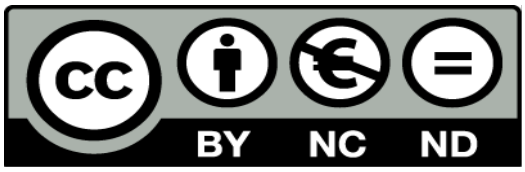
UNIVERSIDAD
DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Desarrollo e implementación de una solución de telemedicina empleando dispositivos Android con conectividad bluetooth y la nube de dispositivos de Digi- aplicación web
Autor/es
Diego Ibáñez Pancorbo
Director/es
Angel Luis Rubio García
Facultad
Facultad de Ciencias, Estudios Agroalimentarios e Informática
Titulación
Grado en Ingeniería Informática
Departamento
Curso Académico
2012-2013



Resumen

La siguiente memoria se encarga de recoger los aspectos más importantes del desarrollo de una aplicación web encargada de facilitar a los pacientes de un centro de salud un seguimiento personal por parte de sus médicos. Este seguimiento será posible gracias a la comunicación del portal web y la nube de Etherios, posibilitando el almacenamiento de los datos de las muestras tomadas y la comunicación con la aplicación Android.

El proyecto surge como una continuación de las practicas realizadas en la empresa Digi International SAU de una serie de proyectos enfocados a estudiar las posibilidades que ofrece la nube de Etherios para el desarrollo de aplicaciones comerciales, y ha sido desarrollado completamente por Diego Ibáñez Pancorbo bajo la tutoría del Dr. Ángel Luis Rubio García.

Summary

The following memory deals with collecting the most important developing issues a web application responsible providing patients, of a health center, a personal monitoring by their doctors. This monitoring will be possible through communication with the web portal and Etherios cloud, enabling the storage of data of samples collected and communication with the Android application.

The project emerged as a continuation of the practices carried out in the company Digi International SAU of a series of projects focused on studying the potential of Etherios cloud for commercial application development, and has been entirely developed by Diego Ibáñez Pancorbo under the tutorship of Dr. Angel Luis Rubio García.

Índice

Índice de figuras

Capítulo 1

Introducción

En este capítulo se describirá el contexto en el cual se ha desarrollado el proyecto así como los objetivos que se buscaban cumplir en la realización del mismo.

1.1. Introducción

El TFG parte tomando como base el trabajo que se ha estado realizando durante los cuatro primeros meses de prácticas para la empresa Digi International SAU. La aplicación completa se compone de dos partes, un portal web y una aplicación nativa para terminales con sistema operativo Android. Tanto como en las prácticas en la empresa como en este TFG la parte que corresponde al proyectante, Diego Ibáñez, es el desarrollo del portal web.

El portal web a su vez contará con dos partes, la primera y más extensa será el proseguir y terminar la parte orientada hacia los usuarios (pacientes) de la aplicación y es de la parte de la que ya se dispone material desarrollado en prácticas. Por su parte la segunda parte será de carácter más sencillo y tendrá como objetivo el proporcionar al otro tipo de usuarios (médicos) la posibilidad de hacer un seguimiento a los pacientes.

De la primera parte del portal web como ya se ha dicho previamente se cuenta con material que se ha desarrollado en la parte de prácticas en empresa, más detalladamente este material es:

- Un primer prototipo del aspecto que va a tener el portal web que ha permitido el testear el funcionamiento del trabajo desarrollado.

- Algunos servlets que se encargan de las partes más básicas como puede ser el login, el logout, el registro y el acceso a las distintas pantallas que van a componer la aplicación web.

- Una versión auxiliar de la base de datos con los métodos de la capa de persistencia necesarios para las funciones que se han detallado en el punto anterior.

- Estudio de los requisitos y distintas herramientas que vamos a utilizar para gestionar las dos nubes que vamos a utilizar en el proyecto.

Cabe destacar los dos servicios en los que vamos a guardar o la aplicación o los datos recogidos por esta:

1.1.1. Google App Engine

Google App Engine es la infraestructura que permite subir aplicaciones para que otros usuarios puedan usarlas sin necesidad de montar ningún servidor. Gracias a este servicio podremos mantener la aplicación en la web sin ningún coste aunque esto nos presente algunas restricciones a la hora de tomar ciertas decisiones de carácter técnico. Pese a esto estas restricciones son menos importantes que las ventajas que nos ofrece y por ello ha sido elegido como servidor de la aplicación.

1.1.2. Etherios

Es la nube de dispositivos de Digi. La gran diferencia y ventaja que posee respecto a otras, y el motivo por el que se ha elegido utilizar este servicio, es que no gestiona carpetas como lo hacen otras nubes, aunque puede, sino que gestiona dispositivos asociados a la cuenta y esto da una ventaja para la comunicación bidireccional.

1.2. Objetivos

1.2.1. Objetivos globales del proyecto

- Desarrollo de un sistema de comunicación bidireccional entre un dispositivo Android y el portal web del sistema que muestre y ayude a distintos tipos de pacientes y a sus médicos a llevar un seguimiento de la salud de los primeros.

- Realización de un vídeo demostrativo del conjunto de la aplicación funcionando correctamente.

- Redacción de un artículo de presentación del proyecto para la empresa con la que se colabora en la realización de este.

1.2.2. Objetivos particulares del proyecto

- Hacer las fases de análisis, diseño, implementación y pruebas de las dos partes en las que se va a dividir el TFG, teniendo en cuenta en todas ellas todo el material aprovechable del periodo de prácticas en empresa.

- Reestructuración de todo el material desarrollado en las prácticas en empresa para conseguir una base sólida y funcional desde la cual avanzar en la creación del portal web.

- Desarrollo de una nueva interfaz la cual sea elegida entre el proyectante y la empresa con la que se colabora en el TFG.

- Añadir la posibilidad de introducir distintas pruebas, de forma manual, a través del portal web.

- Realización de las pantallas de seguimiento mediante los datos obtenidos con las distintas pruebas y la posibilidad de mostrarlos tanto gráficamente como por listado.

- Implementación de un calendario que permita la gestión de las distintas actividades que tendrán los pacientes que utilicen la aplicación y un sistema de notificaciones a través de la aplicación Android para el recordatorio de estas actividades.

- Realización de las pantallas y las modificaciones en la base de datos necesarias para que una serie de médicos con pacientes asociados puedan mantener un seguimiento y mandar actividades con sus respectivas notificaciones en caso de que el paciente lo requiera.

- Corrección de varios bugs que se han detectado en el trabajo realizado en la parte de prácticas en empresa y de los que pudiesen surgir en la fase de pruebas del proyecto.

Capítulo 2

Planificación

En este capítulo se detallarán la asignación de tareas para cada una de las semanas en las que se realizará el proyecto.

2.1. Descripción de la planificación

La planificación se desarrollará siguiendo una metodología que podría asemejarse a la metodología ágil Scrum, ya que se desarrollará trabajo durante una semana y al final de esa semana se le presentarán los avances al cliente, pero que no va a ser considerada como tal debido a que el objetivo semanal no es crear una parte definitiva si no que ir avanzando en el desarrollo de la aplicación web, aunque puede que ciertas semanas se centre exclusivamente en finalizar una parte de la aplicación y enseñar su funcionamiento definitivo. Por lo tanto la siguiente tabla refleja las 14 semanas de desarrollo que va a tener el proyecto y las 2 semanas de memoria, empleando 19 horas de trabajo cada semana para alcanzar un total de 304 horas dedicadas al proyecto.

2.2. Tabla de la planificación

Semana	Dedicado a
Semana 1	Recopilación del trabajo desarrollado en prácticas.
Semana 2	Desarrollo de los formularios de acceso y de creación de cuentas y usuarios.
Semana 3	Implementación del menú de usuarios y del menú principal.
Semana 4	Implementación de la pantalla de selección del dispositivo.
Semana 5	Desarrollo de las funcionalidades para subir nuevas cuentas y crear médicos desde el portal del administrador.
Semana 6	Implementación de los métodos de acceso a la nube.
Semana 7	
Semana 8	Implementación de la visualización de valores en las pruebas.
Semana 9	Implementación del formulario para introducir valores de pruebas.
Semana 10	Desarrollo de la página encargada de mostrar el calendario.
Semana 11	Implementación del sistema de eventos.
Semana 12	Desarrollo del portal del médico.
Semana 13	Desarrollo del formulario de asignación de pacientes a médicos
Semana 14	Implementación de servlets para cambiar de usuario o salir de la cuenta.
Semana 15	Introducción, planificación, análisis y diseño de la memoria.
Semana 16	Implementación, pruebas y conclusiones de la memoria.

Capítulo 3

Análisis

En este capítulo se detallarán los requisitos tanto funcionales como los propios de la interfaz y se adjuntarán los diagramas de casos de uso acompañados por los casos de uso más representativos.

3.1. Requisitos del sistema

3.1.1. Requisitos funcionales

En esta sección se van a detallar los requisitos funcionales que deben cumplir cada una de las partes en la que está dividida la aplicación. Debido a que no se trata de una aplicación orientada a un usuario final específico puede que algún requisito funcional quede a elección de la empresa que contrate este servicio.

Requisitos funcionales del portal para los usuarios

- Los usuarios podrá crear, siempre que haya cuentas de Etherios disponibles en la base de datos, su propia cuenta con un usuario asociado. Una vez creada la cuenta se podrán ir agregando todos los usuarios que se deseen. Toda cuenta tendrá asociada a su vez una cuenta de Etherios, que habrá sido introducida en la base de datos por un administrador, y todos los usuarios de una misma cuenta compartirán esa cuenta de Etherios, sin la posibilidad de poder ver los datos de otros usuarios. Para el registro de los distintos usuarios se deberá introducir un correo que no esté ya en la base de datos.

- En el formulario que se encarga de iniciar la sesión el usuario dispondrá de dos opciones. Una que consistirá en introducir el email con el que se ha registrado y su contraseña de usuario que le dará acceso directo al menú principal de su cuenta . La otra opción de registro recogerá el nombre de la cuenta y la contraseña y llevará al usuario a una pantalla intermedia donde le mostrará los distintos usuarios que están asociados a la cuenta. En esta pantalla el usuario seleccionará su usuario y accederá al menú principal introduciendo su contraseña de usuario.

- El menú principal tendrá una serie de elementos dinámicos que son los referentes a las distintas pruebas que tiene disponibles el usuario. Estos elementos solo se mostrarán si se ha añadido alguna prueba tanto desde la aplicación Android como desde la opción para añadir pruebas que proporciona la aplicación web al usuario. Una vez que se haya añadido una prueba el sistema no permitirá la opción de volver a añadirla.

- Será posible cambiar los datos de cada usuario así como su contraseña aunque por motivos de consistencia el usuario nunca podrá cambiar el correo electrónico con el que se ha registrado ni la cuenta a la que pertenece dicho usuario. En caso de que un usuario desee cambiar alguno de estos dos campos debería contactar con el administrador del sistema para consultar si le proporciona esta posibilidad.

- Cada usuario será capaz de seleccionar el dispositivo al que desea que le lleguen las alarmas de una lista de dispositivos que previamente han accedido con ese usuario a través de la aplicación Android. En caso de que no se haya accedido nunca a través de la aplicación Android no saldrá ningún dispositivo seleccionable y no se le dará la posibilidad al usuario de establecer alarmas para los eventos.
- El sistema recogerá una serie de eventos introducidos por el usuario o por el doctor y las señalará en el calendario del paciente asignándoles un color según el tipo de tarea del que se trate. Si lo desea el usuario podrá visualizar en detalle los eventos propios de cada día. Los eventos se podrán añadir de forma manual desde la página del calendario seleccionando el día al que quieres añadir el evento o desde la propia página del día. También, al crear los eventos, dispondremos de la opción de hacerlos periódicos para se muestren con una repetición que deseemos en el intervalo de tiempo seleccionado.

- Cada evento dará la opción de activar una alarma que se establecerá en el dispositivo móvil para establecer la alarma el sistema contactará con el dispositivo que tenga asociado. Si no se dispone de ningún dispositivo o no se ha asociado ninguno no debería mostrarse la opción de establecer alarma.

- Sera posible la visualización de los distintos valores de las distintas pruebas que cada usuario tenga disponible independientemente si han sido recogidos por la aplicación Android como por el portal web. La visualización podrá realizarse de dos formas, mediante un gráfico lineal ordenado por la fecha y la hora o por una lista, ordenada de la misma que el gráfico, que nos mostrara para cada valor

recogido el propio valor, la fecha y hora a la que ha sido tomado el valor. También, para los valores que superen el rango establecido como adecuado por el médico, tendrán un color distinto y una breve descripción sobre el motivo de la alerta.

Requisitos funcionales del portal para los médicos

- Utilizarán la misma pantalla de acceso que los usuarios pero solo tendrán la opción de acceder a su cuenta a través de su email y la contraseña que tengan asociada. Los médicos podrán editar sus datos una vez que hayan accedido a su cuenta y cambiar su contraseña pero nunca un médico podrá registrarse como tal, la capacidad de dar de alta a un médico solo la tendrá un administrador.

- Podrán acceder a un listado de todos los pacientes que tienen asociados y filtrar los resultados por IDs de pacientes que contengan la cadena de texto que hayan introducido en el text box. En cada paciente el médico tendrá la opción de acceder a dos secciones. La primera se encargará de la gestión de las estadísticas del paciente seleccionado. Esta sección tendrá un gráfico con todos los valores de las distintas pruebas que ese usuario tiene activas y una pantalla de introducción de los valores límite que el paciente debe tener en cada prueba. En la segunda sección el médico tendrá un calendario como el que visualiza el paciente, gestionando de la misma forma los eventos.

Requisitos funcionales del portal para los administradores

- El administrador tendrá su propia pantalla de acceso y se le proporcionará una cuenta activa que habrá sido creada al desplegar la aplicación.

- Será necesario que el administrador introduzca en la base de datos de la aplicación nuevas cuentas de Etherios que serán asociadas a cada cuenta que el usuario se cree. En la pantalla de selección del fichero que contiene las cuentas que se desean subir también se podrá ver cuántas cuentas están disponibles en cada momento.

- El administrador se encargará de la creación de cada médico cuando el centro lo solicite, también será el encargado de asociar a cada paciente su médico. Para

esto el administrador tendrá una pantalla con todos los médicos y los usuarios registrados en el sistema. En esta pantalla para facilitar la labor del administrador se contará con un filtro por id tanto para los médicos como para los pacientes y estará permitido asociar varios pacientes a un mismo médico de una sola vez.

- En el caso de que un paciente solicite el cambio de médico no se debería tener que realizar ningún cambio previo, para esto el administrador deberá acceder a la pantalla de asignación de pacientes a médicos y volver a asignar al paciente, aunque ya posea un médico, a su nuevo médico.

3.1.2. Requisitos de la interfaz para los usuarios

Uno de los objetivos principales que se ha perseguido durante todo el desarrollo no solo ha sido dotar la aplicación de funcionalidad sino que también se ha buscado que tenga una interfaz uniforme, sencilla y usable para poder orientar la aplicación a toda clase de públicos sin que la interfaz sea una barrera para aquellos con pocos conocimientos informáticos. Este objetivo no es propio solo de la aplicación web, sino que se ha buscado la simetría entre la aplicación Android y la aplicación web para ayudar al aprendizaje y la usabilidad. Así, en esta sección se hablara de las reglas de usabilidad que se intentarán cumplir:

Facilidad de aprendizaje: minimizar el tiempo que se requiere desde el no conocimiento de una aplicación hasta su uso productivo.

Tiempo de respuesta: capacidad del software de expresar los cambios de estado del usuario. Este factor es muy variable, ya que depende de las características que tenga el equipo del usuario.

Flexibilidad: formas de intercambiar la información el usuario con el sistema. Aportar flexibilidad al sistema implica brindar control al usuario, capacidad de sustitución y capacidad de adaptación.

Robustez: caracteriza la necesidad de que el usuario cumpla con sus objetivos y que disponga del asesoramiento necesario.

Recuperabilidad: grado de facilidad que una aplicación permite al usuario para corregir una acción una vez está reconocido un error.

Consistencia: Es la capacidad de utilizar de la misma manera todos los mecanismos, sea cualquiera el momento que se necesite.

Disminución de la carga cognitiva: los aspectos cognitivos de la interacción proporcionan la necesidad que tienen los usuarios de confiar más en los reconocimientos que en los recuerdos (no tienen que recordar abreviaciones y códigos muy complicados). Este aspecto condicionará la disposición y el diseño de los distintos elementos interactivos que aparecerán en la interfaz.

3.2. Casos de uso

En esta sección se recogerán los diagramas de casos de uso propios de los actores que utilizarán la aplicación y algunos de los casos más importantes que forman estos diagramas. Para una mejor visualización de las imágenes y hacer más sencillo la comprensión de los diagramas se separarán en tres imágenes que corresponderán con los tres tipos de actores a los que está enfocada la aplicación: cuentas refiriéndose a los pacientes, que a su vez englobará a los usuarios que compartan una cuenta, médicos y administradores.

3.2.1. Cuentas de los pacientes

A continuación se mostrará el diagrama de casos de uso que contiene los casos de uso disponibles para los pacientes y los dos casos de uso más representativos: visualizar eventos y añadir evento periódico. Los casos de uso restantes se pueden encontrar en el anexo A.

Figura 1: Diagrama de casos de uso de las cuentas para pacientes.

Caso de uso	Visualizar eventos
Descripción	El usuario verá por pantalla, colocados en el calendario o en una lista que contenga los eventos de cada día, los eventos que tiene asociados y de qué tipo son cada uno.
Actores	Usuario.
Precondiciones	El usuario está autenticado en el sistema.
Pasos	1. El usuario accede a un mes o a un día del calendario. 2. El sistema obtiene los eventos cuya fecha sea el mes en el que se sitúa el calendario o el día que se está visualizando y los muestra por pantalla.
Variaciones	

Caso de uso	Añadir evento periódico
Descripción	El usuario añadirá un evento indicando una periodicidad: diario, semanal o mensual. También deberá indicar una fecha de comienzo y una fecha de finalización del evento.
Actores	Usuario.
Precondiciones	El usuario está autenticado en el sistema.
Pasos	1. El usuario accede a un día del calendario y selecciona la opción de añadir un evento. 2. El usuario indica que desea que el evento sea periódico y completa los campos requeridos. 3. El sistema añade el evento con la periodicidad seleccionada en los días correspondientes.
Variaciones	

3.2.2. Cuentas de los médicos

A continuación se mostraran el diagrama de casos de uso que contiene los casos de uso disponibles para los médicos y los dos casos de uso más representativos: visualizar valores de pruebas y añadir rangos de pruebas. Los casos de uso restantes se pueden encontrar en el anexo A.

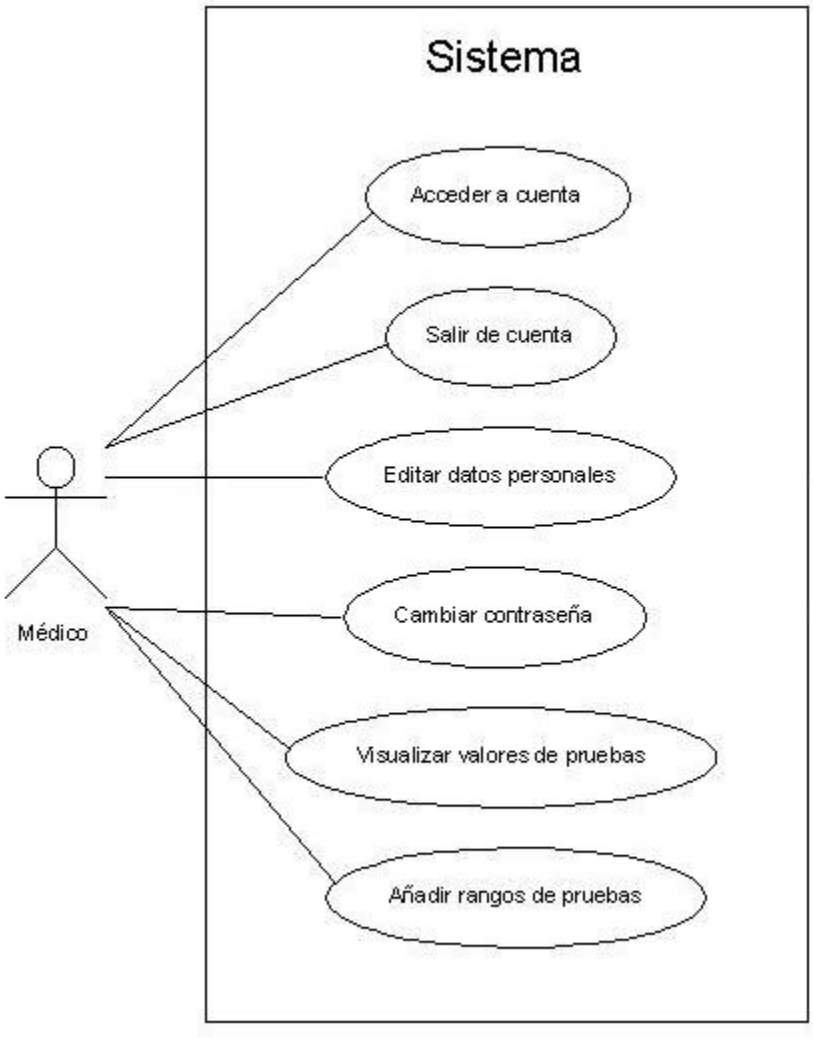


Figura 2: Diagrama de casos de uso de las cuentas para médicos.

Caso de uso	Visualizar valores de pruebas
Descripción	El médico podrá ver a través de un gráfico lineal todos los datos que el paciente seleccionado tiene almacenados de sus distintas pruebas activas. Al médico le será posible mostrar u ocultar los datos de una determinada prueba.
Actores	Médico.
Precondiciones	El médico está autenticado en el sistema.
Pasos	1. El médico accede a la lista de pacientes. 2. El médico selecciona la opción del paciente de la columna que contenga las estadísticas y accede a visualizar el gráfico.
Variaciones	

Caso de uso	Añadir rangos de pruebas
Descripción	El médico añadirá unos valores permitidos para una prueba. Esos valores indicaran un rango que en caso de que el paciente lo sobrepase con algún valor tomado por sus pruebas el sistema deberá mostrarle un aviso.
Actores	Médico.
Precondiciones	El médico está autenticado en el sistema. El paciente tiene alguna prueba activa.
Pasos	1. El médico debe acceder a la lista de pacientes. 2. El médico selecciona la opción del paciente de la columna que contiene las estadísticas y accede la pantalla donde visualiza el gráfico con los valores de las pruebas del paciente. 3. El médico cambia a la pestaña en la que debe seleccionar a que prueba desea añadir rangos.
Variaciones	

3.2.3. Cuentas de los administradores

En esta sección se mostrará el diagrama de casos de uso que contiene los casos de uso disponibles para los administradores y sus dos casos de uso más representativos: añadir cuentas y asignar paciente/s a médico. Los casos de uso restantes se pueden encontrar en el anexo A.

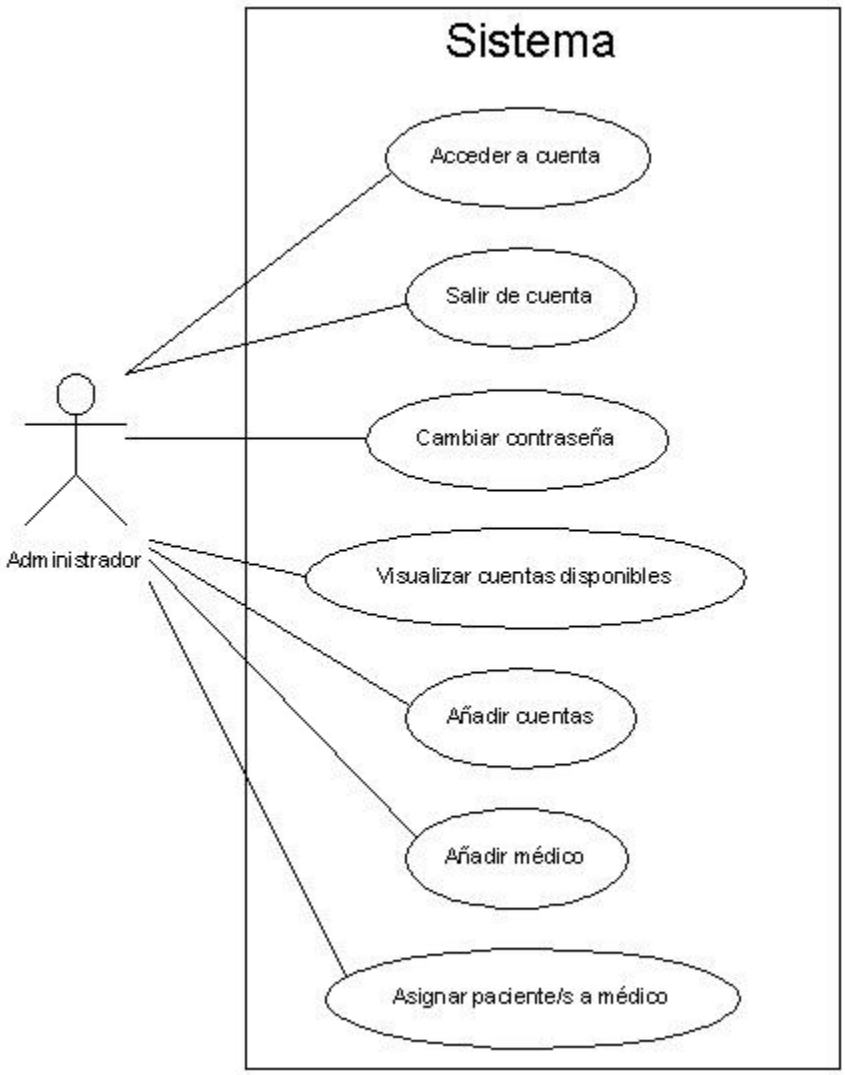


Figura 3: Diagrama de casos de uso de las cuentas para administradores.

Caso de uso	Añadir cuentas
Descripción	El administrador le proporcionara al sistema un fichero .TXT que contendrá las cuentas que el equipo de soporte de Etherios le ha proporcionado. Una vez se le hayan proporcionado al sistema este las añadirá a la base de datos.
Actores	Administrador.
Precondiciones	El administrador está autenticado en el sistema.
Pasos	1. El administrador accede a la pantalla de carga del fichero .TXT 2. El administrador selecciona el fichero .TXT de su equipo. 3. El sistema añade a la base de datos las cuentas contenidas en el fichero .TXT.
Variaciones	

Caso de uso	Asignar paciente/s a médico
Descripción	El administrador seleccionará un médico y uno o varios pacientes que quedaran asignados al médico seleccionado.
Actores	Administrador.
Precondiciones	El administrador está autenticado en el sistema.
Pasos	1. El administrador accede a la ventana de asignación. 2. Selecciona el médico y el paciente o los pacientes que se desean asignar. 3. El sistema actualiza la base de datos asignando al médico los pacientes seleccionados.
Variaciones	

Capítulo 4

Diseño

En este capítulo se explicarán las decisiones que se han tomado referentes al diseño de la interfaz, la base de datos y se explicará el funcionamiento paso a paso de algunas de las partes más complejas de la aplicación.

4.1. Diseño de la interfaz de usuario

La interfaz de usuario es como ya se ha indicado una parte muy importante a tener en cuenta debido a que la aplicación está orientada a un público que no necesariamente va a poseer conocimientos informáticos, por lo tanto se ha perseguido una interfaz general, fácil de aprender para cualquier tipo de usuario, intuitiva y usable.

Para tal objetivo se realizaron una serie de mockups digitales los cuales van a ser la guía a la hora de dar el aspecto al portal web. Estos mockups no han sido evaluados independientemente sino que han debido guardar siempre unas similitudes con los realizados para la parte web. Todo esto no persigue otro objetivo que el de dar al usuario una experiencia casi igual independientemente de qué aplicación esté utilizando, ya sea la aplicación web o la aplicación Android.

En este punto cabe destacar que la interfaz final que presenta el proyecto no es la que se escogió desde un principio ya que ha sufrido dos cambios sustanciales desde la primera interfaz propuesta a la empresa. Estos cambios se centraron desde reducir el número de elementos textuales y ampliar el número de imágenes que nos guiasen a través de la navegación por el portal web, hasta adaptar los colores de los CSS a los colores corporativos.

En los siguientes apartados se mostraran las partes más importantes de la interfaz de usuario y se explicarán las decisiones que se han tomado en cada una de estas partes.

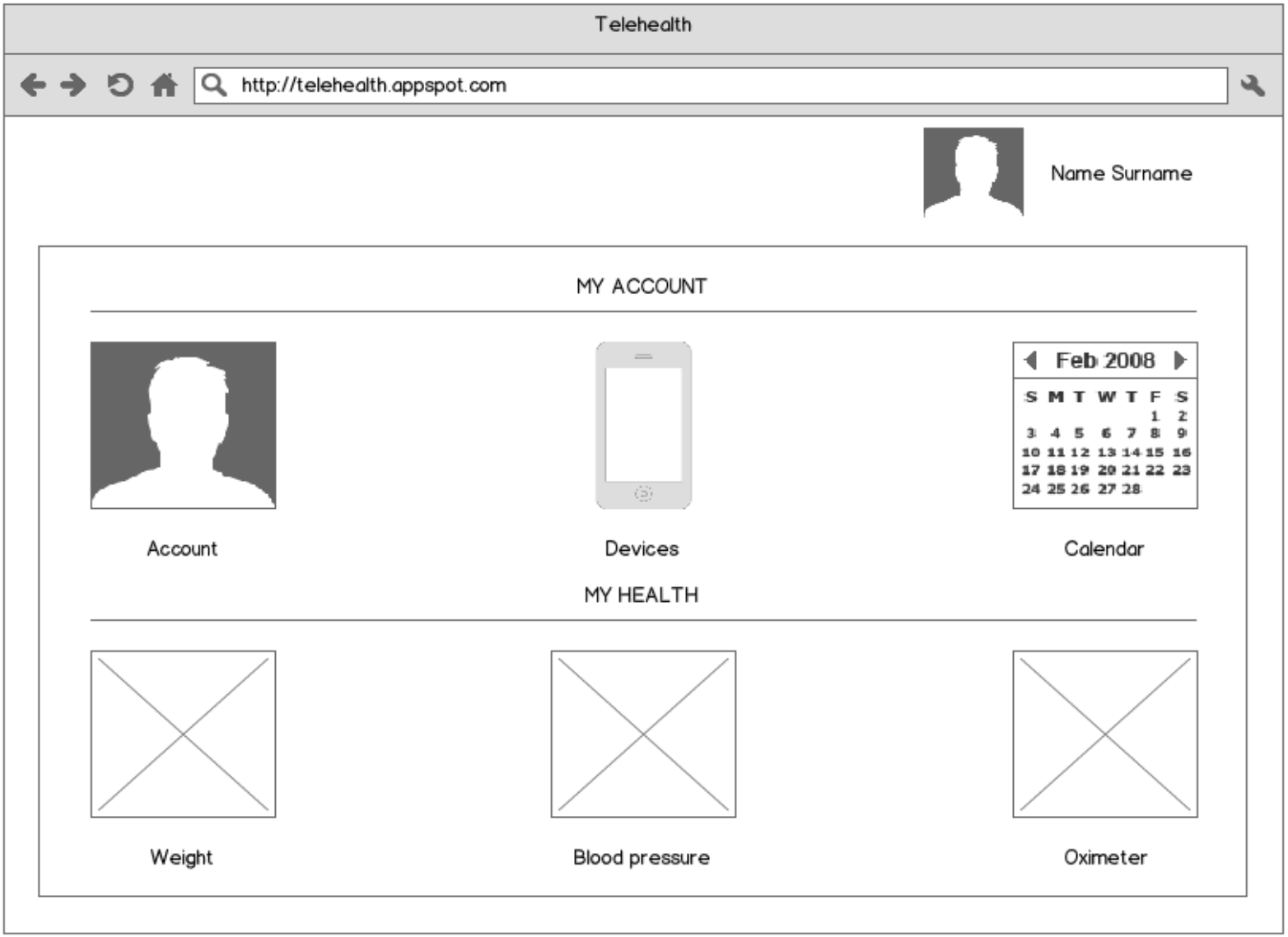


Figura 4: Ejemplo de uno de los elementos mockups realizados para la primera parte del diseño.

4.1.1. Generalidades

Las partes generales de la interfaz son una mezcla entre algunos elementos específicos tomados del diseño corporativo de la página de Etherios, como los colores o la distribución de algunos elementos en la página.

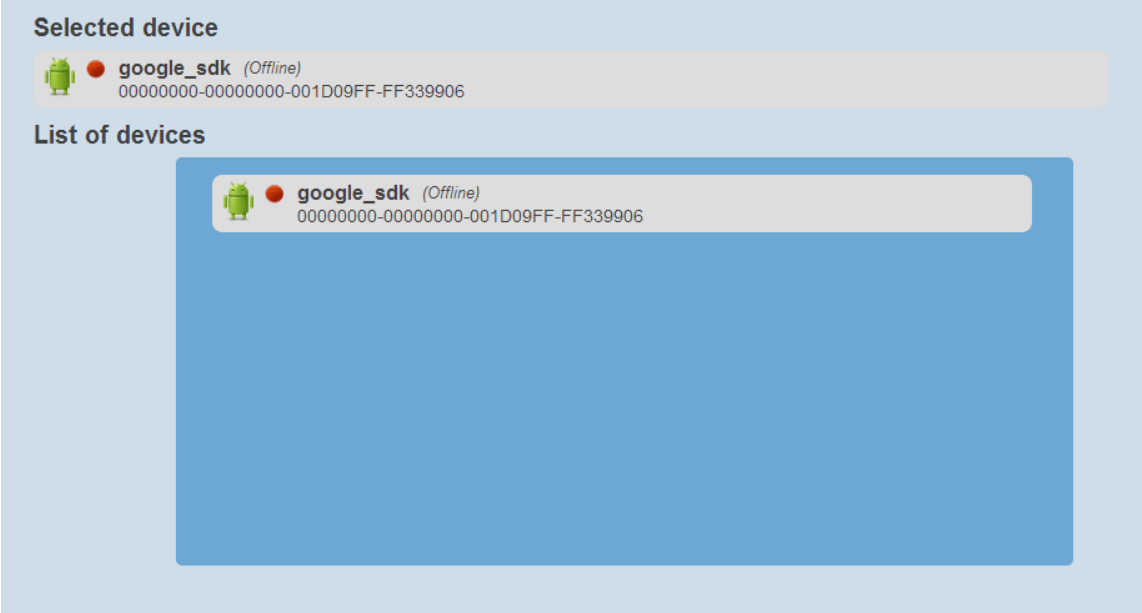


Figura 5: Ejemplo de uno de los elementos gráficos adquiridos de las demos de Digi.

Para obtener estos colores y elementos sea ha utilizado como guía orientativa varios proyectos proporcionados por la empresa en la etapa que se realizaron las practicas en empresa para tener un mayor conocimiento de la forma de trabajar de esta y por lo tanto el proyectante no partió de cero a la hora de desarrollar algunos de los elementos visuales del portal web.

El resto de requisitos de la interfaz de usuario se eligió tras varias reuniones con el cliente en la que fueron presentados distintos mockups orientativos y llegando a unos acuerdos como que las ventanas no debían tener scroll y que se deberían de simplificar casi todos sus elementos a imágenes.

4.1.2. Gráficos

Para las partes que contienen los gráficos lineales que van a mostrar, tanto a los pacientes como a los médicos, los valores que el usuario tiene guardados para las distintas pruebas que utiliza se propusieron dos librerías de gráficos: Highcharts y Google Charts, eligiendo finalmente Highcharts. Esta decisión podría ser más propia del

apartado de implementación, pero ya que los motivos por los que se eligió fueron por temas de estética y usabilidad para los usuarios se incluirán los gráficos en este apartado.

La librería Highcharts proporciona muchas opciones en el apartado gráfico pero la más destacable y principalmente por la cual ha sido para este proyecto es que nos proporciona la posibilidad de seleccionar que líneas deseamos visualizar en nuestros gráficos y cuales queremos ocultar. Esto será de mucha ayuda para los médicos a la hora de realizar el seguimiento de las pruebas de un paciente, teniendo todas las pruebas en una misma ventana pero permitiéndole ocultarlas si se da un exceso de información o algún gráfico interfiere con otro.

Para finalizar también cabe destacar el abanico de posibilidades que ofrece Highcharts a la hora de elegir un tema que se adaptase al diseño de nuestro portal web.

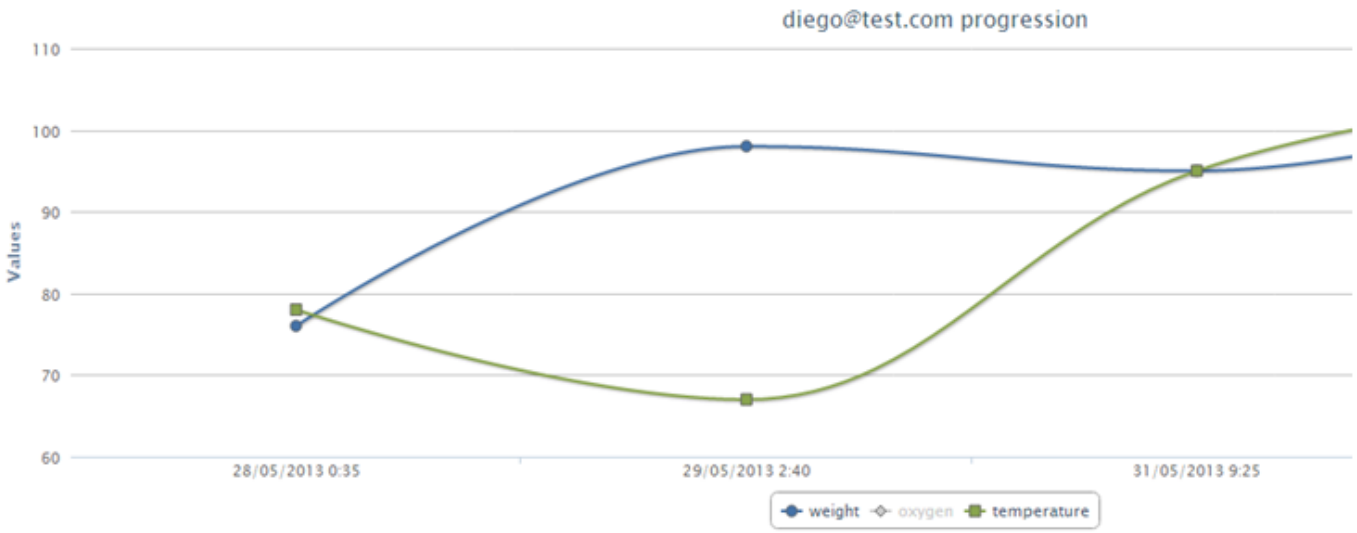


Figura 6: Ejemplo de un gráfico con una prueba oculta y dos visibles.

4.1.3. Calendario

En el calendario se ha buscado conseguir que sea un elemento simple pero que a su vez muestre mucha información al usuario sobre los eventos que tiene asignados. Para esto se ha diseñado un calendario que se encargue de mostrar, con ayuda de tres colores, que tipo de evento contiene cada día . Para esto se investigaron otros calendarios que tuviesen un diseño similar escogiendo finalmente basarse en el Google Calendar. Los detalles sobre que tres colores se han asignado a cada uno de los tres tipos distintos de eventos se detallará más adelante en el apartado de implementación.

4.1.4. Menú principal

Para el menú principal se optó por un diseño ligero, libre de carga cognitiva para conseguir que los usuarios tuviesen una sensación de control sobre la pantalla que les va a proporcionar el acceso a todas las secciones de las que dispone el portal web. Por lo tanto se ha optado por seguir un modelo que no solo cumpliese estas características sino que también fuese lo más simétrico con la aplicación Android para que la curva de aprendizaje del usuario se viese reforzada. Este modelo es el de representación de las secciones por iconos que el usuario asocie a estas secciones ayudándole con un título para la sección.

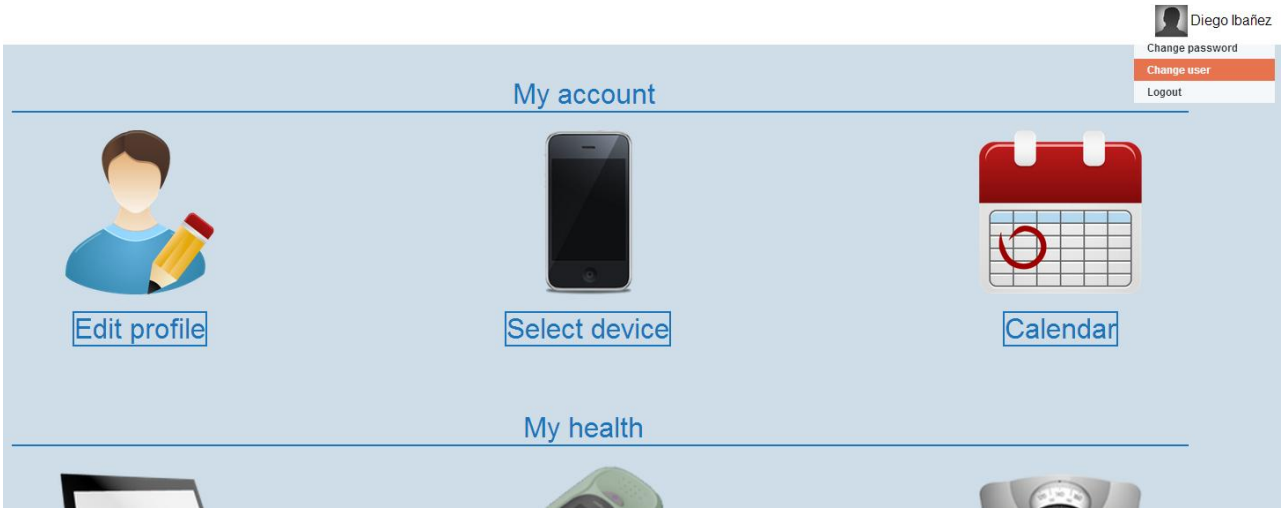


Figura 7: Ejemplo del menú principal para los pacientes.

4.2. Diseño de la base de datos

Con relación a la base de datos no se contó con la opción de elegir la forma en la que la se iba a implementar ya que uno de los requisitos de trabajar con Google App Engine es que cuenta con una estructura de base de datos especifica que consiste en una estructura JDO(Java Data Objects). Esta estructura no tiene complicaciones desde el punto de vista de la implementación pero sí que exige un cambio a la hora de diseñar la base de datos respecto al tipo de base de datos relacionales estudiadas en las distintas asignaturas de la titulación.

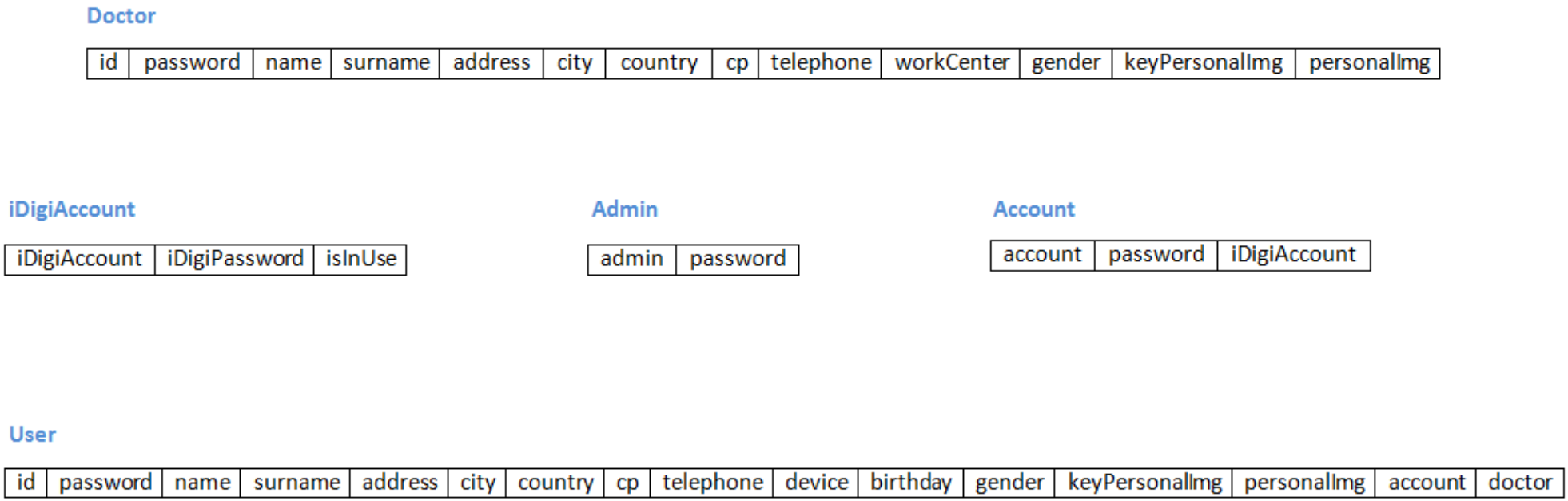


Figura 8: Modelo relacional de la base de datos de la aplicación.

Pese a este cambio de estructura en la base de datos se ha buscado el simular el diseño como si de una base de datos relacional se tratase para conseguir un mejor entendimiento de la misma. Añadir que esta adaptación acabó siendo más fácil que lo que en un principio se planteaba, ya que al tratarse de un diseño orientado a objetos la primera idea del proyectante fue realizar un diagrama de clases para establecer las relaciones pero conforme se avanzaba se abandonó esa idea y se sustituyó por un modelo relacional, estableciendo las clases como tablas y las propiedades como los atributos como se puede ver en la figura 8.

Capítulo 5

Implementación

En este capítulo se indicarán que tecnologías se han utilizado para desarrollar el proyecto, cuáles han sido los lenguajes de programación elegidos y los motivos de las elecciones y se mostrarán tres de los algoritmos más representativos del proyecto.

5.1. Tecnologías utilizadas

5.1.1. Entorno de desarrollo

La elección del entorno de desarrollo como de los lenguajes a utilizar para desarrollar la aplicación como ya veremos más adelante, fue propuesto por el cliente aunque dio opción de proponer otros. Tras detallar los requisitos finales del producto, las tecnologías que se iban a emplear y conocer los motivos que llevaban al cliente a pedir al proyectante el uso de la tecnología propuesta se decidió que el entorno de desarrollo con el que se iba a llevar a cabo el proyecto iba a ser Eclipse. Los motivos por lo que se ha elegido Eclipse son los siguientes:

- Se trata de un entorno de desarrollo conocido para el proyectante, del cual se conoce muchas opciones de configuración y que por lo tanto se ahorraría el tiempo dedicado al aprendizaje y a la configuración.
- Como se verá más adelante, cuando se hable los lenguajes de programación elegidos, Eclipse es una opción que nos permite desarrollar en todos los lenguajes necesarios para realizar el proyecto.
- Cuenta con la posibilidad de añadir una gran cantidad de plug-ins, esta razón es una de las que más peso han tenido en la elección del entorno de desarrollo ya que gracias a estos plug-ins se ha reducido mucho esfuerzo a la hora de desplegar la aplicación en Google App Engine ya que Google cuenta con varias guías de integración con Eclipse y mucha documentación sobre la utilización de esta herramienta. Otro de los plug-ins que se sabía que era compatible con Eclipse era el plug-in que tiene a su disposición para tener un sistema de control de versiones como Git, con el cual el proyectante debía trabajar a petición del cliente para tener almacenados los cambios que iba sufriendo el proyecto.
- Era el entorno de desarrollo que se había utilizado en el periodo de prácticas y todo el material aprovechable estaba desarrollado en Eclipse, por lo tanto se valoró mucho la continuidad que proporcionaba y evitar posibles problemas de migración de código por incompatibilidades con alguna de las librerías empleadas.

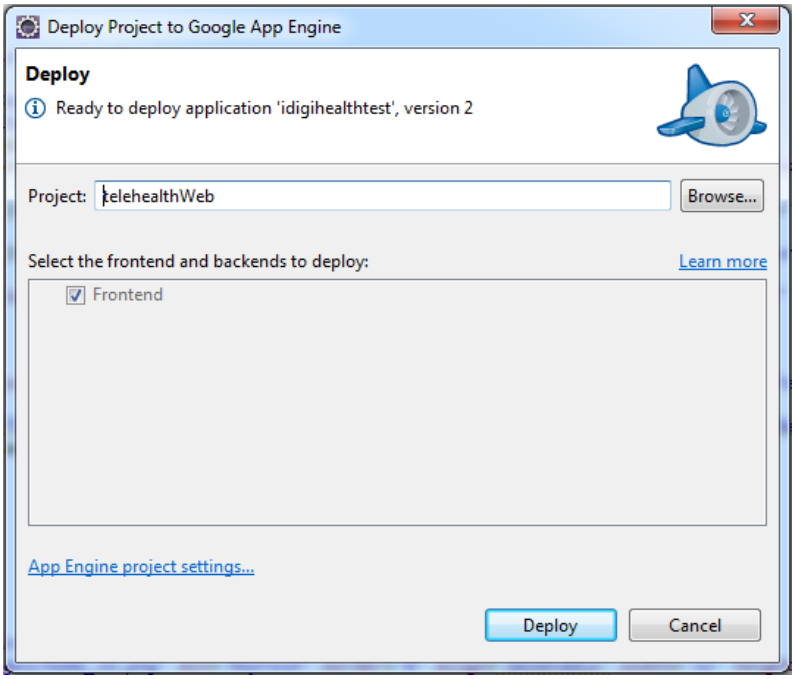


Figura 9: Ejemplo del plug-in que gestiona el despliegue de un proyecto en Google App Engine.

5.1.2. Lenguajes de programación

Para el desarrollo del portal web se han empleado cinco lenguajes de programación distintos: Java, JavaScript, JSP, HTML y CSS. A continuación se va proceder a detallar los motivos por los que se han elegido algunos en los que había posibilidad de haber utilizado otro lenguaje y a explicar que partes se han desarrollado con cada uno de ellos.

Java

La elección de Java al igual que otras partes como el entorno de desarrollo no ha sido una elección que se haya tomado de forma individual sino que ha entrado en un conjunto que engloba al servidor donde se va a desplegar la aplicación, lenguajes de programación que domina el cliente para que pueda realizar un posterior mantenimiento o ampliar el portal y entornos de desarrollo que soportasen todo esto. La primera de estas opciones nos obligaba a dos posibles lenguajes de programación para crear los Servlets encargados de la parte del servidor: Java o Python. Así que por la experiencia previa que se tenía sobre Java y por seguir con la dinámica de unidad entre la aplicación Android, la cual utiliza también este lenguaje de programación, y la aplicación web se ha escogido este lenguaje para la realización de esta parte del proyecto.

Las partes que se han realizado en Java son los Servlets que se encargan de la generación dinámica de los contenidos que el servidor envía al navegador de los usuarios. Dicho de una forma más simple los Servlets son los encargados de tareas como autentificar en cada momento que el usuario al que se le va a responder una petición es el correcto, cerrar las sesiones que proporcionan la falsa ilusión de que HTML tiene estado, paso de parámetros entre pantallas, etc.

```
public class LogoutServlet extends HttpServlet {

	protected void doGet(HttpServletRequest request, HttpServletResponse
response)
	throws ServletException {

		try {

			HttpSession sesion = request.getSession(false);

			if(sesion!=null)
			{
			    sesion.invalidate();
			}
			response.sendRedirect("/Login");

		} catch (IOException e) {
			e.printStackTrace();
		}

	}
}
```

Figura 10: Ejemplo de un Servlet encargado del cierre de sesión de un usuario.

También, como ya se ha indicado más arriba, se ha empleado para crear las clases que componen la base de datos especial que utiliza Google App Engine que componen los objetos llamados entidades.

JavaScript

JavaScript es un lenguaje de programación interpretado, definido como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico que se ha utilizado para mejorar la experiencia del usuario haciendo la interfaz más dinámica y atractiva. A todo esto hay que sumarle partes en las que se ha utilizado, a través de JavaScript que trabaja en la parte del navegador del cliente, Ajax. Ajax es el encargado de que se establezca un comunicación en tiempo real entre el navegador y el servidor de

forma completamente transparente para el usuario dando una alta sensación de dinamismo a las páginas.

JSP

Se trata de una tecnología que encargada de crear páginas web dinámicas basadas en HTML y XML entre otros tipos de documentos. JSP es similar a PHP pero usa el lenguaje de programación Java. Y por la elección de Java, porque en la fecha de inicialización del proyecto Google App Engine no permitía el uso de PHP en sus proyectos y por el interés tanto del proyectante como del cliente por el uso de este lenguaje, fue el elegido para el desarrollo del proyecto.

```
<c:choose>
  <c:when test="${user.keyPersonalImg!=null}">
     <img src="/serve?blob-key=${user.keyPersonalImg}" alt="Family" />
  </c:when>
  <c:otherwise>
     <img src="images/user_default.png" alt="Family" />
  </c:otherwise>
</c:choose>
```

Figura 11: Ejemplo de JSP encargado de comprobar si el usuario tiene una imagen de perfil.

HTML y CSS

Para finalizar, indicar que también se ha utilizado HTML, que se trata del principal lenguaje a la hora de construir páginas web y que se basa en un sistema de marcado y CSS que es un lenguaje que se encarga de definir los estilos usados en un documento que utilice el sistema de marcas.

5.2. Librerías utilizadas

5.2.1. Prototype

Prototype es un framework escrito en JavaScript que se orienta al desarrollo sencillo y dinámico de aplicaciones web. Es una herramienta que implementa las técnicas AJAX y se ha utilizado en la aplicación para conseguir unos pop ups con unas propiedades muy concretas. Estas propiedades consisten en: mantener el pop up generado en la parte central de la pantalla para una mejor visibilidad, el uso de Ajax a través de los

elementos que utiliza esta librería y el mantener el pop up sobre el fondo de tal forma que mientras que no acabemos de realizar las operaciones que deseemos en el pop up o lo cerremos no podremos seguir con otras actividades, todo esto con la posibilidad de personalizar la interfaz de usuario.

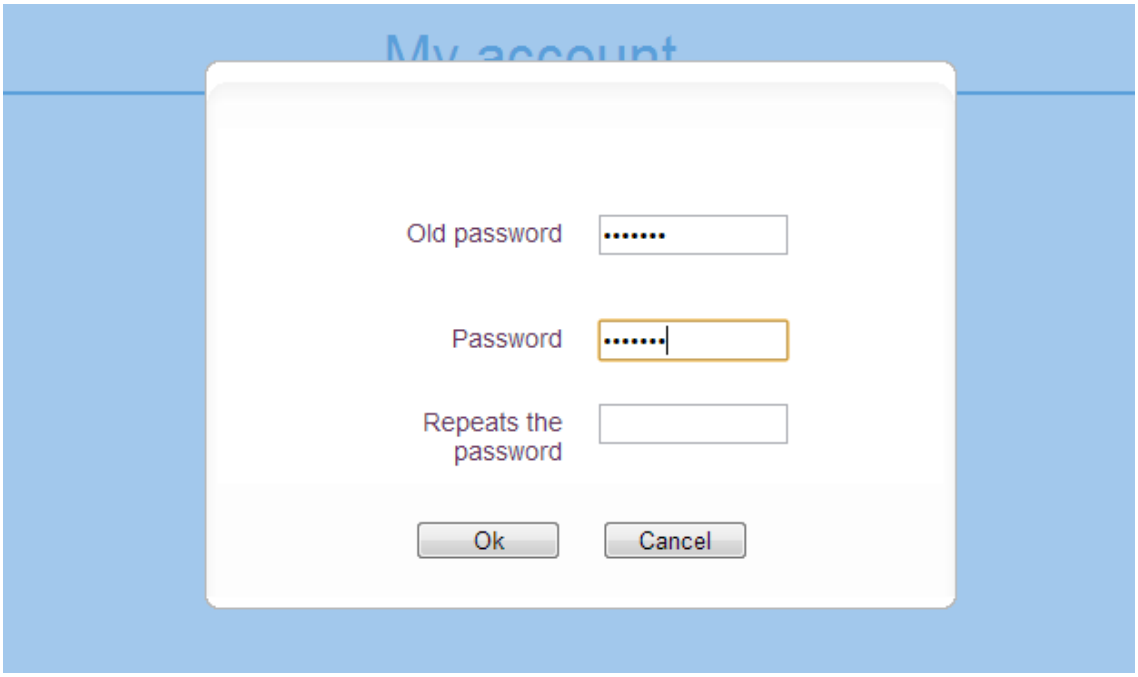


Figura 12: Ejemplo de un pop up obtenido mediante la librería Prototype.

5.2.2. Coda Slider

Se trata de una librería que proporciona la funcionalidad necesaria para implementar pantallas que funcionen como un slide en el que podremos personalizar varias opciones, en concreto para este proyecto se ha utilizado para controlar el paso automático de diapositivas y generar distintos eventos según en que slide esté situado el usuario. Esto es de gran ayuda a la hora de poder actualizar los datos y de tener operaciones tan distintas como la visualización y la inserción de datos, todo esto sin tener que cambiar de ventana. Por otra parte también ofrece una apariencia bastante similar a la que proporcionan los sistemas operativos móviles a la hora de navegar con varias aplicaciones y que cada vez más se está trasladando a la web.

5.2.3. Highcharts

Esta librería, de la que ya se ha hablado de su funcionalidad y de los motivos por los que se eligió ante otras opciones disponibles en la parte del diseño, cuenta con una magnifica API para desarrolladores en la cual podemos probar los distintos gráficos que nos ofrece antes de ponernos a implementar y ver un modelo aproximado de lo que será nuestro modelo final. Por lo tanto estamos ante una librería que rompe con el tópico de que las APIS de JavaScript no son demasiado usables a la hora de buscar propiedades mientras estamos implementando.

5.2.4. JSCal 2

JSCal 2 es una librería que proporciona un calendario fijo o desplegable y la posibilidad de configurar nuestros propios eventos. La implementación de esta librería ha obligado a crear dos ficheros CSS debido a falta de una API oficial y de opciones a simple vista de configuración, ya que el cliente pidió dos tipos de calendarios dinámicos para esta aplicación.

El primero sé trata el calendario de pequeñas dimensiones que suele acompañar a los formularios para ayudar al usuario a poder navegar por un calendario sin tener que salir de la propia ventana de la aplicación y para que no haya errores a la hora de introducir la fecha por parte de los usuarios de distintas nacionalidades(que según al país que pertenezcan tendrán un formato especifico de fecha). Este calendario ha sido implementado con el CSS que trae el paquete descargable de la librería.

Para la segunda forma de implementación del calendario, siguiendo las especificaciones dadas por el cliente, se necesitaba un calendario dinámico que fuese de grandes dimensiones que mostrase a simple vista, ayudándose de distintos colores representativos, los eventos disponibles para el mes en el que el usuario se encuentra situado. Por lo tanto hubo que hacer una remodelación completa del CSS y añadir distintos métodos JavaScript que se encargasen de gestionar los eventos al pasar de mes o año.

Cabe destacar, y más en comparación con la librería Highcharts mencionada en el punto anterior, que ante la ausencia de API muchas veces el proceso de implementación estuvo basado en el método heurístico de ensayo y error haciendo más costosa la implementación del calendario de lo que en un principio se había planificado.

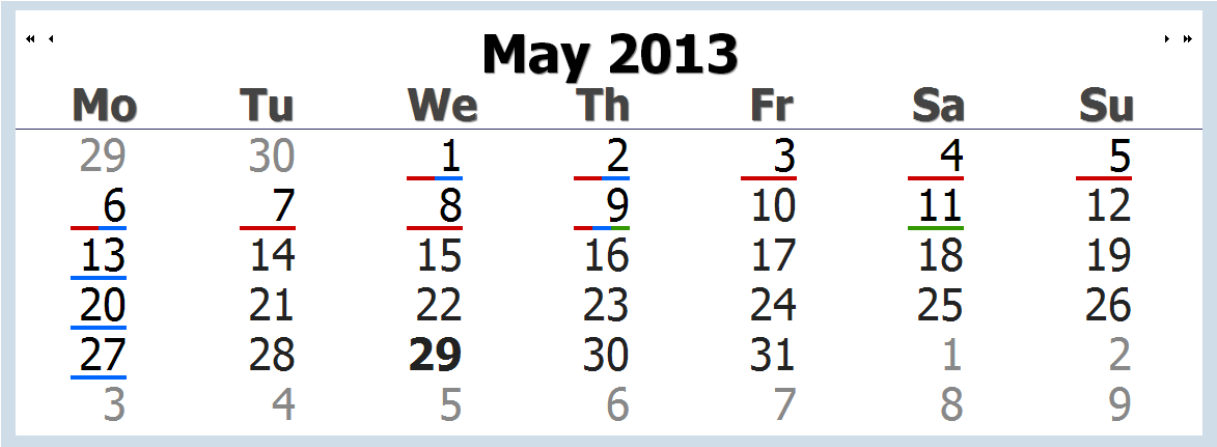


Figura 13: Ejemplo del resultado obtenido en el segundo caso de implementación de JSCal 2.

5.2.5. JQuery

JQuery se trata de un framework que se ha utilizado tanto como para posibilitar el funcionamiento de otras librerías como para posibilitar el uso de AJAX y poder implementar algunos de los métodos JavaScript. En casos como el de la pantalla que muestra los eventos de un día ha dado problemas de compatibilidad con Prototype pero que se han podido solucionar gracias a una opción que incluye JQuery para este tipo de conflictos.

```
var $$ = jQuery.noConflict();
```

Figura 14: Ejemplo de cómo cambiar la referencia a JQuery para que no ocasione conflictos con otra librería.

5.3. Algoritmos representativos

5.3.1. Algoritmo de conexión con la nube de Etherios y de actualización del XML que contiene los valores de una determinada prueba

El algoritmo, implementado en Java, se encarga de abrir una conexión con la nube utilizando los datos de acceso, que tiene asociados a la cuenta, guardados en la base de datos de Google App Engine:

```
public static boolean addNewSample(String iDigiAccount, String folderName,
Sample sample) {
             HttpURLConnection conn = null;

        try {
            // Create url to the iDigi server for a given web service request
            URL url = new
URL("http://login.etherios.co.uk:80/ws/FileData/~/"+folderName);
            conn = (HttpURLConnection) url.openConnection();
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setRequestMethod("GET");

            // Build authentication string
            iDigiAccount iDigi=
iDigiAccountUtil.getIDigiAccountById(iDigiAccount);
            String userpassword = iDigi.getiDigiAccount() + ":" +
iDigi.getiDigiPassword();

            // can change this to use a different base64 encoder
            String encodedAuthorization =
Base64.encodeBase64String(userpassword.getBytes()).trim();

            // set request headers
            conn.setRequestProperty("Authorization", "Basic "
                    + encodedAuthorization);
            // Get input stream from response and convert to String
            InputStream is = conn.getInputStream();

            Scanner isScanner = new Scanner(is);
            StringBuffer buf = new StringBuffer();
            while (isScanner.hasNextLine()) {
                buf.append(isScanner.nextLine() + "\n");
            }
            String responseContent = buf.toString();
            isScanner.close();
```

Como se puede ver en el anterior algoritmo a partir de la línea 17 los datos de acceso deben ir cifrados, es un requisito para que no dé errores, y nos sirve para obtener los datos de cualquiera de los ficheros de las distintas pruebas. Esta parte es muy similar siempre que se desee establecer una conexión con la nube de Etherios. Tras establecer la conexión se comprueba que no se han producido problemas a la hora de recibir los datos tanto porque ha ocurrido un fallo de conexión o porque los datos solicitados no existían:

```
String newText="";

            int ends=responseContent.lastIndexOf("</sample>")+9;

            if(responseContent.contains("GET FileData error. Error reading
FileData entity"))
            {
             newText="<?xml version=\"1.0\"?>\r\n<samples>";
             ends=0;
            }
```

Y tras realizar esta comprobación para evitar futuros problemas estructurales con el XML el sistema añade los datos recogidos para la prueba seleccionada, cierra el XML y cierra las conexiones que se han abierto con la nube:

```
String first=responseContent.substring(0, ends);
            String
add="\r\n<sample>\r\n<id>01</id>\r\n<date>"+sample.getDate()+"</date>\r\n<hou
r>"+sample.getHour()+"</hour>\r\n<observations></observations>\r\n<value>"+sa
mple.getValue()+"</value>\r\n<classification></classification>\r\n</sample>\r
\n";
            String last="</samples>";

            String newXMLString=newText+first+add+last;

            createAndUploadNewXMLFile(iDigi,newXMLString, folderName);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (conn != null)
                conn.disconnect();
        }
             return false;
      }
```

5.3.2. Algoritmo encargado de colocar una notificación visual en los días del calendario con eventos

Se trata de un algoritmo implementado en JavaScript que se encarga de colocar distintas marcas de colores según el tipo de evento que el usuario tenga activo en cada día del mes y de actualizar el calendario a través de una petición AJAX siempre que el usuario se desplace a otro mes o año.

Este algoritmo está dividido en dos funciones, la primera es un método recursivo al que en la cabecera le llega un objeto que contiene la fecha con el que se crea el calendario y

que cada vez que se avanza o retrocede a otro mes o año se dispara un evento que se encarga de que el método se llame a sí mismo pasándose la nueva fecha actualizada como parámetro:

```
function moveCalendar(calendarObject)
{
	document.getElementById("main").innerHTML='<table class="description-
table" summary="Step"><tr><td><h2>Calendar</h2><p class="main-text">You can
see what events you have in this month according the color or color that the
days contains.</p></td></tr></table>';

	now = calendarObject;

	var cal1 = Calendar.setup({ cont: "main", date: now, onclick:
function() {}, onSelect: function() {redirectDate(this)} });

	cal1.addEventListener("onChange", function(cal1, now) {

		now.setMonth(now.getMonth());

		document.getElementById("main").innerHTML="";

		moveCalendar(now);
	});

	if((now.getMonth()+1)<10)
	{
		month="0"+(now.getMonth()+1);
	}
	else
	{
		month=(now.getMonth()+1);
	}

	eventsInDay(month, (now.getFullYear()));
}
```

Siempre que no se lance el evento de que se realice un cambio de mes o año se llamará al segundo método que compone el algoritmo. Este método se encargará de pedir al servidor que días del mes y año en el que tenemos posicionado el calendario tienen eventos y con cuántos de los tres tipos de eventos(medicine, test y/o doctor) cuenta ese día.

Para saber que tipos distintos de eventos están activos cada día, y por lo tanto saber cuántos colores debe contener la notificación de ese día, se elaboró una tabla como la siguiente:

Tipos de eventos que contiene el día	Número que se le asigna al día
Test	1
Doctor	2
Medicine	3
Test y Doctor	4
Test y Medicine	5
Medicine y Doctor	6
Test, Medicine y Doctor	7

Por lo tanto al recibir la respuesta de AJAX nos llegará un objeto del tipo hashmap para que aquellos días que tengan eventos se puedan modificar y que muestren una notificación visual que, como se ha explicado en la parte encargada de describir el diseño, ayudara al usuario a ser consciente a simple vista del número aproximado de eventos que tiene.

5.3.3. Algoritmo generador de eventos periódicos

Este algoritmo, que forma parte del método doPost del servlet EventsDay, es el encargado de crear, una vez confirmado que el evento va a ser de tipo periódico, todos los eventos que se encuentran entre dos fechas indicadas por el usuario en el formulario que realiza la llamada del método. Tras comprobar que se debe generar un evento se pasaría a, con ayuda de una fecha auxiliar, crear los eventos según su nivel de repetición y llamar al método que los añadirá al fichero correspondiente en la nube de Etherios como se muestra a continuación:

```
if(period.compareTo("weekly")==0)
{

	while(cal1.compareTo(cal2)<1)
	{
	Event event=new Event(title, description, type,
	formatoDelTexto.format(cal1.getTime()), hour, alarm);

	Utils.addNewEvent(a.getiDigiAccount(),
	"telehealth/"+u.getId().replace("@", "-.at.-
	")+"/events/dataEvent.xml", event);


	cal1.add(Calendar.DATE, 7);
	}
}
```

Capítulo 6

Pruebas

En este capítulo se detallará brevemente cómo se han ido realizando las pruebas necesarias del proyecto.

6.1. Metodología para la realización de las pruebas

Debido a la forma en la que se ha desarrollado el proyecto la fase de pruebas ha sido constante y ha estado presente desde el principio del proyecto. Esto es debido a las reuniones semanales con el cliente para presentarle los avances realizados la fase de pruebas ha quedado diluida con la fase de implementación y las distintas iteraciones durante las presentaciones.

No obstante, antes de las presentaciones semanales al cliente, el proyectante siempre se aseguraba de probar los requisitos más básicos como comprobar que la integridad del material presentado en otras reuniones no había sido alterado, que la interfaz gráfica aunque no estuviese terminada fuese lo más agradable posible o la distinta compatibilidad de la aplicación web con los navegadores utilizados.

Finalmente una vez acabado todo el portal, el proyectante dio acceso al personal especifico de realizar las pruebas por parte del cliente antes de que este diese por finalizado el proyecto. En esta fase de pruebas se encontraron problemas relacionados con la comunicación del portal web con la aplicación Android, debidos a la entrada de datos en los formularios realizados con la librería Prototype por lo que el proyectante re-abrió la parte del proyecto donde se encontraban los formularios para conseguir que la entrada de datos fuese la esperada por ambas aplicaciones.

Añadir que se realizó un periodo de pruebas sobre el portal, cuando este estaba en sus primeras etapas del desarrollo, para determinar las causas de la lentitud del mismo. Estas pruebas indicaron que la causa era el alto tiempo de respuesta que ofrece Google App Engine en su versión gratuita, problema solucionable obteniendo la versión de pago.

Capítulo 7

Conclusiones y propuestas

En este capítulo se describirán las conclusiones derivadas de la realización del proyecto y las propuestas que el proyectante considera interesantes para la aplicación de la aplicación.

El proyecto descrito es hasta ahora es el primer proyecto real al que el proyectante ha tenido que enfrentarse, a este reto también hay que añadirle la experiencia de trabajar para un cliente, que en este caso se trata de la empresa donde se desarrollaron las prácticas, y que fuese un proyecto que aunque se hubiese podido quedar en un proyecto individual ha ido más allá combinándolo con un proyecto paralelo que ha realizado otra compañera.

7.1. Conocimientos y experiencia adquiridos

Pese a que la parte fundamental del proyecto es comprobar que el proyectante ha adquirido durante la carrera los conocimientos suficientes como para desarrollar un proyecto por sí mismo también ha tenido que utilizar nuevas tecnologías, librerías y, como ya se ha descrito en el capítulo referido a las pruebas, ha participado por primera vez en todas las fases que forman un proyecto dentro de una empresa.

Por todo esto podemos hablar que no solo se han puesto a prueba sus conocimientos sino que el proyectante ha comprobado de primera mano los cambios que puede sufrir un proyecto. Concretamente la empresa para la que realizaba el proyecto tenía un servicio de almacenamiento en la nube, que debido a decisiones de la propia empresa este servicio paso a Etherios. Esto obligó al proyectante a borrar cualquier referencia, tanto en la memoria como en el proyecto, del anterior servicio a petición de la empresa y como es lógico a aplicar los cambios necesarios para garantizar la continuidad en el funcionamiento de la aplicación web.

Ha obligado a adquirir nuevos conocimientos como son el aprender nuevas tecnologías que no se habían estudiado durante la carrera como puede ser la comunicación con una nube de dispositivos o el despliegue y mantenimiento de una aplicación web en Google App Engine, que más adelante en otras asignaturas de la carrera usaría por su cuenta debido a los buenos resultados obtenidos en la realización de este proyecto. Y se han ampliado los conocimientos que ya se poseían como el manejo de JavaScript, sus librerías y sus APIs.

El proyectante ha experimentado el funcionamiento interno propio de una empresa de ingeniería y ha aprendido a tratar con un cliente, que en este caso es la propia empresa con la que realizaba el proyecto, siendo esta experiencia, en opinión del proyectante,

una de las experiencias más útiles de cara al futuro, preparando al proyectante para futuras experiencias similares.

Por lo tanto no solo se ha tratado de demostrar los conocimientos que poseía el proyectante sino que ha sido una experiencia que ha continuado de una forma distinta con la formación del proyectante.

7.2. Posibles mejoras

Debido al tiempo de 300 horas que se disponía para realizar el proyecto el proyectante no ha podido realizar más que lo que sería una parte inicial, completamente funcional, de un proyecto que sin duda tiene unas grandes posibilidades de ampliación. Por lo tanto se sugerirán algunas de ellas y de qué manera las hubiese afrontado el proyectante de haber dispuesto de más tiempo para la realización del proyecto:

- Encriptación con el algoritmo SHA-1 de los datos más vulnerables de la base de datos.
- Mejoras en la seguridad general de la página web con sus respectivas pruebas.
- Ampliación tanto del portal web del médico como del portal web del paciente para añadir un servicio de comunicación bilateral médico-paciente. Este servicio de comunicación podría emular una bandeja de correo interna para la aplicación con la capacidad de que los pacientes redirigiesen esos mensajes a sus correos personales.
- Ofrecer la posibilidad de cada paciente tenga más de un médico asignado e incluso que esos médicos estén divididos por su especialidad.
- Ampliar la lista de pruebas que ofrece la aplicación.

Bibliografía

[1] Dan Sanderson, *Porgramming Google App Engine*.
O'Reilly Media, Inc, USA.

[2] David Sawyer McFarland, *JavaScript & jQuery: The Missing Manual*.
Pogue Press.

[3] Robin Nixon, *Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites*.
O'Reilly Media, Inc., USA.

Anexo A

Casos de uso

En este anexo se adjuntan los casos de uso que se realizaron en la fase de análisis y que o se incluyeron en la memoria.

Cuentas de los pacientes

Caso de uso	**Crear cuenta**
Descripción	El paciente creara una cuenta personal que le permitirá acceder al sistema para consultar sus datos siempre que lo desee.
Actores	Cuenta.
Precondiciones	
Pasos	1. El paciente completa un formulario de registro. 2. El sistema comprueba que los datos son correctos y crea la cuenta.
Variaciones	

Caso de uso	**Cambiar de usuario**
Descripción	El usuario abandonará la sesión que tiene iniciada hasta el momento e iniciara sesión con un usuario distinto.
Actores	Usuario.
Precondiciones	El usuario está autenticado en el sistema.
Pasos	1. El usuario cierra la sesión que tenía iniciada. 2. El sistema borra los datos que permiten mantener sesión al usuario. 3. El usuario accede al sistema con otro usuario. 4. El usuario accede al sistema con otro usuario.
Variaciones	

Caso de uso	Editar datos personales
Descripción	El usuario modificará sus datos personales a excepción del correo electrónico que no podrá ser editado.
Actores	Usuario.
Precondiciones	El usuario está autenticado en el sistema.
Pasos	1. El usuario accederá al formulario encargado de recoger los nuevos datos. 2. El usuario cambiara los datos que desee modificar manteniendo los que no deban sufrir modificaciones. 3. El sistema guarda los cambios realizados.
Variaciones	

Caso de uso	Seleccionar dispositivo
Descripción	El usuario vinculara un dispositivo Android a su usuario para que este reciba automáticamente las notificaciones.
Actores	Usuario.
Precondiciones	El usuario está autenticado en el sistema.
Pasos	1. El usuario selecciona un dispositivo. 2. El sistema lo guarda como el dispositivo encargado de recibir las notificaciones que se envíen desde la aplicación web.
Variaciones	

Cuentas de los médicos

Caso de uso	**Cambiar contraseña**
Descripción	El médico podrá cambiar la contraseña que tiene asignada para acceder a su cuenta.
Actores	Médico.
Precondiciones	El médico está autenticado en el sistema.
Pasos	1. El médico introduce la contraseña antigua y la nueva. 2 El sistema comprueba que la contraseña vieja coincide y que las nuevas introducidas son la misma. 3. El sistema guarda los cambios si todos los datos son correctos o muestra un mensaje de error si algún valor no coincide.
Variaciones	

Cuentas de los administradores

Caso de uso	**Visualizar cuentas disponibles**
Descripción	El administrador accede a la pantalla que muestra en tiempo real y sin necesidad de actualizar cuantas cuentas hay disponibles.
Actores	Administrador.
Precondiciones	El administrador está autenticado en el sistema.
Pasos	1. El administrador accede a la pantalla donde puede ver cuántas cuentas hay disponibles o introducir nuevas cuentas. 2. El sistema muestra en tiempo real cuantas cuentas están disponibles.
Variaciones	

Caso de uso	**Añadir médico**
Descripción	El administrador por petición del centro de salud introduce los datos del nuevo médico que se quiere dar de alta en el sistema.
Actores	Administrador.
Precondiciones	El administrador está autenticado en el sistema.
Pasos	1. El administrador rellena los campos requeridos para el registro. 2. El sistema comprueba que todos los datos son correctos, si lo son crea al nuevo médico y si no lo son muestra un mensaje por pantalla.
Variaciones	

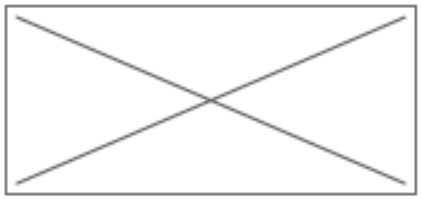
Anexo B

Mockups

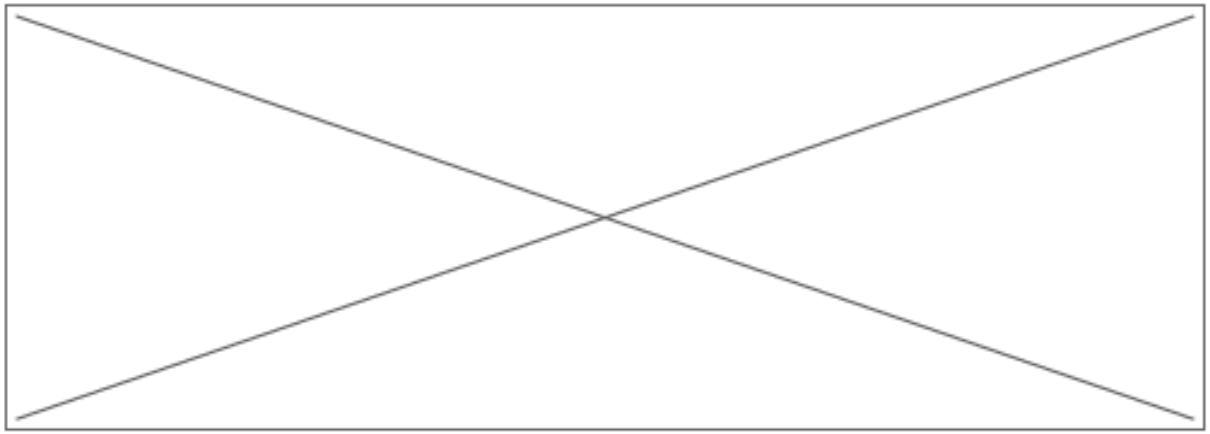
En este anexo se adjuntan los mockups realizados en la fase de diseño de la aplicación.

Telehealth

http://telehealth.appspot.com



One | Two | Three | Four



software statistics teaching
technology tips tool tools
toread travel tutorial tutorials
tv twitter typography ubuntu
usability video videos visualization

Button

Are you a new user?

Telehealth

http://telehealth.appspot.com

Name Surname

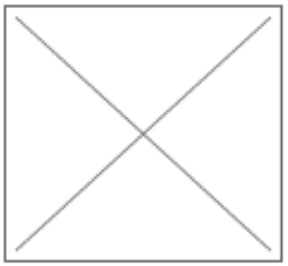
MY ACCOUNT

Account

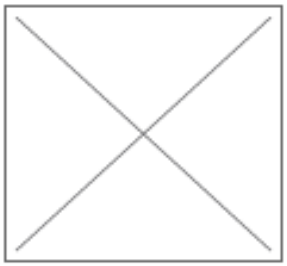
Devices

Calendar

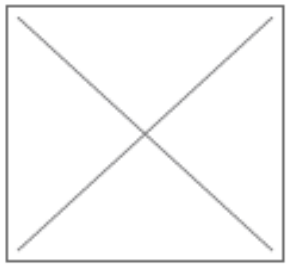
MY HEALTH



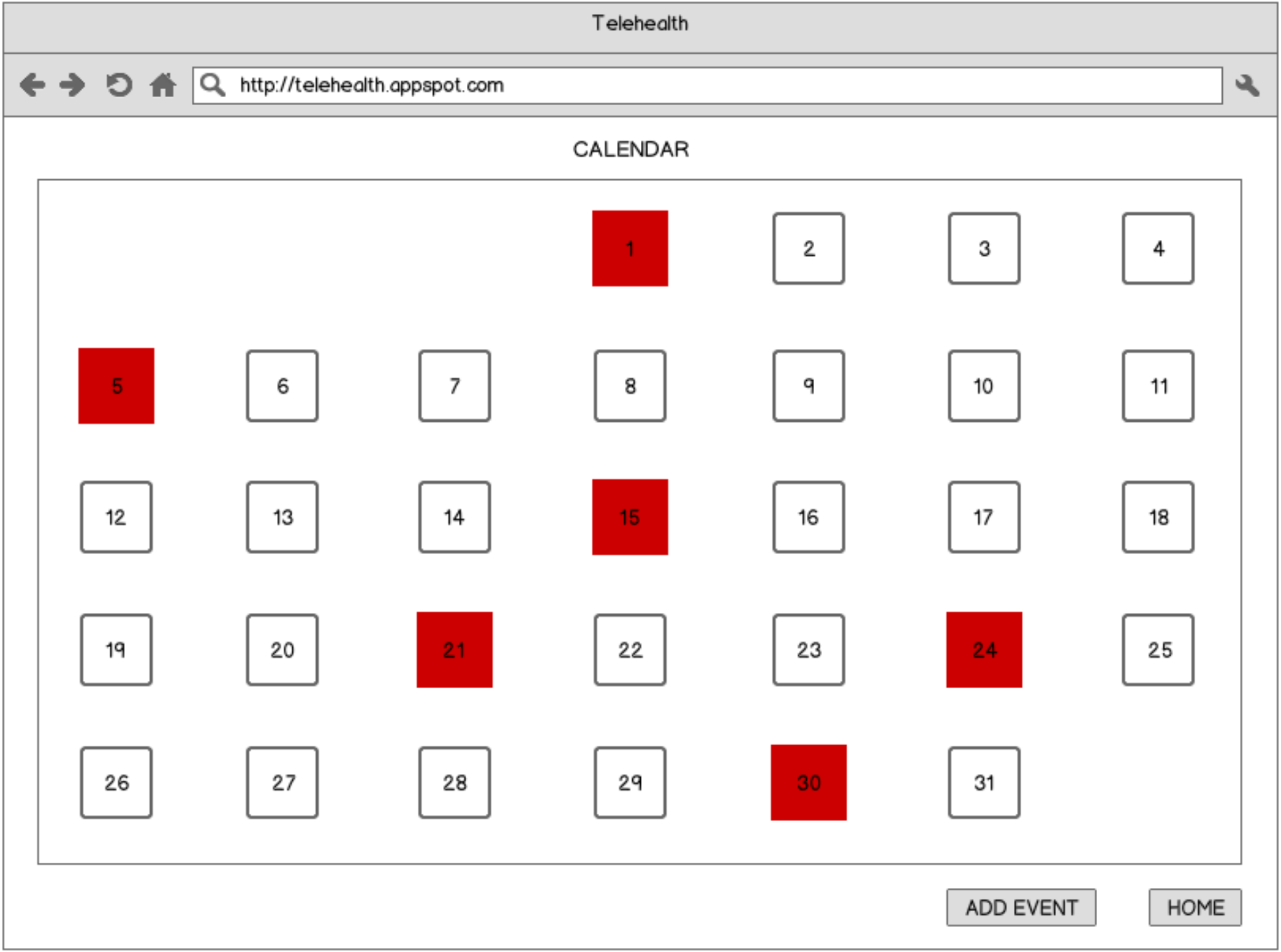
Weight

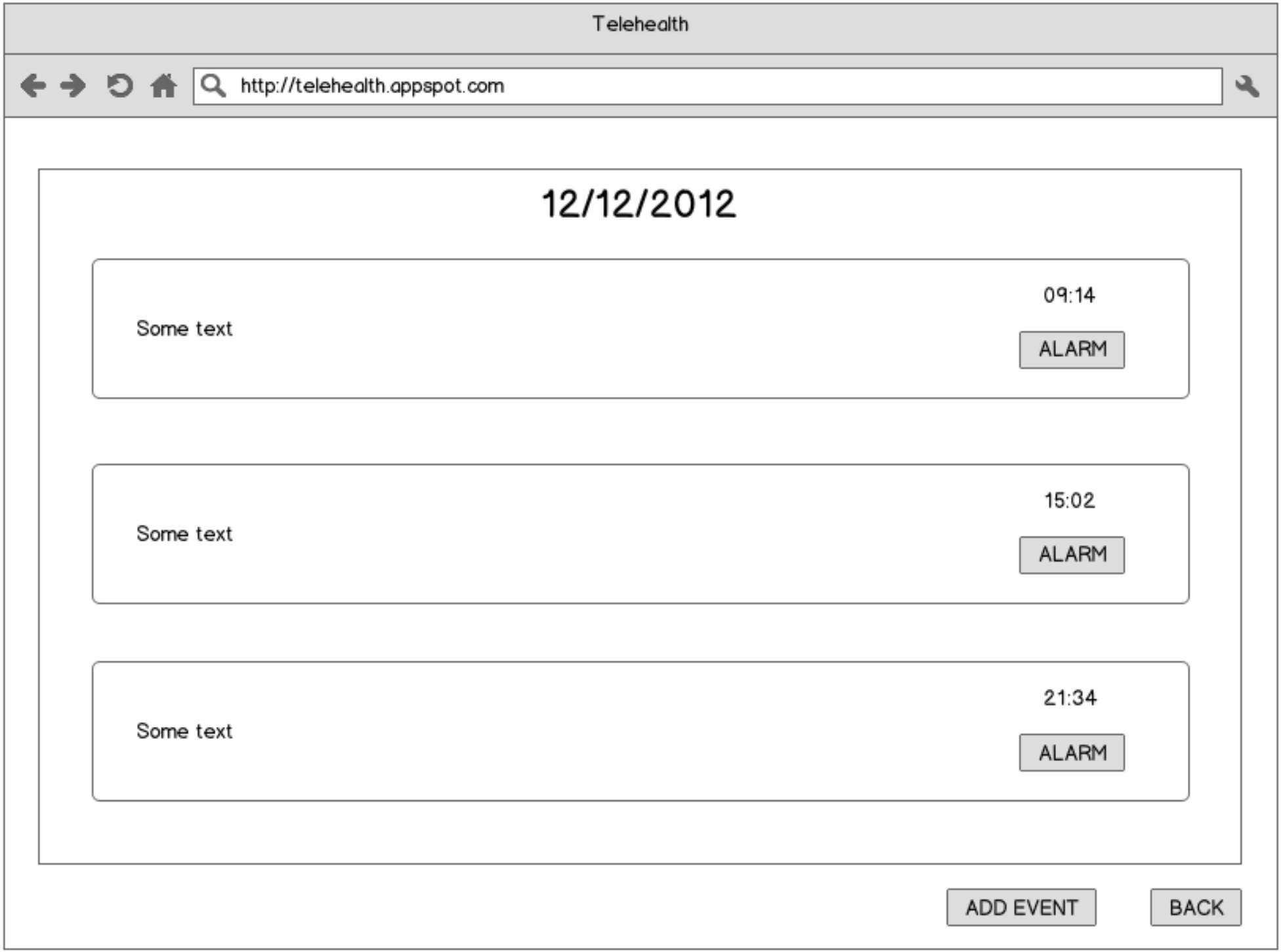


Blood pressure



Oximeter

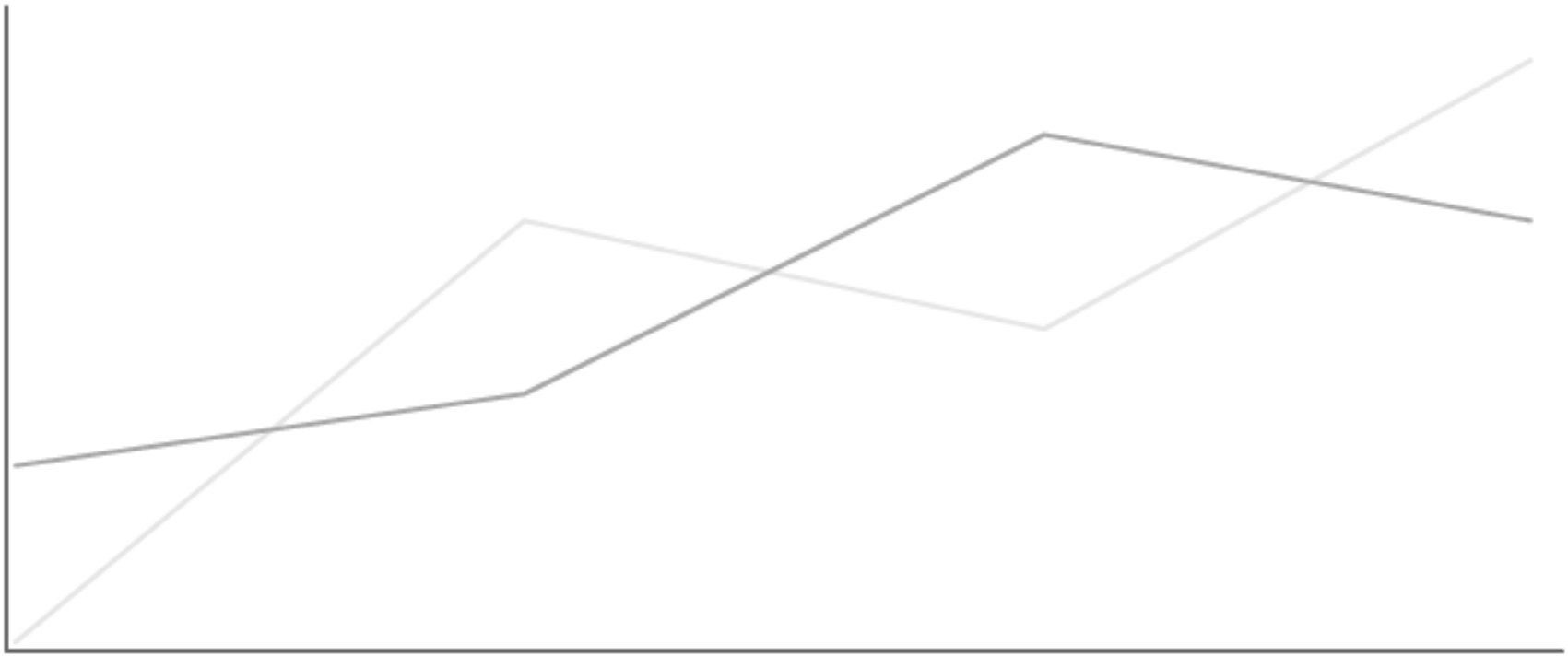

Telehealth
http://telehealth.appspot.com
CALENDAR
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
ADD EVENT
HOME


Telehealth
http://telehealth.appspot.com
12/12/2012
Some text
09:14
ALARM
Some text
15:02
ALARM
Some text
21:34
ALARM
ADD EVENT
BACK

Telehealth

http://telehealth.appspot.com

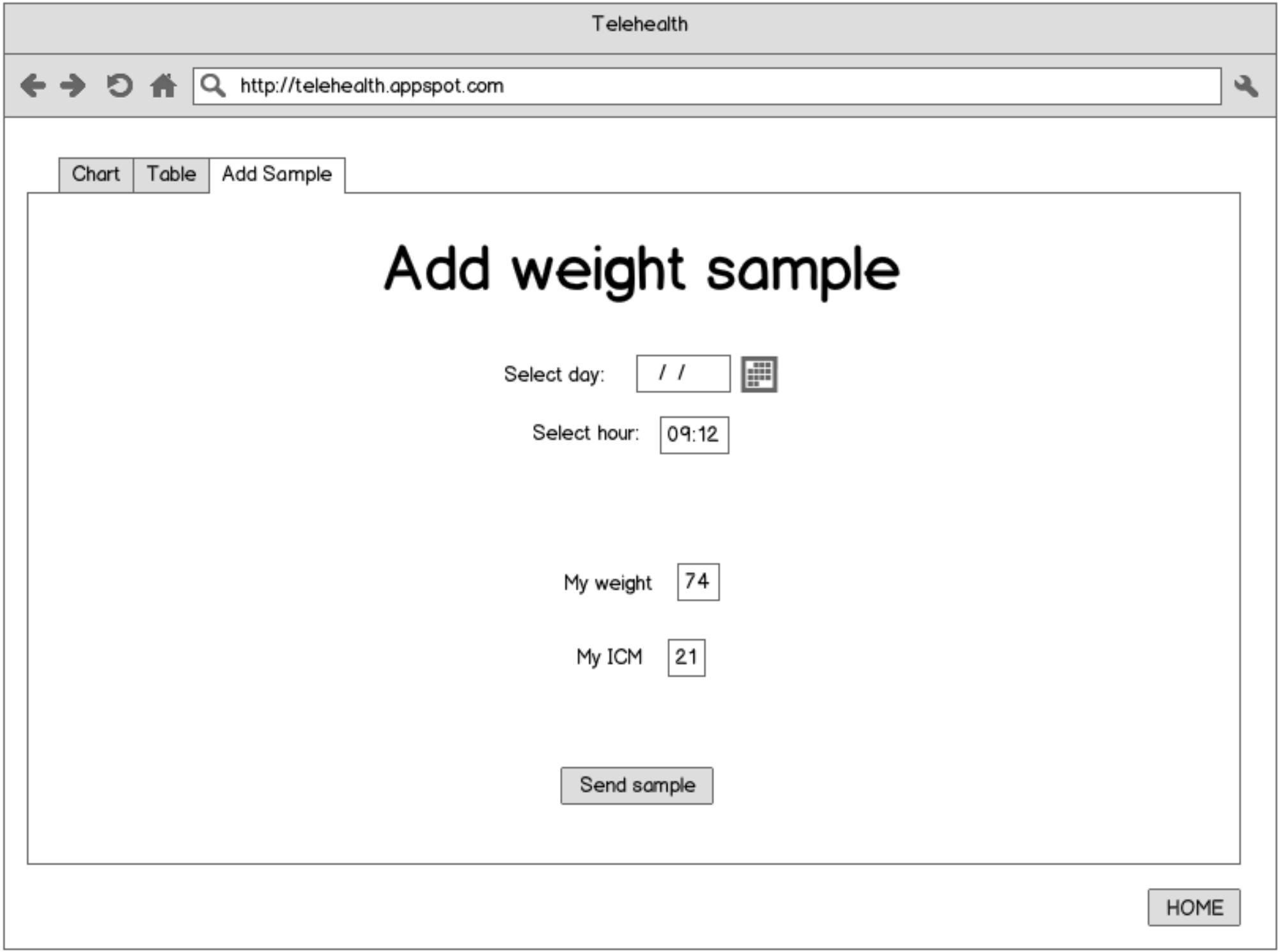
Chart | Table | Add Sample

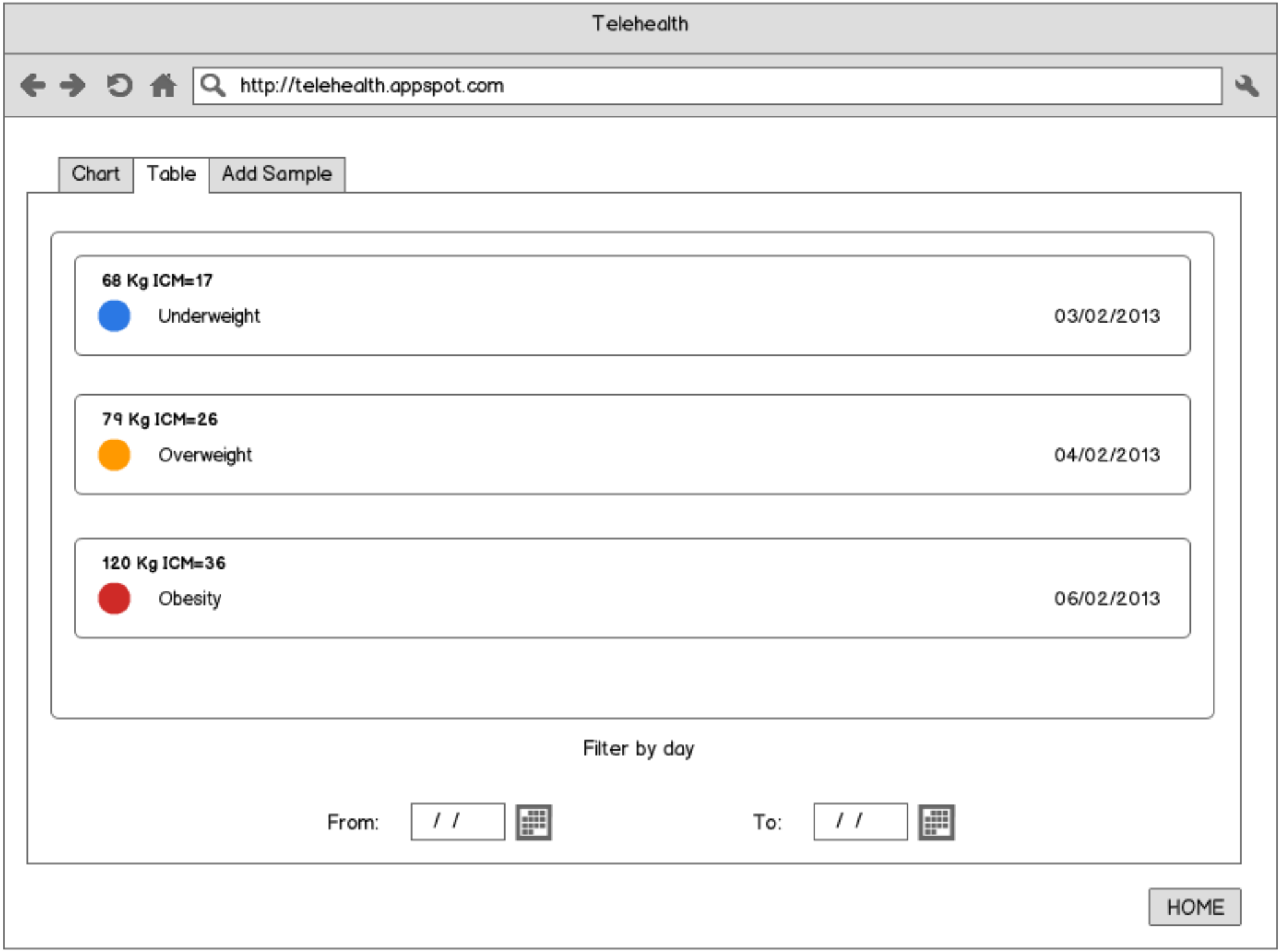


Filter by day

From: / / To: / /

HOME


Telehealth
http://telehealth.appspot.com
Chart
Table
Add Sample
Add weight sample
Select day:
/ /
Select hour:
09:12
My weight
74
My ICM
21
Send sample
HOME


Telehealth
http://telehealth.appspot.com
Chart
Table
Add Sample
68 Kg ICM=17
Underweight
03/02/2013
79 Kg ICM=26
Overweight
04/02/2013
120 Kg ICM=36
Obesity
06/02/2013
Filter by day
From:
/ /
To:
/ /
HOME