

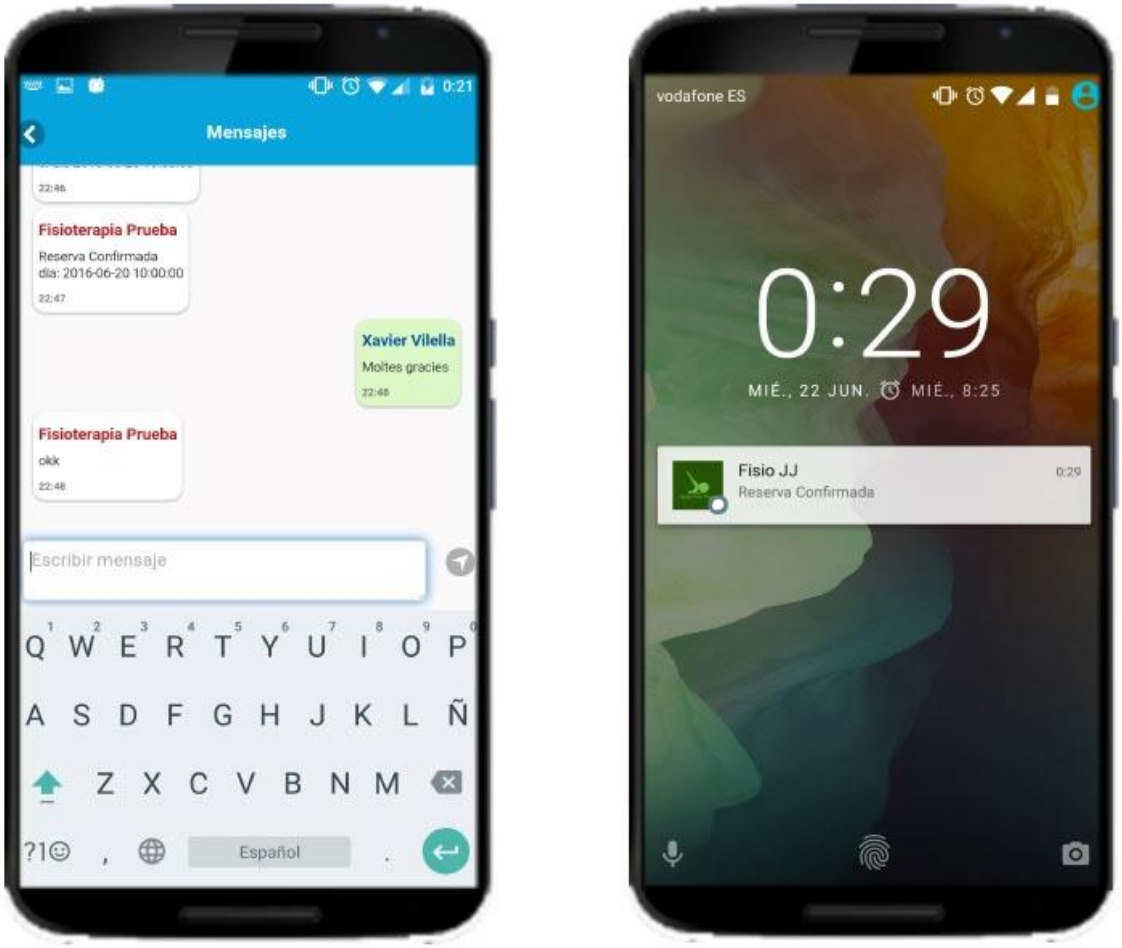


Ilustración 21 – mensajes y notificación paciente

6. Presupuesto

Licencia Android Developer: 25€
Hosting plan junior: 5€ al mes
Dominio web: 20€ al año
Certificado SSL: 85€ al año

670 horas trabajadas aproximadamente, distribuidas de la siguiente forma:

- Definición del proyecto: 40h
- Desarrollo portal web:350h
- Desarrollo app móvil:120h
- Diseño web y móvil:50h
- Corrección de errores:50h
- Preparación entornos desarrollo y producción: 20h
- Securización de la información: 40h

Lo que conlleva un coste suponiendo un sueldo de 12€ la hora de 8.040€

En un futuro se debería comprar la licencia de iOS: 75€ al año
Y ampliar el plan de hosting al Plan master: 21€ al mes.

Los ingresos vienen dados por la venta de licencias a profesionales, partiendo de que todas las que vendamos son el peor caso.
Venta de licencia a 1 autónomo con solamente 1 agenda: 20€ al mes.

El resumen del plan de ventas a 3 años quedaría de tal forma:

Con 30 licencias vendidas la aplicación ya empezaría a dar beneficios, pero no suficientes como para poder tener un salario.

A finales del año con 100 licencias vendidas se podría empezar a tener un salario de aproximadamente 1.500€ al mes. Acabando el año con unos beneficios acumulados de aproximadamente 3.500€.

A mitades del segundo año se haría una inversión en un comercial para potenciar la venta de licencias y en el tercer trimestre un segundo comercial para seguir aumentándolas. De esta forma acabaríamos el segundo año con un beneficio acumulado de unos 53.000€.

El tercer año debería ser el año de la expansión definitiva invirtiendo en más personal tanto comercial como de atención al cliente y técnicos. En el que la previsión seria de vender 4000 licencias de una aplicación ya madura después de dos años en el mercado pudiendo llegar a acumular aproximadamente 540.000€ de beneficios.

		gasto en salarios	gastos mantenimiento	licencias vendidas	ingresos mensuales	beneficios acumulados
2016	enero	0 €	0 €	0	0 €	0 €
	febrero	0 €	-25 €	0	0 €	-25 €
	marzo	0 €	-30 €	0	0 €	-55 €
	abril	0 €	-5 €	0	0 €	-60 €
	mayo	0 €	-5 €	0	0 €	-65 €
	junio	0 €	-90 €	0	0 €	-155 €
	julio	0 €	-80 €	0	0 €	-235 €
	agosto	0 €	-5 €	10	200 €	-40 €
	septiembre	0 €	-5 €	30	600 €	555 €
	octubre	0 €	-5 €	50	1.000 €	1.550 €
	noviembre	0 €	-21 €	70	1.400 €	2.929 €
	diciembre	-1.500 €	-21 €	100	2.000 €	3.408 €
2017	enero	-1.500 €	-201 €	130	2.600 €	4.307 €
	febrero	-1.500 €	-21 €	160	3.200 €	5.986 €
	marzo	-1.500 €	-21 €	200	4.000 €	8.465 €
	abril	-1.500 €	-21 €	250	5.000 €	11.944 €
	mayo	-3.000 €	-21 €	300	6.000 €	14.923 €
	junio	-3.000 €	-21 €	350	7.000 €	18.902 €
	julio	-3.000 €	-21 €	400	8.000 €	23.881 €
	agosto	-3.000 €	-21 €	450	9.000 €	29.860 €
	septiembre	-6.000 €	-21 €	500	10.000 €	33.839 €
	octubre	-6.000 €	-21 €	550	11.000 €	38.818 €
	noviembre	-6.000 €	-21 €	600	12.000 €	44.797 €
	diciembre	-6.000 €	-21 €	700	14.000 €	52.776 €
2018	enero	-12.000 €	-201 €	1000	20.000 €	60.575 €
	febrero	-12.000 €	-21 €	1300	26.000 €	74.554 €
	marzo	-12.000 €	-21 €	1600	32.000 €	94.533 €
	abril	-12.000 €	-21 €	1900	38.000 €	120.512 €
	mayo	-12.000 €	-21 €	2200	44.000 €	152.491 €
	junio	-12.000 €	-21 €	2500	50.000 €	190.470 €
	julio	-12.000 €	-21 €	2800	56.000 €	234.449 €
	agosto	-20.000 €	-21 €	3100	62.000 €	276.428 €
	septiembre	-20.000 €	-21 €	3500	70.000 €	326.407 €
	octubre	-20.000 €	-21 €	4000	80.000 €	386.386 €
	noviembre	-20.000 €	-21 €	4500	90.000 €	456.365 €
	diciembre	-20.000 €	-21 €	5000	100.000 €	536.344 €

Tabla 12 – Estimación beneficios

7. Conclusiones y desarrollo futuro:

Después de realizar este proyecto he comprobado la realidad de que tener un buen producto es solo un 30% para que funcione. Después de todos estos meses desarrollando la idea consiguiendo un prototipo bastante avanzado del producto final, ahora queda por delante el reto más atractivo, ponerlo en el mercado y posicionarse entre otras aplicaciones, que pese no ofrecer lo mismo tienen ciertas similitudes.

El hecho de hacer que una aplicación sea sencilla e intuitiva tiene más posibilidades de funcionar que una que ofrezca más opciones, pero sea muy complicada de utilizar. Por ello para desarrollar el producto es muy importante tener el feedback de quien lo va a usar. Ya que muchas veces lo más difícil es hacerlo sencillo.

En cualquier proyecto es muy importante parar un tiempo antes de empezar a “picar código” y estructurarte bien lo que quieres hacer, ya que este tiempo empleado lo ahorras después al no tener que rehacer cosas que no habías pensado en un primer momento.

De la misma forma que para desarrollar partes del proyecto es importante mirar plugins o código de terceras personas, de código libre o no, ya que, aunque finalmente no se use seguramente se podrán extraer conclusiones positivas que mejoraran la calidad del producto.

Analizar los puntos fuertes y débiles, ya que muchas veces es mejor tener uno o dos puntos muy fuertes que tener muchos entremedios,

Las herramientas como los Frameworks tanto para web como para móvil pueden ser muy potentes y ayudan al desarrollo, pero se tiene que ir en cuenta porque por la misma facilidad de crear código con frameworks es posible que se acabe creando un código que funcione, pero sea muy difícil mantenerlo.

Es muy importante tener como mínimo dos entornos uno para desarrollar y otro estable, ya que cualquier cambio puede influir en algo que no te esperabas y siempre puede ser necesario volver atrás a la última versión estable.

Finalmente, como desarrollo futuro está compilar la versión de la app para iOS, que dadas las circunstancias de necesitar un iMac y el aprendizaje de algunas funciones propias, no me ha dado tiempo.

Una vez este publicada la versión para iOS ya se trata de entrar en el mercado, buscar probadores, todavía sin cobrarles licencia, para obtener una segunda ronda de feedback para seguir puliendo detalles y poder en septiembre sacar ya una primera versión estable sobre la cual buscar compradores con licencia.

Bibliografía:

[1] “Cloud Messaging | Google Developers", Google Developers, 2016. [Online]. Available: https://developers.google.com/cloud-messaging/.

[2] “Documentation - Apache Cordova", Cordova.apache.org, 2016. [Online]. Available: https://cordova.apache.org/docs/en/latest/.

[3] “Documentation | FullCalendar", Fullcalendar.io, 2016. [Online]. Available: http://fullcalendar.io/docs/.

[4] “Getting Started with OneSignal · OneSignal Push Notification Service Documentation", OneSignal Push Notification Service Documentation, 2016. [Online]. Available: https://documentation.onesignal.com/.

[5] "jQuery API Documentation", Api.jquery.com, 2016. [Online]. Available: http://api.jquery.com/.

[6] "Bootstrap · The world's most popular mobile-first and responsive front-end framework.", Getbootstrap.com, 2016. [Online]. Available: http://getbootstrap.com/.

[7] "Moment.js | Docs", Momentjs.com, 2016. [Online]. Available: http://momentjs.com/docs/.

[8] "MySQL Documentation", Dev.mysql.com, 2016. [Online]. Available: http://dev.mysql.com/doc/.

[9] "PhoneGap Documentation | PhoneGap Docs", Docs.phonegap.com, 2016. [Online]. Available: http://docs.phonegap.com/.

[10] "PHP: Manual de PHP - Manual", Php.net, 2016. [Online]. Available: http://php.net/manual/es/index.php.

[11] "Stack Overflow", Stackoverflow.com, 2016. [Online]. Available: http://stackoverflow.com/.

[12] "The Definitive Guide to Yii 2.0", Yiiframework.com, 2016. [Online]. Available: http://www.yiiframework.com/doc-2.0/guide-index.html.