SISTEMA DE INFORMACIÓN PARA GESTIÓN DE INVENTARIO EN FARMACIA HOSPITALARIA

ANTONIO GUTIÉRREZ BLANCO

Departamento de Ingeniería de Sistemas y Automática

Ingeniería en Telecomunicaciones.

Curso 2014-2015

Autor: Antonio Gutiérrez Blanco

Tutores: José María Maestre Torreblanca y Amalia Luque Sendra

Departamento de Ingeniería de Sistemas y Automática

"Quien sólo busca el aplauso de los demás, pone su felicidad en manos ajenas"
Oliver Goldsmith

ÍNDICE

1. RESUMEN DEL PROYECTO

Este proyecto consiste en la realización de una plataforma web que gestione y optimice el stock de medicamentos en varios hospitales de Córdoba. Dicha plataforma servirá de interfaz entre el personal hospitalario y los sistemas de gestión de los medicamentos.

En particular, se pretende disminuir la incertidumbre existente en la demanda esperada de medicamentos y en la cantidad de medicamentos almacenados cuando no existen sistemas automáticos de almacenamiento con objeto de mejorar la eficiencia de los servicios de farmacia de los hospitales.

El logro de los resultados esperados supondría una clara mejora en el funcionamiento de los servicios de farmacia, dado que trabajarían con niveles menores de incertidumbre en dos aspectos claves de la gestión. Asimismo, es de esperar que el stock medio de los medicamentos almacenados pueda ser disminuído, con el consiguiente ahorro económico para los hospitales, que veran reducida las millonarias cantidades inmovilizadas en dichos stocks.

La plataforma será desarrollada haciendo uso de los diferentes lenguajes de programación web disponibles, como son PHP, HTML, JavaScript (JQuery), SQL (MySQL).

A través de dicha plataforma web, se ejecutarán los algoritmos de cálculo de la optimización. Es decir, dicha plataforma recoge los datos ofrecidos por el personal hospitalario y los sirve al sistema de cálculo del stock. Una vez calculados los datos de optimización, se presentarán como resultado a través de la misma web.

En la Figura 1.1 se puede observar el esquema básico de una aplicación web como la desarrollada en el proyecto recogido en este documento.

Se pueden distinguir por tanto tres partes fundamentales que son: la interfaz gráfica, el web service (o acceso a los datos) y la base de datos. La información se

almacena en la base de datos y, a través de la lógica de acceso, se toman dichos datos para que una vez hayan sido procesados, se presenten al usuario en una interfaz gráfica previamente diseñada (*Front-End*).

Figura 1. Esquema básico de una aplicación web.

2. RESUMEN DE LAS TECNOLOGÍAS UTILIZADAS

A día de hoy, existen muchos lenguajes de programación web que nos permiten crear resultados muy profesionales. Aunque algunos están empezando a caer en desuso frente a otros nuevos (por ejemplo, el caso de PHP frente a Python), conviene nombrarlos ya que sin duda han marcado un antes y un después en el desarrollo web.

2.1 HTML

Hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

Sin embargo, a lo largo de sus diferentes versiones, se han incorporado y

suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, smartphones, tablets, etc.). Sin embargo, para interpretar correctamente una nueva versión de HTML, los desarrolladores de navegadores web deben incorporar estos cambios y el usuario debe ser capaz de usar la nueva versión del navegador con los cambios incorporados. Normalmente los cambios son aplicados mediante parches de actualización automática (Firefox, Chrome) u ofreciendo una nueva versión del navegador con todos los cambios incorporados, en un sitio web de descarga oficial (Internet Explorer). Un navegador no actualizado no será capaz de interpretar correctamente una página web escrita en una versión de HTML superior a la que pueda interpretar, lo que obliga muchas veces a los desarrolladores a aplicar técnicas y cambios que permitan corregir problemas de visualización e incluso de interpretación de código HTML. Así mismo, las páginas escritas en una versión anterior de HTML deberían ser actualizadas o reescritas, lo que no siempre se cumple. Es por ello que ciertos navegadores aún mantienen la capacidad de interpretar páginas web de versiones HTML anteriores. Por estas razones, aún existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.

2.2 CSS

Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML). Al igual que en el caso de HTML, el World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada

etiqueta particular mediante el atributo «style».[1]

2.3 PHP

Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico para optar por el mismo como tecnología de servidor. Hoy en día sin embargo, empieza a caer en desuso frente a nuevos lenguajes como Python.

2.4 SQL

Es un sistema de gestión de bases de datos relacional. Una base de datos es una colección estructurada de datos. Mediante MySQL podemos acceder, crear y editar la información almacenada en las bases de datos. Es hoy en día uno de los más importantes y utilizados en el ámbito de la programación web.

Existen muchas herramientas que facilitan el manejo de la información almacenada en bases de datos. En el caso de este proyecto se ha utilizado

[1] Existen multitud de herramientas que permiten realizar el montaje de la web abstrayéndose del código HTML y CSS. Como uno de los componentes más importantes del proyecto es la finalidad didáctica, en este caso se ha escrito todo el código línea a línea, sin usar ninguna otra herramienta más que un simple editor de textos y un buen manual de programación.

Phpmyadmin.

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL.

2.5 JavaScript

JavaScript es un lenguaje de programación interpretado.Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model [2].

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

[2] El ***Document Object Model*** o **DOM** ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

Figura 2. Lenguajes de programación de una aplicación web.

2.6 JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

2.7 AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la

interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

2.8 Google Charts

Google charts es una herramienta muy completa diseñada por Google que proporciona una manera fácil y potente de represantar datos en una aplicación web.

Los diferentes tipos de gráficas se presentan como clases JavaScript y se suelen inyectar en bloques HTML habilitados para éste propósito. Las gráficas se renderizan utilizando tecnología HTML5/SVG, con diseños responsive muy cuidados que facilitan la visualización no sólo en grandes pantallas, sino también en dispositivos móviles y tablets.

2.9 File Transfer Protocol (FTP)

Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos.

El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos.

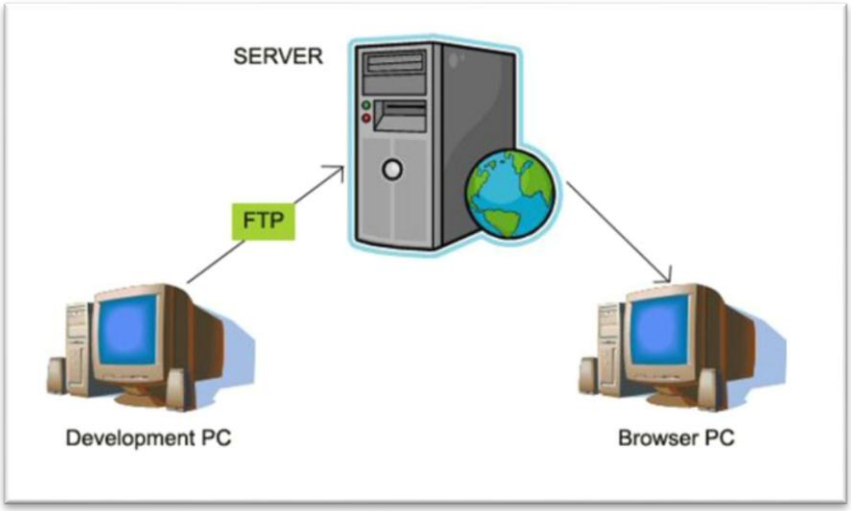


Figura 3. Esquema FTP.

3. MÉTODOS PARA LA ESTIMACIÓN DE LA DEMANDA

3.1 INTRODUCCIÓN: OPTIMIZACIÓN DE ESTIMACIÓN DE LA DEMANDA EN SERVICIO FARMACÉUTICO HOSPITALARIO

Predecir el futuro es algo que resulta sumamente interesante, y siempre ha tenido un atractivo especial para los seres humanos estando vinculado a su existencia, permaneciendo aún en nuestros días como una actividad cotidiana.

Son muchas las frases recogidas en la bibliografía sobre la contradicción entre exactitud y necesidad presente en las técnicas de pronóstico, por ejemplo: “los pronósticos siempre están equivocados”, “lo único exacto de una previsión es que no será exacta al 100%”, “precisión de una previsión resulta un término ambiguo y difícil de definir”, “prever el futuro sobre la base del pasado es como conducir un auto en una carretera de muchas curvas con todos los cristales delanteros y laterales cubiertos y solo mirando hacia atrás”, “los pronósticos jamás son perfectos y serán menos confiables mientras mayor sea el lapso que se pronostique hacia el futuro”.

Sin embargo, su estudio es una necesidad imperiosa y a pesar de las contradicciones enunciadas son innumerables los textos que desde hace años estudian los pronósticos, e incontables las empresas que en su gestión lo emplean atendiendo a su gran importancia para la planificación de la producción, la determinación del nivel óptimo de inventarios, y su influencia en la eficiencia de otras muchas funciones de la gestión de operaciones (Pérez et al. 2007).

3.2 PREDICCIÓN. SU IMPORTANCIA EN LA GESTIÓN DE OPERACIONES

En la gestión de operaciones se adopta una definición específica del pronóstico y se distingue del concepto más amplio de predicción:

Se define *pronóstico,* como el proceso de estimación de un suceso futuro proyectando datos del pasado, los cuales se combinan de forma predeterminada para hacer una estimación del futuro. Mientras que, *predicción,* se define como el proceso de estimación de un acontecimiento futuro basándose en consideraciones subjetivas diferentes a los simples datos provenientes del pasado.

De lo anterior se puede concluir que la predicción posee un carácter de arte y ciencia, pues está matizada por la experiencia y conocimientos de la persona que predice, quien será capaz de modificar los "datos" obtenidos del pronóstico, apoyándose tanto en técnicas cuantitativas como cualitativas para la toma de decisiones.

La previsión es una herramienta importante de la planeación estratégica y operacional, resulta un paso previo a la toma de decisiones, y por lo tanto, desviaciones substanciales tienen consecuencias económicas importantes (Pérez et al. 2007).

La planificación de operaciones se puede dividir en dos áreas principales: *previsión y planificación*. Generalmente en una economía competitiva, son necesarias previsiones de la demanda, siendo el objetivo de la planificación, convertir la salida proyectada por la previsión, en exigencias de entradas de materias primas, personal, equipos, etc.

3.3 TÉCNICAS DE PRONÓSTICO

Las técnicas empleadas en la realización de pronósticos varían en función del contexto objeto de la previsión. En principio, las técnicas pueden clasificarse en dos categorías:

Técnicas cualitativas, se basan fundamentalmente en el conocimiento humano y efectúan las estimaciones futuras a partir de informaciones cualitativas, tales como, opiniones de uno o más expertos, analogías, comparaciones, etc. Son conocidas como técnicas subjetivas, y en ellas, la distinción entre pronóstico y previsión no es tan acusada. Son los únicos métodos utilizables cuando no existen suficientes datos históricos.

Técnicas cuantitativas, se basan en modelos matemáticos, principalmente de tipo estadístico, los cuales han de ser alimentados por abundante información histórica sobre las variables que tienen influencia en el proceso, y sobre ellas, se genera la previsión. No existiendo factores subjetivos de ningún tipo.

¿DATOS HISTÓRICOS?	NO	SI
TÉCNICA DE PRONÓSTICO	cualiativa	cuantitativa

Tabla 1. Diferentes tipos de pronóstico.

En el mundo empresarial, se toman decisiones en diferentes horizontes temporales, de ahí que, las técnicas cuantitativas se recomiendan siempre que se trate de un horizonte a corto/medio plazo, mientras que, las técnicas cualitativas dependen básicamente de la opinión de los expertos y se utilizan cuando no existen datos históricos, o no se consideran útiles para predecir el futuro, por ello se utilizan para realizar previsiones a medio y, sobre todo, largo plazo.

A continuación, se presentan los diferentes tipos de técnicas cuantitativas, (técnicas objeto de estudio), siendo estas realmente efectivas, sólo si, el sistema ha alcanzado cierto nivel de estabilidad (Pérez et al. 2007):

Series temporales, partiendo de un conjunto ordenado de observaciones, recogidas durante intervalos de tiempo idénticos, que indican la evolución temporal de la variable objeto de estudio, se trata de extrapolar ese comportamiento hacia el futuro.

Modelos causales, en este caso, el tiempo no es la variable independiente base para la recogida de la información, sino que se suponen establecidas unas relaciones determinadas entre algunas de las variables que intervienen, y se trata de determinar cuáles son exactamente esas relaciones, siendo la forma más común de encontrarlas mediante ecuaciones de regresión.

En este proyecto nos limitaremos al caso de predecir valores a partir de datos históricos recogidos. Es decir, presentaremos dos métodos diferentes de pronóstico para la demanda, siendo los dos series temporales. Ambos se explican en los puntos siguientes.

3.3.1 PROMEDIO MÓVIL SIMPLE

Este método de pronóstico, utiliza solamente los últimos "n" periodos y la siguiente expresión:

$$\tilde{Y}_{n+1} = \sum_{i=t-n+1}^{t} (Y_i)/n$$

Esta técnica no solo usa todos los datos relevantes de los últimos "n" periodos, sino que además se actualiza fácilmente de un periodo a otro, eliminando la primeraobservación y agregando la última.

Una ventaja de este procedimiento es que los cálculos son muy sencillos y, sinembargo, el sistema es flexible, en el sentido de que la tendencia no es forzada a adaptarse a ninguna función matemática en particular.

3.3.2 ALISADO EXPONENCIAL

Se basa en la idea de que es posible calcular un pronóstico a partir de un promedio anterior, la demanda más reciente y una constante de suavizamiento α. Supóngase, por ejemplo, que se tiene un promedio anterior de valor “a” y que se acaba de observar una demanda de valor “b”, parece razonable que el nuevo promedio esté entre “a” y “b”, dependiendo de qué peso se quiera dar a la demanda que acaba de observarse y qué peso al promedio anterior.

Resulta válida la premisa de que la importancia de los datos disminuye mientras más antiguos sean, de modo que tengan más peso las observaciones recientes que las antiguas. La forma más cómoda de definir unos pesos de este tipo corresponde a utilizar una progresión geométrica de razón menor que la unidad.

Mientras que el método de las medias móviles solo tiene en cuenta un número "t" de valores de y_i, al suponer que los y_i más lejanos en el tiempo ya no tienen influencia, este método considera todos los y_i pasados, aunque a los muy lejanos se les puede dar tan poco peso que prácticamente no ejerzan ninguna influencia.

Otra característica es que este método es capaz de realizar una media móvil ponderada de los n valores de y_i, sin necesidad de almacenar todos esos valores.

Se prescinde del parámetro "t”, pero se añade el valor ponderado $\hat{Y}_1$, que representa la estimación para el primer periodo histórico, recomendándose usar $\hat{Y}_1 = Y_1$ para empezar. Además, se incorpora la constante de suavizamiento α perteneciente al intervalo $(0, 1)$ mediante la cual se consigue que los valores lejanos vayan teniendo cada vez menos influencia, restando así importancia a la correcta elección de $\hat{Y}_1$. Para ello, se definirán n pesos α_i que se obtendrán a partir del parámetro α según una progresión geométrica de razón $q = 1 - \alpha$, con $\alpha_i = \alpha$.

Se deduce que la expresión para estimar el periodo $n+1$ será:

$$\hat{Y}_{n+1} = \alpha \cdot Y_n + (1-\alpha) \cdot \hat{Y}_n\,, \qquad n \geq 1$$

estimación que se realiza en función del valor real obtenido en el periodo anterior, y de la estimación en ese periodo, aunque implícitamente se están considerando los n valores de Y.

Si se eligen valores grandes para α, se está dando más peso a los valores reales recientes, mientras que, dando valores pequeños a este parámetro, los pesos se reparten en mayor medida a lo largo de la serie, por tanto, el uso de los valores grandes se reserva para cuando se prevén cambios sustanciales en el comportamiento de la variable que se estudia, de forma que el modelo responda con mayor rapidez a los cambios de la demanda.

La elección de α debe realizarse sobre la base de prueba/error, seleccionándose un valor mínimo y otro máximo de α, así como, un valor de paso, calculándose el mejor resultado para ese intervalo, sobre la base de la medida de error fijada.

La utilización de este método se recomienda cuando el horizonte temporal de previsión es corto, no se conoce información acerca de ningún factor independiente que influya sobre la demanda, se desea invertir pocos recursos y esfuerzos en la tarea de previsión, y se desea que la previsión pueda seguir las tendencias y estacionalidades (Pérez et al. 2007).

4. ARQUITECTURA DE LA APLICACIÓN DESARROLLADA

El propósito de las siguientes páginas es tratar de describir de la manera más clara posible, cuáles son los diferentes módulos que componen la aplicación aquí desarrollada. Se ahondará en aspectos técnicos y se proporcionarán aquellos fragmentos de código que se estimen convenientes para mejorar la comprensión del funcionamiento de la aplicación por parte del lector.

En líneas generales, podríamos decir que la aplicación está compuesta por los módulos numerados a continuación:

1. Acceso de usuarios.
2. Menú principal.
3. Histórico de consumo.
4. Panel de control.
5. Subida de ficheros.
6. Cálculo del pedido óptimo.

Cada uno de los apartados citados arriba y algunos más se explicarán más en profuncidad a lo largo de este capítulo, insertando ejemplos del código desarrollado para que aquellas partes que sean más prácticas puedan entenderse mejor.

Finalmente, se adjunta un manual de usuario con capturas propias de la aplicación que bien podría ser distribuído al personal sanitario para comprender el funcionamiento de la plataforma.

4.1 ACCESO DE USUARIOS

Cuando un usuario intenta acceder al sistema, se le pide que introduzca unos credenciales de acceso que serán contrastados con la información almacenada en base de datos, la cual ha sido introducida previamente en base de datos por un administrador de la plataforma.

Este formulario llama al fichero *comprueba.php* , que compara los datos introducidos por el usuario (nombre de usuario y contraseña), con los almacenados en base de datos. Si ambos coinciden, el acceso se realiza con éxito y la petición AJAX devuelve el valor "OK" y a continuación se redirige a *form.php*. Si el acceso ha fallado, el valor devuelto será "NOK" y se mostrará un mensaje de error para alertarnos de que algo ha fallado y así el usuario vuelva a introducir sus datos de acceso.

Como esta plataforma no es ni mucho menos definitiva, sino más bien una herramienta provisional para unir hospitales y universidad, no se ha estimado conveniente el uso de algún mecanismo de encriptación o cifrado de datos para aumentar la seguridad. En el caso de que fuese necesario, se recomendaría usar un método de encriptación asimétrica, donde mediante un par de claves pública y privada, se encripte y desencripte aquellos datos que puedan ser susceptibles a ser robados en un ataque al sistema.

Mediante JavaScript, no permitiremos que el usuario trate de iniciar sesión dejando alguno de los campos del formulario vacíos. Una vez se hace click en el botón de "Entrar", comprobamos que ninguno de los campos vaya vacío. Si alguno de ellos está vacío, el script de JavaScript mostrará un mensaje de alerta y devolverá el valor *false*, con lo cual no se hará la petición *Ajax*.

Para este proceso se ha hecho uso de la tecnología *Ajax* descrita al inicio de este documento. Ésta nos permite ejecutar código en segundo plano en el lado del servidor sin necesidad de recargar toda la página, con lo cual todo el proceso pasa desapercibido a los ojos del usuario.

Si el acceso se ha realizado con éxito, se redirige a *form.php* , donde se crea una

sesión para el usuario. Para ello se hace uso del Código 1.

```
session_start();
$_SESSION["usuario"] = $user;
$_SESSION["permisos"] = $result['permisos'];
```

Código 1. Inicio de sesión en php

En la sesión se almacena el nombre de usuario y los permisos de éste. Si el usuario es ***root***, un panel de control aparecerá en la cabecera, mediante el cual el usuario gozará de privilegios como insertar nuevos usuarios al sistema, ver todo el registro de medicamentos, etc.

Si por el contrario el usuario pertenece a alguno de los hospitales o a la Universidad de Sevilla, su uso de la web se verá restringido a lo datos que pertenezcan a su hospital o universidad, es decir, un usuario del Hospital Reina Sofía de Córdoba no podrá ver el historial de consumo del San Juan de Dios y viceversa. De esta forma, se intenta salvaguardar la privacidad de los datos entre diferentes hospitales.

Las cabeceras de *http* no almacenan variables propias de usuario, es decir, el servidor no tiene idea alguna de quién está usando la aplicación.

Para resolver este problema y que se pueda almacenar información relacionada con el usuario, PHP incorpora las denominadas variables de sesión, que por lo general son variables globales que se almacenan hasta que el usuario cierra el navegador.

Iniciar sesión en php:

Para abrir un nueva sesión en PHP, se hace uso de la función nativa *session_start()*. En nuestro código, dicha función se llama en el fichero header.php, ya que éste se incluye en todo los demás ficheros y por lo tanto nos

permitirá acceder a las variables de sesión en cualquier momento.

Acceso a las variables de sesión:

Para acceder a las variables de sesión, hacemos uso de la nomenclatura *$_SESSION.* Dicha variable es de tipo diccionario y en él se almacenan las variables de sesión registradas por el usuario y otros parámetros de control.

Como es habitual, podemos hacer uso de dichas variables para almacenar y leer datos en ella.

Cerrar sesión:

Antes de cerrar sesión, es recomendable borrar todas las variables de sesión, para ello hacemos uso de *session_unset()*. Así nos aseguramos que todas han sido borradas. Por último, cerramos la sesión propiamente dicha. Para ello basta con llamar a la función *session_destroy()*;

La manera habitual de proceder es en ese orden. Primero borramos las variables de sesión y finalmente cerramos la sesión.

Para evitar que un usuario que no haya iniciado sesión pueda acceder a un fichero cualquiera introduciendo la dirección directamente en la barra de direcciones, al inicio del header.php se han incluído unas líneas que comprueban que si la variable de sesión va vacía, se redireccione al usuario a *index.php*

```
session_start();

if(empty($_SESSION) && $title != "Acceso al portal"){
  header('Location: /');
  die();
}
```

Código 2. Redirección si no se ha iniciado sesión.

En el Código 2 vemos que una vez hecha la redirección, se ha incluido la línea *die*(). Es muy importante poner esta función después de cada redirección, ya que así controlamos que la ejecución de dicho fichero se detenga una vez hemos redireccionado.

También es importante comprobar que no estamos en el fichero de inicio de sesión, puesto que entraríamos en un bucle infinito debido a la redirección.

4.2 MENÚ DESPLEGABLE

El framework de PHP-HTML *Bootstrap* incluye diferentes clases predefinidas que nos permiten hacer menus desplegables muy configurables y con un aspecto muy professional.

Como en el desarrollo de esta plataforma no nos hemos centrado mucho en el aspecto visual sino más bien en la funcionalidad, hemos decidido implementar un simple menu desplegable cuando se hace click en él (evento *onclick* de JavaScript).

Una vez desplegado el menú, dependiendo de los provilegios del usuario, se mostrarán una serie de opciones u otras.

Si el usuario que está utilizando la plataforma es administrador, verá disponibles todas las opciones existentes. Estas son las siguientes:

Panel de control: desde el cual podemos añadir o eliminar tanto fármacos como usuarios.

Subir fichero: desde esta sección el usuario puede subir un fichero de texto con tuplas de valores fecha-consumo, de esta manera serán incorporados dichos datos en nuestra base de datos de manera automática.

Calcular Pedido: desde aquí se lanza la función C que nos calcula el número de unidades a pedir en la fecha indicada.

Inicio: nos redirecciona a la pantalla principal de la aplicación, donde podremos

consultar el consumo de un fármaco o inserter nuevos valores de éste.

Cerrar sesión: El usuario sera redireccionado a la pantalla de inicio de sesión tras previamente haber liberado todas las variables y cerrar las conexiones existentes con base de datos.

Junto a las opciones disponibles se han insertado unos pequeños iconos que dan un aspecto más profesional y una rápida visión del cometido de cada pestaña. Estos iconos pertenecen a la librería font-awesome, la cuál se puede descargar gratuitamente desde su web official. Una vez descargada, se suben los ficheros a nuestro servidor y tras incluirlos en nuestro header podremos hacer uso de ellos con las etiquetas <*i*> y añadiendo la clase correspondiente a cada icono.

```
<link rel="stylesheet" href="/font-awesome/css/font-awesome.min.css">
```

Código 3. Ejemplo de icono fontawesome.

```
<i class="fa fa-bar-chart"></i>
```

Código 4. Insertar css para iconos de fontawesome.

La estructura del menú desplegable se muesta en el Código 5.

Antes de nada controlamos que no estemos en la pantalla de inicio de sesión, puesto que ahí no queremos mostrar el menú. Si no estamos, definimos el bloque div de clase dropdown y el botón que hará que se muestre el menú.

Dentro de dicho bloque tenemos que crear una lista que se corresponderá con las

opciones del menú una vez desplegable. Las entradas de esta lista se controlan mediante el valor de la variable **$_*SESSION*['*permisos*']** de sesión.

En función del valor de esta variable se mostrarán unas opciones u otras, como ya hemos comentado anteriormente.

```
if($title != "Acceso al portal" && !empty($_SESSION)){
    ?>
    <div id="header">
      <div class="dropdown">
        <button class="btn btn-primary dropdown-toggle" type="button" data-
toggle="dropdown">Menu de <?=ucfirst($_SESSION['usuario'])?>
        <span class="caret"></span></button>
        <ul class="dropdown-menu dropdown-menu-right">
          <?
          if($_SESSION['permisos'] == "root"){
          ?>
          <li><a href="/panel-de-control"><i class="fa fa-cogs">  Panel del Control</i></a></li>
          <li><a href="#"><i class="fa fa-cloud-upload"></i>  Subir fichero</a></li>
          <?}?>
          <li><a href="/form"><i class="fa fa-bar-chart"></i>  Inicio</a></li>
          <li><a href="#"><i class="fa fa-sign-out"></i>  Cerrar sesion</a></li>
        </ul>
      </div>
    </div>
    <?
}
```

Código 5. Menú desplegable.

4.3 PANEL DE CONTROL

Desde esta pestaña, el usuario con privilegios de administrador puede realizar diferentes tareas relacionadas con la gestión de la base de datos.

Se presentan al usuario las siguientes cuatro opciones:

Añadir usuario:

Crea un nuevo usuario para la plataforma. Habrá que especificar tres campos que son el nombre de usuario, la contraseña, y el hospital para el que trabaja. También se ha añadido el valor *root* al selector para poder otorgar permisos de administrados al nuevo usuario.

Eliminar usuario:

Elimina usuarios del sistema. Hay que especificar nombre de usuario y tipo, es decir, en que hospital trabaja o si es administrador. De esta manera nos garantizamos que puedan por ejemplo existir dos usuarios con el mismo nombre de usuario siempre y cuando pertenezcan a diferentes hospitales.

Añadir fármaco:

Es una opción más bien representativa. Se trata de un formulario donde se definirán los valores de diferentes constantes asociadas a cada fármaco de la aplicación. Estas son necesarias para el cálculo óptimo del pedido y varían de una fármaco a otro.

Eliminar fármaco:

Elimina de forma definitiva un determinado fármaco de la base de datos.

Una vez el usuario hace click en “Añadir/Eliminar”, se hace una petición POST al fichero admin-panel-de-control.php, donde en función del valor de $_*POST*[‘form’] se actuará de un modo u otro:

Acción del formulario	$_POST[‘form’]
Añadir Usuario	1
Eliminar usuario	2
Añadir Fármaco	3
Eliminar Fármaco	4

Tabla 2. Posibles valores de la variable “form”.

4.4 CIERRE DE SESIÓN

Como se puede ver en el Código 6, se trata de un código muy sencillo con el cual liberamos todas las variables de sesión del usuario, cerramos la sesión propiamente dicha, y volvemos a la pantalla inicial de la aplicación, la del ingreso de sesión.

El propio lenguaje PHP nos proporciona una serie de funciones nativas que nos permiten dar una sesión por cerrada.

Aunque no se hayan llamado a estas funciones propiamente desde el código, cuando el usuario cierra el navegador, o pasa un determinado tiempo sin volver a acceder a la aplicación web, la sesión de dice que ha caducado y por tanto desaparece.

Para ello, se borra automáticamente la cookie [3] donde se almacenaba la información de la sesión y ya está.

[3] Una ***cookie*** es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

```php
<?
session_start();

// Borrar variables de sesion
session_unset();

// cerrar la sesion
session_destroy();

//redireccion a index

header("Location: /");
die();
?>
```

Código 6. Cierre de sesión en php.

4.5 FORM

Es la página principal de la aplicación, en ella el personal del hospital podrá tanto consultar el consumo de un determinado fármaco en un intervalo de tiempo, como insertar el consumo de un fármaco para una fecha concreta.

Cuando el usuario selecciona un fármaco con el selector del apartado “Insertar datos”, mediante *JavaScript* detectamos el evento cambio (*onchange*) en la lista de fármacos, este llama a la función *drawChartarea*(). Ésta función, mediante una petición Ajax, obtiene el consumo de dicho fármaco en las últimas cuatro semanas.

```
$(document).ready(function() {
  $("#farmaco").change(function(){
    console.log("change");
    drawChartarea();
  });
});
```

Código 7. Detección cambio del fármaco seleccionado.

Para que la petición *Ajax* nos devuelva una tabla necesitamos hacer un *json_encode()*. El formato *JSON* está muy extendido en el mundo del desarrollo web ya que resulta ser una forma muy eficaz de manejar datos, puesto que de un simple vistazo el programador es capaz de saber los datos que obtiene. Es un formato tan usado que prácticamente la totalidad de los lenguajes web tienen funciones implementadas que permiten el manejo de datos en dicho formato.

La función $.ajax() de *JQuery* nos permite decirle qué formato traen los datos esperados. En este caso hacemos dataType: "json", y de esta manera la propia función nos devolverá el resultado de la petición AJAX parseado.

Por otro lado, el fichero *grafica.php* formará el json que posteriormente se representará en la gráfica. Este script tomará la fecha de hoy y la de hace cuatro semanas y consultará con base de datos el consumo para ese intervalo de tiempo.

Mediante la función array_push() de PHP vamos insertando tablas en un array que finalmente será convertido a *JSON* con la función json_encode. El push al array se hará una vez por cada resultado encontrado en base de datos.

```
$sql = "SELECT * FROM `registro` WHERE `nombre` LIKE '$farmaco' AND `fecha` <= '$fin'
AND `fecha` >= '$inicio' ";
//echo $sql;die();
$result = mysqli_query($conn, $sql);
//echo mysqli_affected_rows($conn);
$index = 0;
$response = array();

while($block = mysqli_fetch_assoc($result)){
  array_push($response, array("fecha" => $block['fecha'], "pedido" => intval($block['cantidad'])));
}
echo json_encode(array("status" => "OK", "data" => $response));
```

Código 8. Consumo de un fármaco en una ventana temporal.

Una vez de vuelta en la función *drawChartarea*() de form.php, insertamos en la variable data las columnas 'Fecha' y 'Consumo', y una fila por cada valor devuelto de "data" en el json. Con esto ya podemos pintar nuestra gráfica deseada.

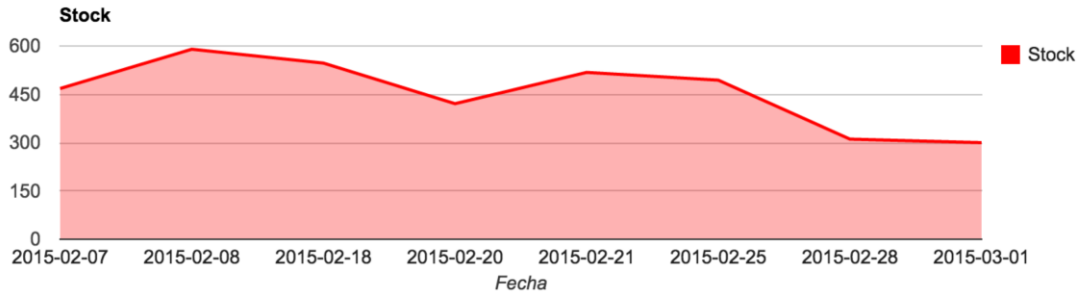


Figura 4. Consumo del fármaco seleccionado.

El formato de la tabla viene estipulado en la documentación de ***Google Charts***, que ha sido la herramienta seleccionada para la realización de gráficas en la plataforma.

Es una librería muy extensa y bien documentada, con lo cual se ha considerado que podría ser una buena herramienta para empezar a representar datos. Es realmente potente y muy personalizable.

Cuando el usuario ha introducido todos los datos requeridos y pulsa el botón enviar, el fichero insertar.php se encarga de guardar los datos introducidos por el usuario en base de datos. Este fichero se lanza mediante una petición Ajax, de forma que todo el proceso se haga en segundo plano y de manera transparente al usuario.

Una vez en el fichero insertar.php, tan solo que tenemos que lanzar la query (insertar query) contra la base de datos. Esta query, en caso de que ya exista un valor fármaco-fecha en nuestra base de datos, actualizará el consumo de éste, de modo que prevalezca siempre el último valor insertado por el usuario.

Para terminar, dicho fichero devolverá el valor que a su vez devuelva la función *mysqli_query*(). Como en este caso la query es un *INSERT*, ésta función nos devolverá o *TRUE* o *FALSE*.

El valor devuelto se capturará en el código *JavaScript* que lanzó la petición *AJAX*. En caso de que dicho valor sea *TRUE*, se mostrará un mensaje de éxito y se volverá a llamar a la función *drawChart*() para que nos actualice la gráfica correspondiente. En caso de que el insert haya salido mal por cualquier motivo, el valor devuelto será false, y lo único que haremos en este caso será mostrar el mensaje de que algo ha salido mal y no se ha podido insertar el valor en la base de datos.

Como en hemos hecho en ocasiones anteriores, para evitar que los formularios vayan vacíos, hacemos uso del método *JQuery submit*().

Este método se aplica sobre el identificador de los formularios, y detecta cuándo se hace un submit en dicho formulario. Una vez detectado el submit, tan sólo nos queda comprobar que los valores de los input no vayan vacíos. Si van vacíos, devolvemos false para que no se envíe el formulario y mostramos una alerta al usuario indicándole que no ha rellenado los campos. Si por el contrario están

completos, el formulario se enviará para que se siga el hilo de la ejecución.

El caso del formulario de registro hemos visto que es un poco diferente. En este caso siempre devolveremos el valor false para evitar que el formulario se envie contra la url indicada en action (en este caso va en blanco). En el Código 9 se puede ver con más detalle el código que valida dicho formulario.

```
$('#form-registro').submit(function(){
    var fecha = $("#insert-date").val();
    var consumo = $("#consumo").val();
    var farmaco = $("#farmaco").val();
    $('#msg-insertar').empty();
    if(fecha != "" && consumo != "") {
      $.ajax({
        method: "POST",
        url: "insertar.php",
        data: { fecha: fecha, consumo: consumo, farmaco: farmaco }
      })
        .done(function( msg ) {
          if(msg){
            $('#msg-insertar').html("Registrado con exito");
            drawChartarea();
          }
          else
            $('#msg-insertar').html("Error al registrar");
        });
      return false;
    } else {
      alert("Fecha o consumo incompletos");
      return false;
    }
  });
```

Código 9. Validación del formulario antes de la insercción.

4.6 HISTÓRICO DE CONSUMO

En la página principal de la aplicación (form.php), se presenta al usuario un formulario donde podrá seleccionar un intervalo temporal definido por dos fechas (inicio y fin) y un tipo de fármaco. Una vez se envía el formulario, se presentan diferentes gráficas donde se representa el consumo de dicho fármaco en el intervalo seleccionado y un histórico total en forma de calendario donde mostrar todos los datos de consumo disponibles en base de datos.

Figura 5. Formulario histórico de datos.

Para la elaboración de dichas gráficas se ha utilizado la librería JavaScript que gratuitamente proporciona Google y que se llama Google Charts.

Además del histórico de consumo, se han representado gráficamente los estimadores desarrollados para dicha aplicación. Para el estimador de media aritmética, simplemente se ha ido calculando la media de los valores recogidos de base de datos. Finalmente, como se puede apreciar en la Figura 6, se ha representado en una gráfica de líneas el consumo del fármaco (en color azul), y el del valor medio (en color rojo).

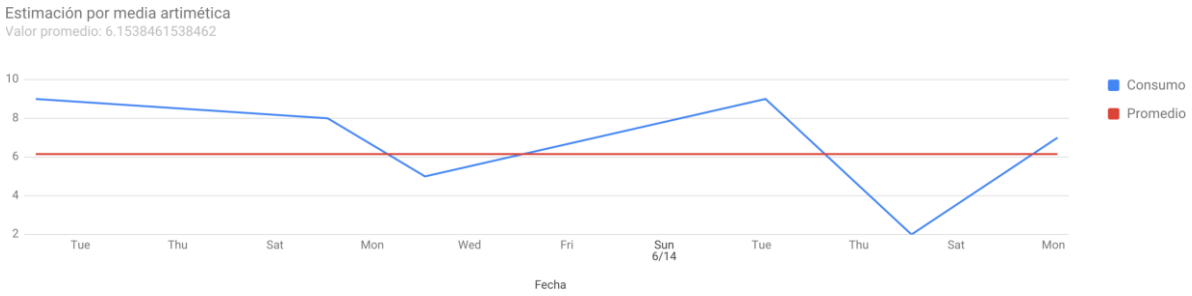


Figura 6. Estimación por media aritmética.

Para el gráfico de tipo calenderio, se ha hecho uso del objeto *date*() de *JavaScript*. La fecha recogida de base de datos, una vez ha sido parseada y pasada de php a *JavaScript*, mediante codificación *JSON,* se van insertando filas donde el primer valor del array es el objeto con la fecha respectiva, y el segundo elemento del array es el consumo del fármaco para dicha fecha.

La clase *date*() es muy útil en este caso porque nos representa de manera automática el consumo sobre el calendario, sin tener que desplazar manualmente los meses hacia el 0 para enero, el 1 para febrero, etc.

Este es el protocolo usado para la codificación de los meses en *Google charts*, y afortunadamente coincide con el del objeto *date*() de *JavaScript*.

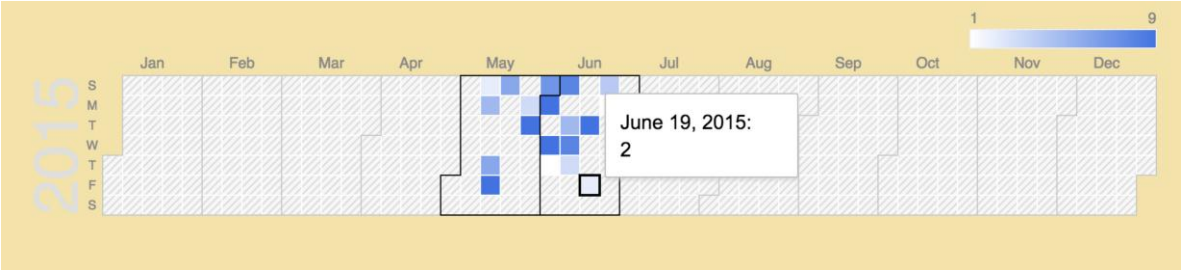


Figura 7. gráfica de tipo calendario para representar el histórico de consumo.

Fecha	Consumo
Jun 1, 2015	9
Jun 3, 2015	9
Jun 4, 2015	1
Jun 7, 2015	8
Jun 9, 2015	5
Jun 10, 2015	8
Jun 11, 2015	3
Jun 16, 2015	9
Jun 19, 2015	2
Jun 21, 2015	4
Jun 22, 2015	1

Tabla 3. Ejemplo de consumo de un fármaco.

Como en esta pantalla estamos usando diferentes tipos de gráficas (tabla, líneas, calendario, etc), hemos rellenado un array con los paquetes que queremos cargar y el cual pasaremos al método load de la clase proporcionada por Google. En este caso cargamos tres paquetes diferentes: “calendar”, “table” y “line”.

```
google.load("visualization", "1.1", {packages:["calendar", "table", "line"]});
```

Código 10. Carga de paquetes de google charts.

4.7 GENERACIÓN ALEATORIA DE CONSUMO DE FÁRMACOS

Para simular un entorno realista donde poder probar la plataforma, hemos realizado un pequeño script donde se generarán de forma aleatoria diferentes valores de consumo para diferentes fármacos y diferentes días del año. A continuación se describe el funcionamiento.

```
<?
include('header.php');

for($i = 0; $i <= 6000; $i++){
  $fecha= rand(strtotime("2010-01-01"),strtotime("now"));
  $nombre = "farmaco " . rand(1, 10);
  $cantidad = rand(300, 700);
  //echo date("d-M-Y", $fecha);
  echo "<br/>";
  $sql = "INSERT INTO `registro` (`nombre`, `fecha`, `cantidad`) VALUES ('$nombre',
'".date("Y-m-d", $fecha)."', '$cantidad') ON DUPLICATE KEY UPDATE `cantidad` =
'$cantidad'";
  echo mysqli_query($conn, $sql);
}

?>
```

Código 11. Generación aleatoria de consumo.

Como se puede observar en el Código 11, se trata de un bucle *for* en el cual, para cada iteración, se crea una fecha aleatoria entre el 1 de Enero de 2010 (2010-01-01) y el día de hoy (el día de la ejecución).

Para esa fecha se generan otros dos valores aleatorios, uno para el fármaco

(comprendido en el rango del 1 al 10, pues consideramos que sólo disponemos de 10 fármacos diferentes), y otro para la cantidad consumida de dicho fármaco.

Por último, por cada iteración en el bucle, se hace un *INSERT* para guardar el valor generado en base de datos y así poder hacer uso posteriormente de él.

Si el valor ha sido correctamente insertado, el echo de la última línea devolverá el valor 1, así se puede comprobar si todo ha salido bien sin necesidad de tener que acudir a la base de datos.

Algo a destacar en este fichero es, que como la generación de datos se realiza de manera aleatoria, puede resultar se generen en dos o más iteraciones diferentes, la misma fecha y tipo de fármaco. El resultado esperado sería el de dos o más filas en base de datos con la misma fecha y fármaco, y diferente (también es posible que coincidan, aunque la probabilidad es baja) consumo.

A priori , esto no tiene sentido puesto que suponemos que para un mismo día y fármaco, el personal hospitalario sólo introducirá una vez el consumo ese día.

Para solucionar este problema, basta con añadir en el query string la cláusula ON DUPLICATE KEY. Gracias a ella, y habiendo definido previamente como *primary key (llave primaria)* la tupla *nombre-fecha*, se actualiza el valor del campo *cantidad* para así asegurarnos que sólo se inserta una fila.

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null
Edit Drop	**PRIMARY**	BTREE	Yes	No	id	5148	A	No
Edit Drop	**nombre**	BTREE	Yes	No	nombre	9	A	No
					fecha	5148	A	No

Figura 8. Llave primaria en base de datos.

Para depurar código se ha hecho uso de la herramienta para desarrolladores que incorpora el navegador *Google Chrome*, el *Inspector de Elementos*.

Es una herramienta muy potente que permite ver el código fuente, editarlo, modificar propiedades CSS de los elementos, lanzar llamadas a funciones JavaScript, etc.

Sin duda es una herramienta que ayuda mucho sobre todo para el desarrollo de la parte de Front-End.

En la figura 9 podemos observar, por ejemplo, cómo al cambiar el fármaco seleccionado en form.php se hace una petición POST al fichero grafica.php que devuelve los datos necesarios para obtener la gráfica con google chart.

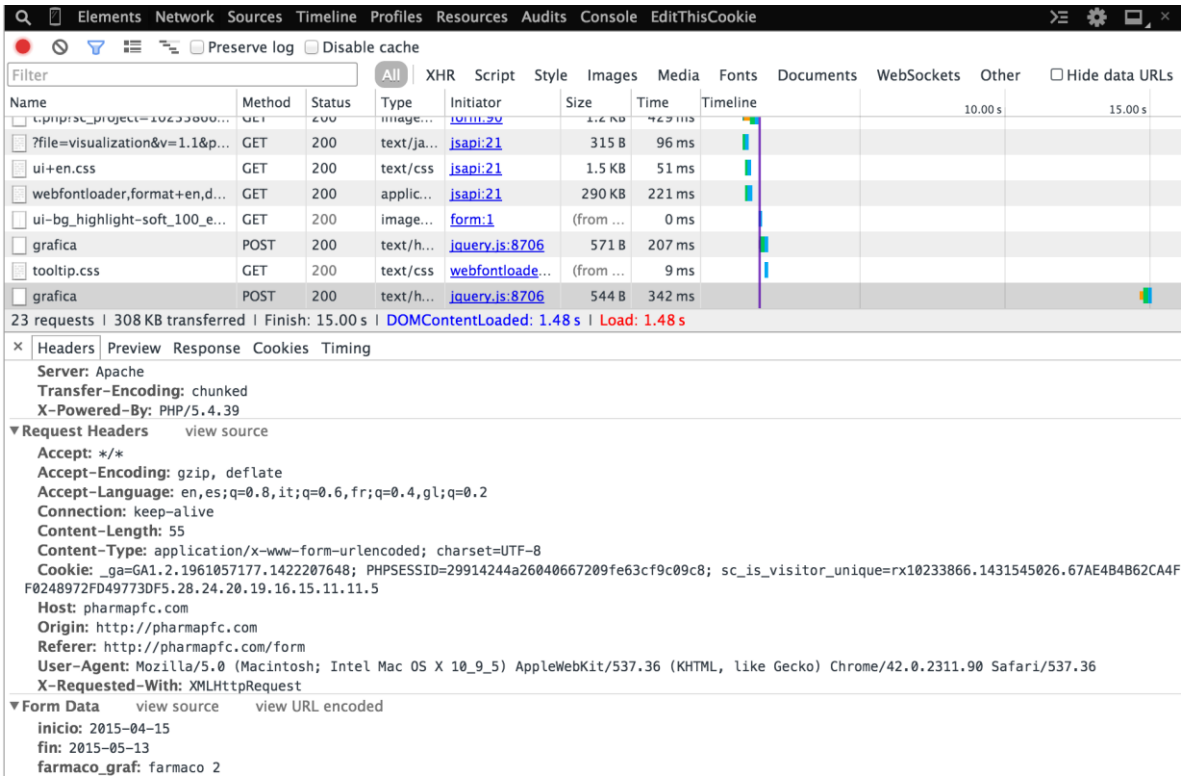


Figura 9. Petición ajax capturada con el inspector de google chrome.

En el apartado *Response* se puede comprobar lo devuelto por dicho fichero:

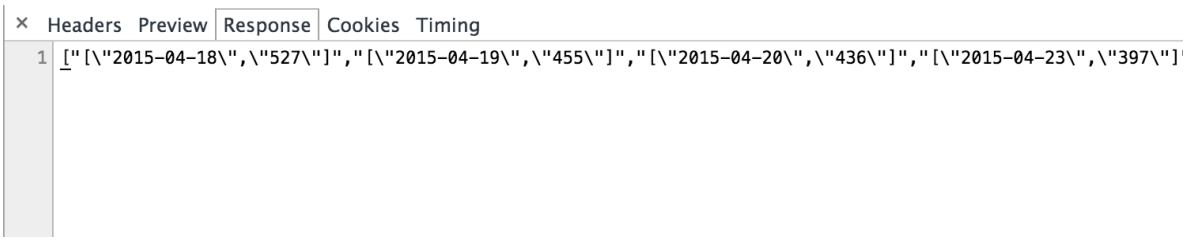


Figura 10. respuesta de la petición ajax capturada.

4.8 CÁLCULO DEL VECTOR DE PEDIDO ÓPTIMO.

Una vez el usuario ha establecido un número de días en el horizonte para los cuales se quiere realizar la estimación, se ejecuta un comando Shell (terminal) que lanza el ejecutable OFH con los parámetros requeridos para el funcionamiento.

Estos parámetros son:

Número de días en el horizonte:

Indican el máximo número de días antes del cual hay que realizar el pedido. Por ejemplo, si decimos que el número de días sea 10, el programa deberá buscar la solución óptima para que al menos se realice un pedido antes de 10 días.

Número de pedidos:

Todos los ficheros pertenecientes al algoritmo de cálculo para la optimización se encuentran en la carpeta OFH. El programa leerá los datos del fichero datos.pha, donde el formato acordado con mi compañero ha sido el siguiente:

Cada día se simboliza con una nueva línea en el fichero. En cada línea se encuentra un número escrito que indica el consumo de ese día. Si ese día no se han consumido unidades de ese determinado fármaco, o simplemente no se disponen datos para el mismo (por ejemplo si el personal del hospital no ha insertado datos para ese día), se indicará mediante un cero en la línea que corresponda. En resumen, el fichero tendrá la siguiente forma:

La llamada al programa C desde un fichero php se realiza mediante la función shell_exec, que nos permite ejecutar comandos como si de una terminal Unix se tratáse.

```
$output = shell_exec('./OFH 10 5');
echo $output;
```

Código 12. Lanzamiento del programa escrito en c.

La salida que nos originialmente nos facilitaba el programa es del estilo a la representada en el Código 13. El programa en C es el resultado de la elaboración de un Proyecto Fin de Carrera elaborado por el compañero César Hoyos Martín.

Posteriormente, nosotros mismos hemos adaptado la salida de dicho programa para que sea más fácil de parsear con PHP, ya que ésta se guarda en una cadena de texto y es más cómodo de trabajar con ella si por ejemplo hemos eliminado previamente los saltos de línea y el adorno que engloba a la palabra *Resultado*. El algoritmo de cálculo del programa no se ha modificado en absoluto.

Además, se ha interpretado un de los errores que devuelve la ejecución de dicho programa para que, si es detectado en nuestro fichero PHP, se detenga la ejecución del mismo y nos devuelva un mensaje de error indicando que algo ha fallado.

```
Jmin= 452.000000

Vector Óptimo de pedido: 1 0 8 0 8 0 8 0 0 1

Stock del pedido óptimo: 10 8 12 3 5 2 8 6 4 2


===============

===Resultado===

===============


13/5/2015

1

15/5/2015

8

17/5/2015

8

19/5/2015

8

22/5/2015

1


Tiempo transcurrido: 0.000000
```

Código 13. Resultado ejecución del programa en c.

La llamada al fichero que lanza el programa en C se hará mediante una petición Ajax desde el fichero calcula.php. La url a la que se realiza la petición será launch-c.php y este fichero se encargará de llamar a la función C, recoger los datos que ésta devuelve, parsearlos para realizar una correcta interpretación, y montar un json que será devuelto como respuesta de la petición Ajax y gracias al cual montaremos la gráfica que nos permite la interpretación de los resultados.

El resultado de la ejecución C se almacena en la variable $output. El tipo de variable es string (es decir, una cadena de texto). Lo ideal sería que el resultado se obtuviese en formato JSON ya que sería fácilmente interpretable por PHP. Como no es así y para respetar el trabajo del compañero, se ha optado por parsear la cadena de texto para obtener la información que nos interesa.

Para empezar, mediante la función “explode” partimos la cadena de texto por cada uno de sus saltos de líneas y lo metemos en una tabla. Es decir, en dicha tabla tendremos cada una de las filas de la salida de nuestro programa en C.

Para la representación gráfica nos interesan dos cosas. La primera es el vector de pedidos calculado teniendo en cuenta el horizonte y el número de pedidos indicado por el usuario. Y en segundo lugar, queremos coger el vector óptimo de pedidos para mostrar al personal del hospital cuál sería la opción más económica a tener en cuenta.

Si observamos la salida de la función en C y teniendo en cuenta que hemos partido la cadena de texto por saltos de línea, nos damos cuenta de que nos interesan la segunda y tercera línea, esto es, el elemento uno y dos de nuestro array.

Antes de continuar con la ejecución normal, comprobaremos que no se ha producido ningún error. En caso de que el algoritmo no haya podido calcular un vector de pedido nos devolverá el siguiente error:

“ERROR9: No existe ninguna posibilidad válida para nuestro problema”

En dicho caso, en el campo “status” del JSON devolveremos el valor ‘*ERROR9’* y

no dejaremos que se pinte la gráfica, sino que mostraremos un mensaje de alerta al usuario.

Si no se produce ningún error, procederemos como anteriormente, partiendo la cadena en diferentes elementos, sólo que esta vez hacemos la partición a través de los espacios. Gracias a la función trim() eliminamos cualquier espacio existente al principio y final de la cadena, con lo cual evitaremos guardar elementos vacío en nuestra tabla.

Una vez hecho el segundo explode(), obtendremos dos tablas que contendrán los valores deseados (en formato de texto, posteriormente se convertirán a entero con la función *intval*).

Por último nos queda montar el JSON. Para ellos, antes de nada vamos a definir la fecha de hoy. Hacemos uso de la función date(), dejando el segundo parámetro vacío obtenemos la cadena de texto que indica la fecha en el instante de la ejecución.

Finalmente vamos a ir recorriendo nuestro vector donde se encuentran guardados los datos de pedido y pedido óptimo. Como cada entrada de la tabla se corresponde a un día en el tiempo, empezando por el día de la ejecución, para insertar en el JSON la fecha deseada que se mostrará en la gráfica, nos vale con sumar a la fecha anterior 86400 segundos (1 día) y tendremos la fecha en la que realizar ese pedido.

Con la función array_push montamos nuestra tabla donde cada elemento será otra tabla asociantiva, y finalmente con la función json_encode codificamos al formato que espera la petición AJAX.

Finalmente añadimos el campo “status” del JSON con el valor OK, para indicar que no ha habido ningún problema y que hemos obtenido los vectores correctamente

```
$output = explode("\n", trim($output, "\n"));

if(trim($output[0]) == "ERROR9: No existe ninguna posibilidad valida para nuestro problema"){
  echo json_encode(array("status" => "ERROR9"));
  die();
}

//Quitamos el inicio de la frase
$pedidos = str_replace("Pedido:", "", $output[1]);
$optimos = str_replace("Optimo:", "", $output[2]);

/*****************************************************
Separamos por espacios y quitamos los de inicio y fin
La posicion cero del array se corresponde con el dia
de hoy y asi sucesivamente.
*****************************************************/
$pedidos = explode(" ", trim($pedidos));
$optimos = explode(" ", trim($optimos));

$date = date("Y-m-d");//dia de hoy

$data = array();
$i = 0;
for($i=0; $i<count($pedidos); $i++){
  array_push($data, array("fecha" => $date, "pedido" => intval($pedidos[$i]), "optimo" =>
intval($optimos[$i])));
  $date = date("Y-m-d", strtotime($date) + 86400);
}
echo json_encode(array("status" => "OK", "data" => $data));
```

Código 14. Parseo del resultado de la ejecución del programa en c.

La otra parte implicada del código es la encargada de interpretar los datos ofrecidos en el JSON y montar la gráfica.

```
var data = new google.visualization.DataTable();
data.addColumn('string', 'Fecha');
data.addColumn('number', 'Pedido');
data.addColumn('number', 'optimo');


for (var i = 0; i < jsonData.length; i++) {
  data.addRow([jsonData[i].fecha, jsonData[i].pedido, jsonData[i].optimo]);
}
```

Código 15. insercción de filas con los resultados.

Como podemos observar en el Código 15, se define una clase google.visualization.DataTable(), y gracias a los métodos que ofrece la librería de Google Chart iremos montando nuestro array que contiene a los datos.

Primero añadimos tres columnas. La primera se corresponde con el eje x y será donde representemos la fecha de nuestro pedido. La segunda será uno de los datos respresentados en el eje y, en este caso la cantidad de unidades a pedir. Por último, la tercera columna indica el número de unidades para el pedido óptimo.

Una vez definidas las columnas, tan solo nos queda ir insertando las filas. Insertaremos una fila por cada valor que nos devuelva el JSON de la petición AJAX.

Para ellos creamos un bucle for, donde en cada iteración se llama al método addRow() con los valores de la fecha(cadena de text), pedido a realizar, y pedido a realizar en el caso óptimo.

Una vez creadas filas y columnas, se llama al método draw() de la clase

charts.Bar para que pinte la gráfica dentro del div indicado.

4.9 PROMEDIO MÓVIL SIMPLE

Para la estimación del consumo mediante el procedimiento del promedio móvil simple, se hace una llamada a la siguiente URL:

http://pharmapfc.com/media-aritmetica?inicio=2015-01-01&fin=2015-02-01&farmaco=farmaco%201

La estructura de la llamada es la siguiente. Se trata de una petición get al fichero media-aritmetica.php. Además, se deben pasar tres valores obligatorios para realizar la estimación (*inicio, fin* y *farmaco*).

Como hemos visto, el promedio móvil simple consiste en una media arimética sobre un conjunto de valores de consumo de un determinado fármaco.

Este conjunto de valores viene definido por los parámetros inicio y fin. Ambos son fechas en formato “yyyy-mm-dd” y definen el rango de valores del fármaco consumido a tener en cuenta. Este rango temporal se puede definir como ventana, y sobre ella calcularemos la media.

Todos los parámetros pasados como argumentos se leerán en el fichero mediante la palabra reservada $_GET[‘param’].

A continuación se definen los parámetros utilizados:

$inicio: Fecha de inicio de la ventana temporal en formato *yyyy-mm-dd*

$fin: Fecha de fin de la ventana temporal en formato *yyyy-mm-dd*

$farmaco: Fármaco sobre el cual se quiere realizar la estimación.

Una vez guardados los parámetros en variables, comenzamos montando la consulta a base de datos que nos devuelva la información del consumo almacenado.

```
$sql = "SELECT cantidad, fecha FROM `registro` WHERE `nombre` LIKE '$farmaco' AND `fecha` <= '$fin' AND `fecha` >= '$inicio' ";
```

Código 16. Selección de consumo en una ventana temporal.

En las variables $fin, $inicio y $farmaco se encuentran almacenados los parámetros proporcionados.

En la figura 11 podemos ver un ejemplo de los resultados obtenidos:

cantidad	fecha
408	2015-01-01
388	2015-01-06
431	2015-01-08
677	2015-01-12
645	2015-01-14
565	2015-01-16
374	2015-01-17
595	2015-01-18
376	2015-01-22
479	2015-01-24
405	2015-01-25
657	2015-01-26
319	2015-02-01

Figura 11. resultados obtenidos directamente al ejecutar la query en base de datos.

Mediante un SELECT, nos traemos de la tabla `registro` todos los campos que correspondan con el fármaco seleccionado y además pertenezcan al intervalo temporal definido como [$inicio, $fin].

```
$result = mysqli_query($conn, $sql);
$sum = 0;
while($data = mysqli_fetch_assoc($result)){
  $sum += $data['cantidad'];
  $contenido .= $data['cantidad'] . "\n";
}
```

Código 17. Sumatoria de todos los consumos obtenidos.

Como lo que nos interesa a fin de cuentas es almacenar los valores en un fichero para que nuestro programa en C pueda leerlo, vamos a coger el contenido del fichero “datos.pha”, lo vamos a borrar, y vamos a ir insertando los valores que vamos obteniendo.

Con el bucle ‘while’ de la figura de arriba vamos sacando los valores obtenidos en la consulta y a su vez los vamos sumando en la variable $sum. Cuando el bucle haya finalizado, en esta variable tendremos la suma de todo el consumo de dicho fármaco para el intervalo de tiempo seleccionado.

Por último, lo vamos escribiendo en la variable $contenido, que será la cadena de texto que al final escribamos en nuestro fichero datos.pha

Una vez tenemos la suma, lo único que queda es dividirlo por el número total de elementos calculados. Una manera muy elegante de hacerlo es mediantes la función nativa de php ‘mysqli_affected_rows()’.

A esta función se le pasa como argumento el objeto de conexión mysql y te devuelve el número de filas obtenidas en la última consulta.Por tanto nos basta con dividir la variable $sum con el valor devuelto por dicha función y tendremos el

promedio calculado.

```
$media = $sum / mysqli_affected_rows($conn);
echo "media: " . $media;
```

Código 18. Cálculo de la media aritmética

Para tener suficientes valores como para que el programa en C pueda funcionar, vamos a meter 40 líneas de valores en el fichero ***datos.pha***. Primero hemos insertado los datos del pasado que hemos recogido de base de datos, y el resto de valores que faltan hasta 40 los iremos rellenando con el valor de la media aritmética calculado.

```
for($i= 40-mysqli_affected_rows($conn); $i>0; $i--){
  $contenido .= round($media) . "\n";
}
```

Código 19. Completamos las líneas hasta tener 40 filas.

Finalmente, con la función file_put_contents() escribimos la cadena resultante en el fichero datos.pha

4.10 ALISAMIENTO EXPONENCIAL

El método de pronóstico de alisamiento o suavizamiento exponencial pertenece a la categoría de series de tiempo, es decir, aquellos métodos donde se utiliza información de la demanda histórica para poder pronosticar el futuro. Su nombre se debe a que cada incremento del pasado se reduce en **(1 – α)** por lo cual se considera válido que la importancia de los datos disminuye en la medida

que son más antiguos.

Para poder generar un pronóstico a través del método de alisamiento exponencial necesitamos el pronóstico más reciente, la demanda que se presentó para ese período y una constante de suavizamiento α.

$$F_t = F_{t-1} + \alpha\,(A_{t-1} - F_{t-1})$$

El valor de alfa es entre 0 y 1. En esta escala para valores de alfa relativamente pequeños se reducen las variaciones de corto plazo asociadas al pronóstico lo cual es razonable cuando la demanda real tiene un comportamiento relativamente estable. Sin embargo, si la demanda presenta cambios significativos en el corto plazo nos interesará seguir éstos más de cerca y en ese caso debiéramos seleccionar una constante alfa más grande.

En nuestro problema en concreto, hemos desarrollado un algoritmo que iterará, de modo que se vaya incrementando el valor de alfa con un paso de 0.1 en cada iteración.

Una vez dispongamos de todos los pronósticos para todos los valores de alfa, simplemente nos limitaremos a elegir aquél que tenga un error menor. Esto significa que iremos guardando el valor en una tabla para posteriormente poder estudiarlos.

Para este cálculo hemos usado los datos reales de consumo del fármaco (denominado de aquí en adelante fármaco 1 por temas de confidencialiad) proporcionados por el Hospital Reina Sofía de Córdoba, desde el 05-01-2012 hasta el 13-12-2013

Hemos ido eligiendo diferentes ventanas de tiempo, todas de duración aproximada igual a cuatro meses, y hemos utilizado el fichero *alisamiento-exponencial.php* para ir calculando el pronóstico y el error del consumo.

```
$fin = date("Y-m-d", strtotime("2012-12-13"));
$inicio = date("Y-m-d", strtotime("2012-09-13"));
$farmaco = "farmaco 1";

$sql = "SELECT * FROM `registro` WHERE `nombre` LIKE '$farmaco' AND `fecha` <= '$fin'
AND `fecha` >= '$inicio' ";
$result = mysqli_query($conn, $sql);

$consumos = array();
while($item = mysqli_fetch_assoc($result)){
  array_push($consumos, $item['cantidad']);
}
```

Código 20. Insercción del consumo del "fármaco 1" en base de datos.

Dicha ventana temporal viene definida por el valor de las variables *fin e inicio.* Con estas variables y seleccionando sólo los valores del campo "trastuzumab" de base de datos, formamos una query que nos devolverá los datos de la cantidad para cada día en ese intervalo de tiempo. Una vez disponemos de todos los datos, procedemos como es común, a insertarlos uno a uno en una tabla ($consumos) con la función *array_push*.

A continuación, asignamos el valor inicial $\alpha = 0.1$ y creamos un bucle while dentro del cual iremos aumentando el valor de alfa en 0.1 por iteración. Como alfa tiene que estar comprendido entre 0 y 1, la condición del bucle while será alfa >= 1

En cada iteración, aplicamos la ecuación que nos calcula el pronóstico de consumo para el intervalo i-*ésimo* , que escrita en php resulta ser cómo en el Código 21.

```
$pronosticos[$i] = $consumos[$i-1] * $alpha + (1 - $alpha) * $pronosticos[$i-1];
```

Código 21. Cálculo del pronóstico en el intervalo de tiempo i-esimo.

```
array_push($errors, abs($pronosticos[$i] - $consumos[$i]));
```

Código 22. Cálculo del error.

Por último, se calcula el valor medio del error por cada valor de alfa como se indica en el Código 22.

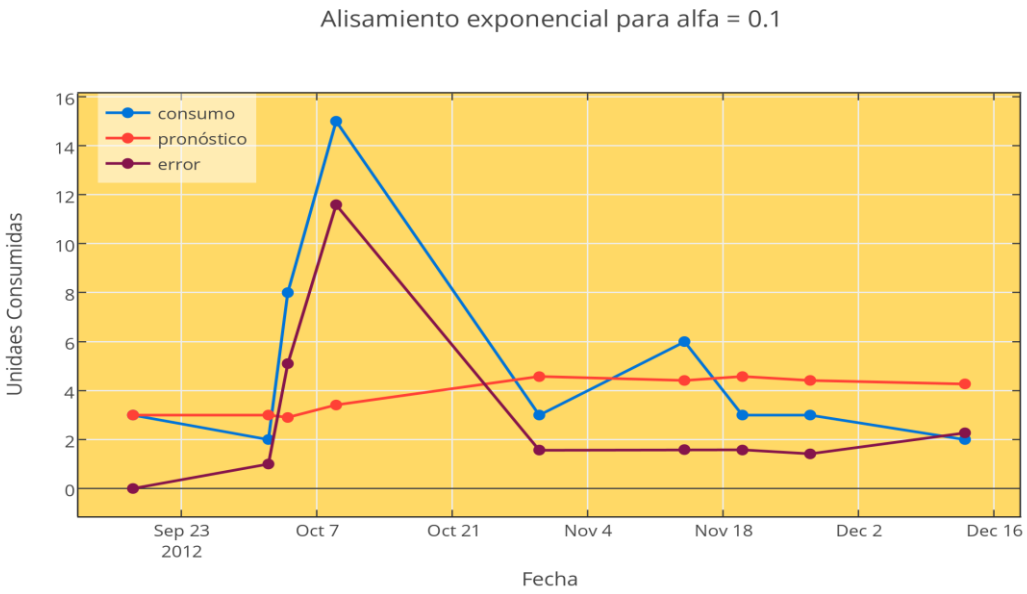
```
$average = array_sum($errors) / count($errors);
```

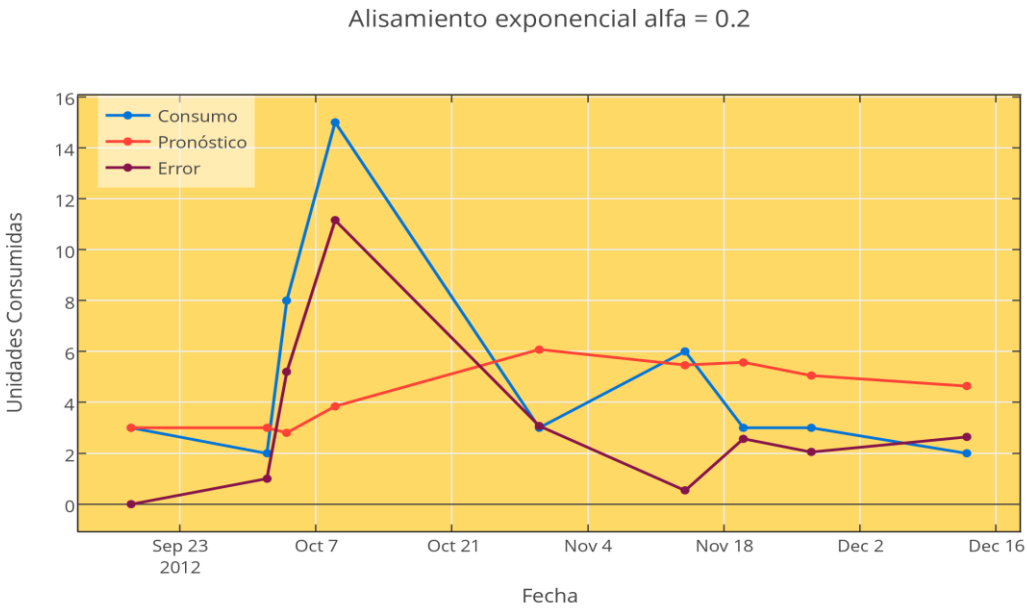
Código 23. Cálculo del error medio.

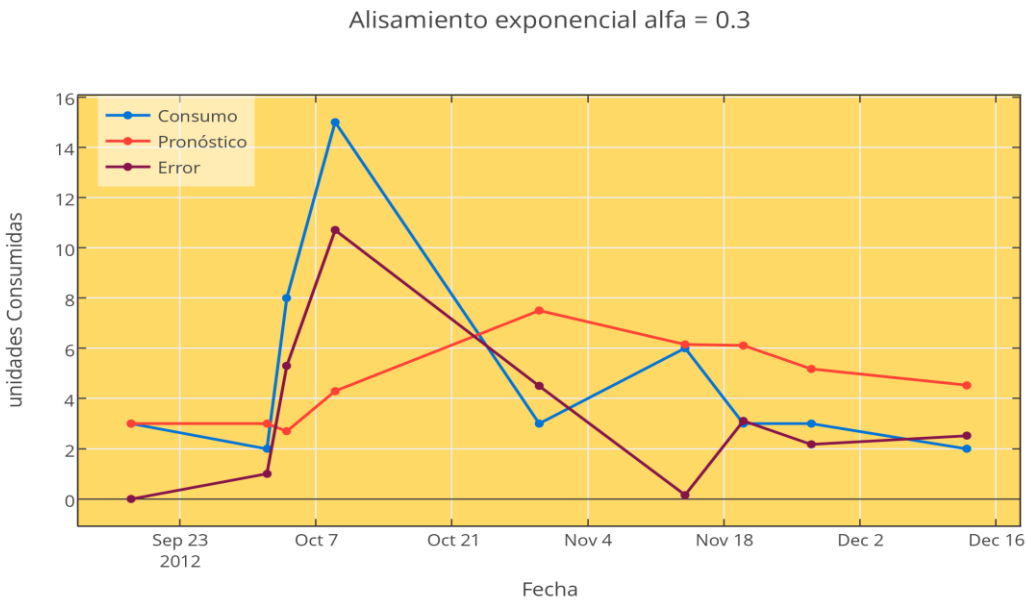
A continuación se representan diferentes pronósticos obtenidos para una ventana temporal determinada por el intervalo [13-09-2012, 13-12-2012] y para el fármaco trastuzumab. Para la representación se ha utlizado la herramienta gráfica plotly, donde hemos importado los ficheros generados en cada iteración del bucle *while*, en el que hemos guardado, separado mediante tabuladores y filas, la fecha, consumo, pronóstico y error calculados.

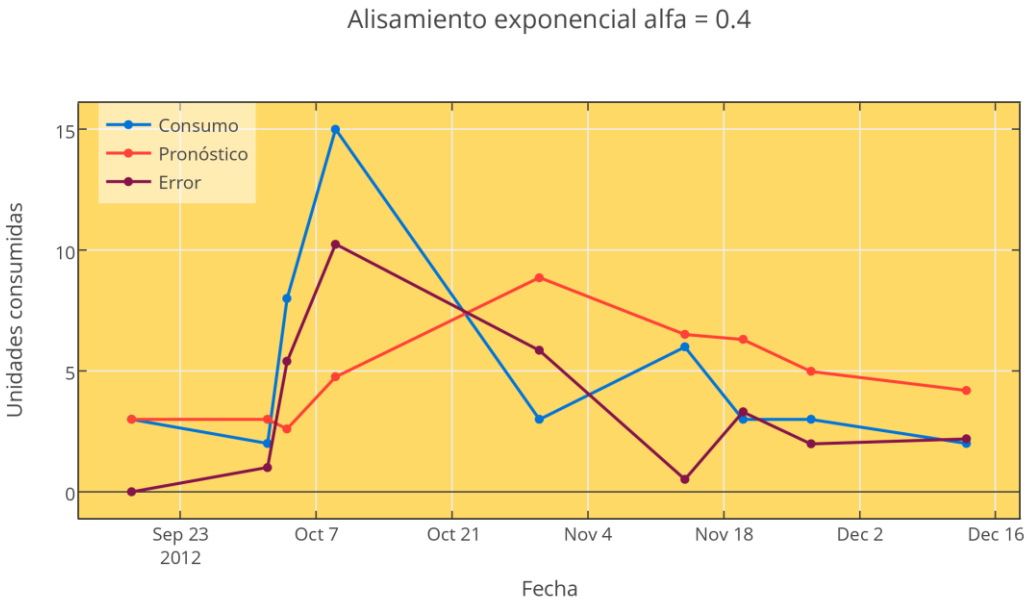
```
$fin = date("Y-m-d", strtotime("2012-12-13"));
$inicio = date("Y-m-d", strtotime("2012-09-13"));
$farmaco = "farmaco 1";
```

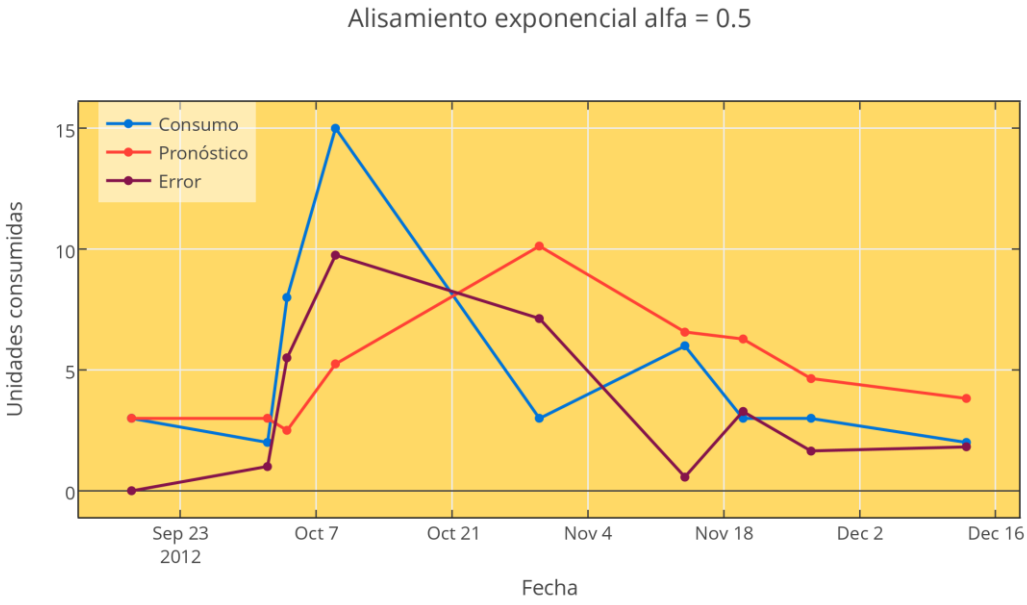
Código 24. Valor de las variables para este ejemplo concreto.

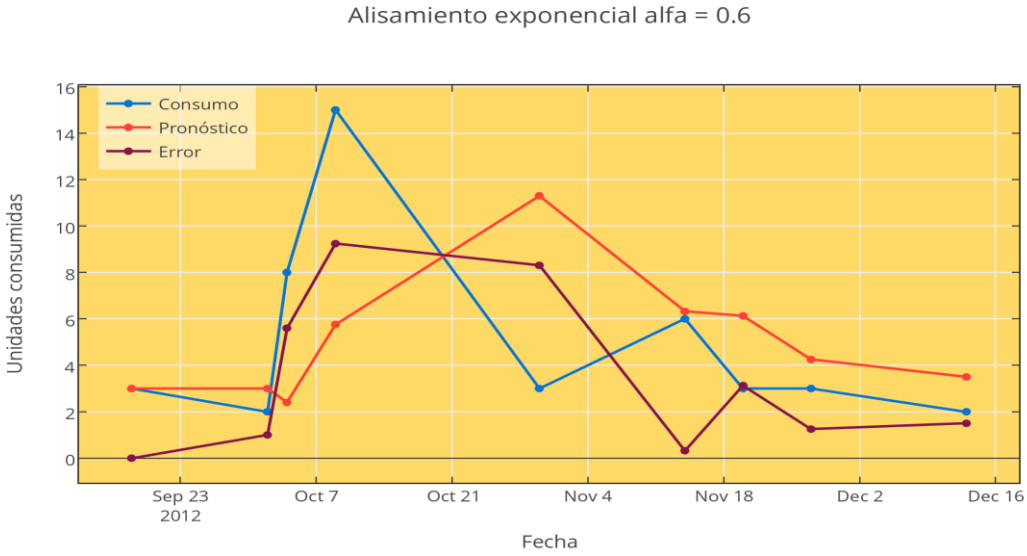

Alisamiento exponencial para alfa = 0.1
consumo
pronóstico
error
Unidades Consumidas
Fecha
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
0
2
4
6
8
10
12
14
16

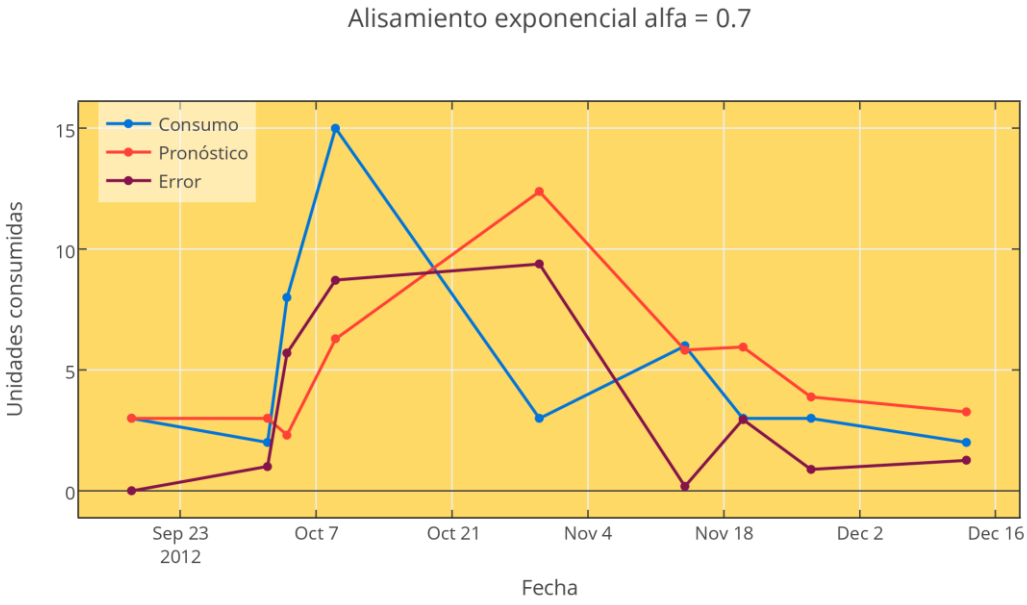

Alisamiento exponencial alfa = 0.2
Consumo
Pronóstico
Error
Unidades Consumidas
Fecha
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
0
2
4
6
8
10
12
14
16

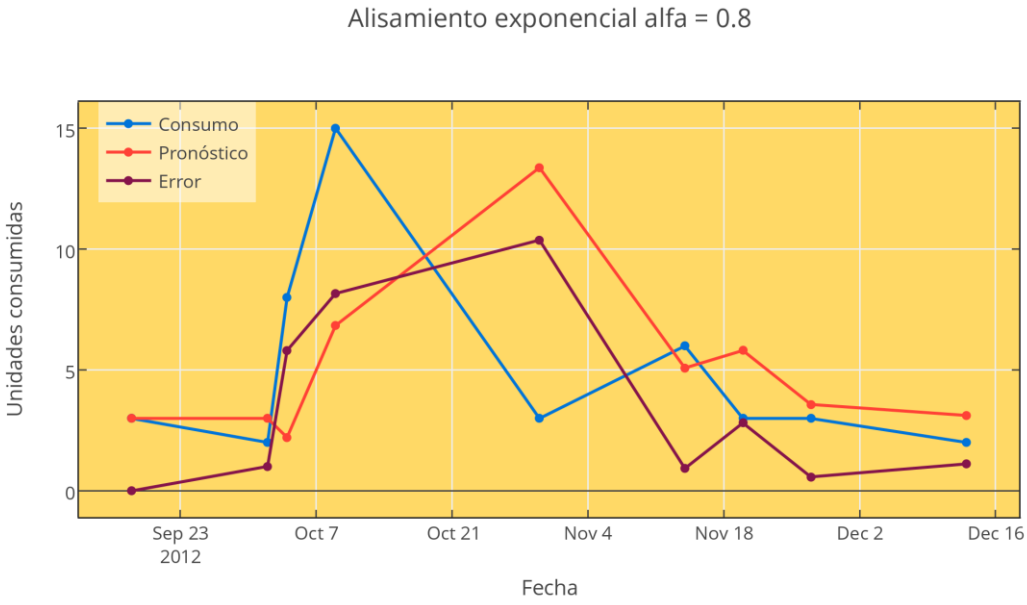

Alisamiento exponencial alfa = 0.3
Consumo
Pronóstico
Error
unidades Consumidas
16
14
12
10
8
6
4
2
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha

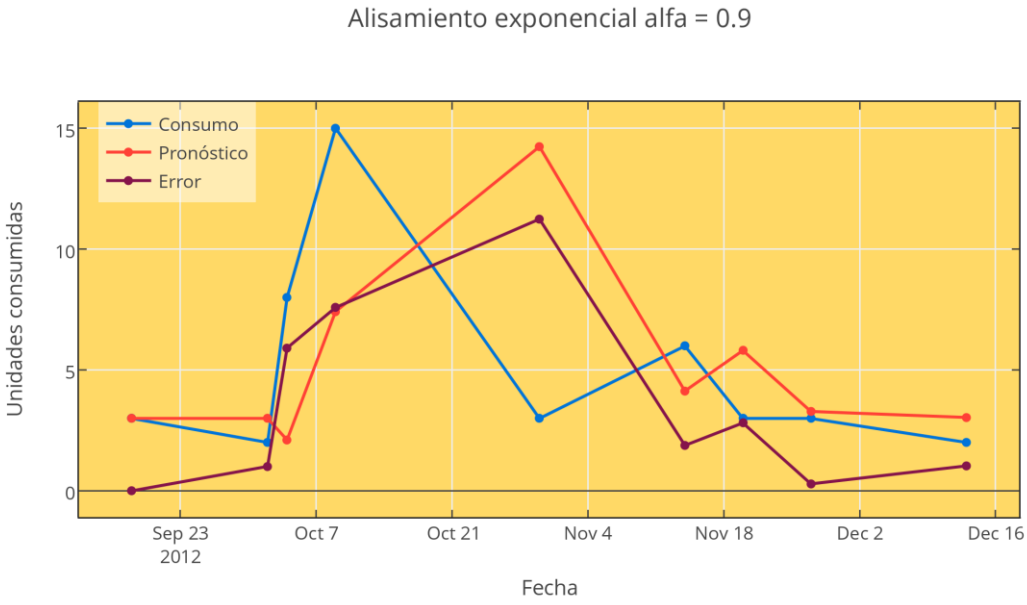

Alisamiento exponencial alfa = 0.4
Consumo
Pronóstico
Error
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha

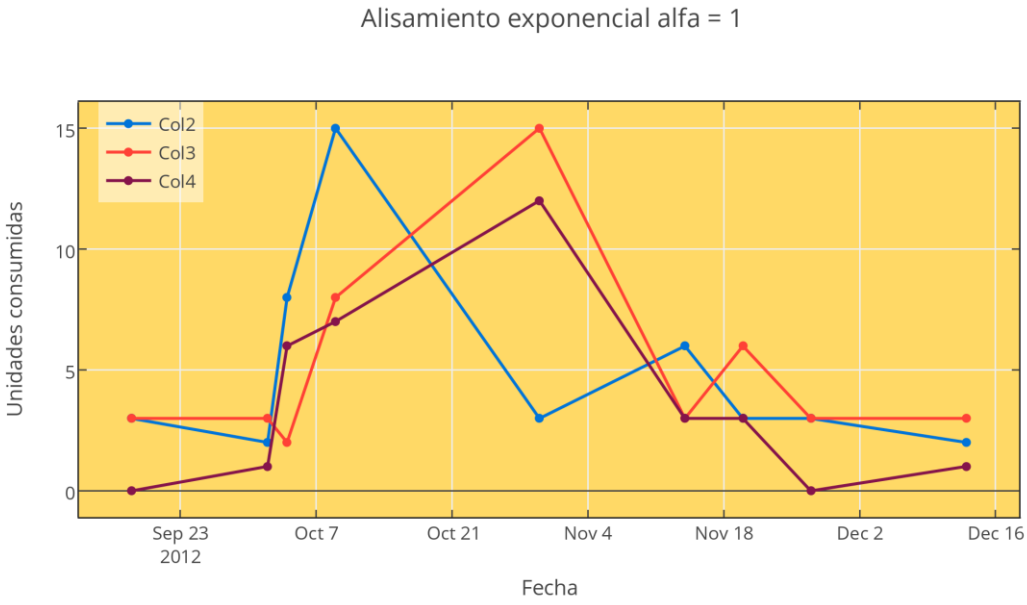

Alisamiento exponencial alfa = 0.5
Consumo
Pronóstico
Error
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha


Alisamiento exponencial alfa = 0.6
Consumo
Pronóstico
Error
Unidades consumidas
16
14
12
10
8
6
4
2
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha


Alisamiento exponencial alfa = 0.7
Consumo
Pronóstico
Error
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha


Alisamiento exponencial alfa = 0.8
Consumo
Pronóstico
Error
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha


Alisamiento exponencial alfa = 0.9
Consumo
Pronóstico
Error
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha


Alisamiento exponencial alfa = 1
Col2
Col3
Col4
Unidades consumidas
15
10
5
0
Sep 23
2012
Oct 7
Oct 21
Nov 4
Nov 18
Dec 2
Dec 16
Fecha

4.11 SUBIDA DE FICHEROS

Con la intención de hacer más rápida la inserción de información sobre consumo de un determinado fármaco, se ha implementado un algoritmo que permite la subida de ficheros de texto donde se encuentren almacenados los históricos de consumo de dicho fármaco y los vaya insertando en base de datos.

La subida de ficheros se hace mediante un determinado tipo de formularios HTML.

Antes de nada hay que habilitar dicha subida de ficheros en nuestro servidor. Esto se hace configurando el fichero php.ini, poniendo el valor de "*file_uploads = On*". Si no se ha configurado, la subida de ficheros no estará permitida.

A continuación debemos crear un formulario en la parte de la web desde donde queramos subir ficheros. Se puede ver el comportamiento descrito en el Código 25.

```
<form action="upload" method="post" enctype="multipart/form-data">
    <h3>Subir Fichero:</h3>
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Subir Fichero" name="submit">
</form>
```

Código 25. Formulario para la subida de ficheros.

Es muy importante que el método por el que se envía el formulario sea POST para que funcione correctamente.

Al crear un campo "input" donde el tipo sea "type=file", el propio navegador lo interpretara como un gestor de ficheros, así cuando hagamos click sobre él se nos desplegará la típica ventana donde podremos seleccionar el fichero que queramos subir.

Una vez hayamos seleccionado el fichero y hagamos click en el botón de subida, el navegador nos redireccionará al archivo upload.php, el cual se explica a continuación.

Este fichero se puede dividir en dos partes. La primera se encarga de coger el fichero subido y guardarlo en una carpeta seleccionada. La segunda, se encarga de abrir el fichero que acabamos de guardar, leerlo, interpretarlo, e insertar los datos en nuestra base de datos. A continuación explicaremos con más detalle ambas partes del fichero.

```
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);

if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo "El archivo ". basename( $_FILES["fileToUpload"]["name"]). " fue cargado con exito.";
} else {
    echo "Error al subir el fichero.";

    die();
}
```

Código 26. Manejo del fichero subido.

En la variable *$target_dir* guardamos el nombre de la carpeta donde queremos guardar los ficheros subidos. Luego, creamos la ruta donde guardaremos el fichero, esta estará compuesta por la carpeta que hemos elegido y por el nombre del fichero que hemos cargado. Para obtener el nombre del fichero, hacemos uso uso de la variable PHP *$_FILES*, y accedemos a la clave "*fileToUpload*" que es como hemos llamado al campo input en el formulario anterior. Luego tomamos el nombre del fichero accediendo a la clave "*name*" de dicha variable.

Una vez obtenido esto, hacemos uso de la función *basename* para concatenar la cadena con el nombre de la carpeta y así obtener la ruta final donde guardaremos el fichero para su posterior lectura.

Para mover el fichero subido a la ruta deseada, hacemos uso de la función de php "*move_uploaded_file*", a la cual pasamos como argumento la ruta a la que mover el fichero subido.

Si dicha función devuelve un 1, podemos asegurar que la subida se ha hecho adecuadamento y que el fichero se ha movido al directorio deseado. Si no, imprimimos un mensaje de error para que el usuario sea consciente de que ha ocurrido algún fallo y terminamos la ejecución para que no se ejecute la segunda parte del fichero.

Como hemos mencionado en el párrafo anterior, si todo ha ido bien la ejecución continuará con normalidad y pasaremos a leer el fichero para su posterior inserccion en base de datos.

Para que todo el procedimiento se lleve a cabo con éxito, la estructura del fichero subido debe ser la descrita a continuación.

Se escribirá una línea por cada entrada de datos. Esta línea estará formada por una fecha en cualquier formato aceptado por PHP, ya que se hace uso de la función strtotime para la conversión de texto a número de segundos desde el 1 de enero de 1970, fecha muy utilizada bajo sistemas UNIX.

Los formatos aceptados se pueden consultar en la Figura 12. Dichos formatos han sido extraídos de la documentación oficial del lenguaje PHP.

A continuación de la fecha, se inserta una tabulación, la cual se recoge en el código mediante el indicado "\t", el cual significa tabulación. Inmediatamente después del tabulador se indica el numero de fármacos consumidos para esa fecha.Un ejemplo de fichero podría ser el mostrado en la Figura 13.

En este caso se trata del consumo real de un fármaco cedido por el Hospital Reina Sofia de Córdoba, el consumo se indica con números enteros negativos, pero en

nuestra plataforma hemos decidido trabajar con el consumo como números enteros positivos.

Notaciones regionales

Descripción	Formato	Ejemplos
Mes y día americanos	*mm* "/" *dd*	"5/12", "10/27"
Mes, día y año americanos	*mm* "/" *dd* "/" *y*	"12/22/78", "1/17/2006", "1/17/6"
Año de cuatro dígitos, mes y día con barras	*YY* "/" *mm* "/" *dd*	"2008/6/30", "1978/12/22"
Año de cuatro dígitos y mes (GNU)	*YY* "-" *mm*	"2008-6", "2008-06", "1978-12"
Año, mes y día con guiones	*y* "-" *mm* "-" *dd*	"2008-6-30", "78-12-22", "8-6-21"
Día, mes y año de cuatro dígitos, con puntos, tabulaciones o guiones	*dd* [.\t-] *mm* [.-] *YY*	"30-6-2008", "22.12\t1978"
Día, mes y año de dos dígitos, con puntos o tabulaciones	*dd* [.\t] *mm* "." *yy*	"30.6.08", "22\t12\t78"
Día, mes textual y año	*dd* ([\t.-])* *m* ([\t.-])* *y*	"30-June 2008", "22DEC78", "14 III 1879"
Mes textual y año de cuatro dígitos (el día se restablece a 1)	*m* ([\t.-])* *YY*	"June 2008", "DEC1978", "March 1879"
Año de cuatro dígitos y mes textual (el día se restablece a 1)	*YY* ([\t.-])* *m*	"2008 June", "1978-XII", "1879.MArCH"
Mes textual, día y año	*m* ([.\t-])* *dd* [„.stndrh\t]+ *y*	"July 1st, 2008", "April 17, 1790", "May.9,78"
Mes textual y día	*m* ([.\t-])* *dd* [„.stndrh\t]*	"July 1st,", "Apr 17", "May.9"
Día y mes textual	*d* ([.\t-])* *m*	"1 July", "17 Apr", "9.May"
Abreviatura de mes, día y año	*M* "-" *DD* "-" *y*	"May-09-78", "Apr-17-1790"
Año, abreviatura de mes y día	*y* "-" *M* "-" *DD*	"78-Dec-22", "1814-MAY-17"
Año (y sólo el año)	*YY*	"1978", "2008"
Mes textual (y sólo el mes)	*m*	"March", "jun", "DEC"

Figura 12. Formatos de fecha válidos en PHP.

1/5/12	-3
1/10/12	-6
1/21/12	-7
2/1/12	-6
2/1/12	-9
2/7/12	-5
2/10/12	-7
2/16/12	-1
2/20/12	-9
2/24/12	-6
3/14/12	-3
3/16/12	-4
3/20/12	-3
3/29/12	-2

Figura 13. formato de fichero apto para la subida automática.

5. LÍNEAS DE DESARROLLO FUTURO

5.1 DISEÑO RESPONSIVE

Puesto que cada día es más grande el alcance de los dispositivos móviles como smartphones y tablets, y que su incorporación al mundo laboral supone una indiscutible mejora, se ha utilizado un *diseño adaptativo* de la plataforma.

El diseño adaptativo, conocido como RWD (Resposive Web Design), es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las aplicaciones o páginas web al dispositivo que se está utilizando.

Dicha tecnología pretende que con un solo diseño, tengamos una visualización adecuada en cualquier plataforma. El rápido crecimiento de los dispositivos móviles es el que ha hecho que el RWD se vuelva tan popular.

Este tipo de prácticas también evita tener que desarrollar un aplicación nativa para cada S.O. en el mercado, esto es, nos evita por ejemplo que tengamos que desarrollas aplicaciones para Android y Iphone. Con un simple navegador instalado en el móvil seríamos capaces de interactuar perfectamente con la aplicación web.

Para este proyecto, hemos decidido usar el framework de HTML, CSS y JavaScript Bootstrap para realizar un diseño responsive.

Pese a que no es el objetivo principal del proyecto, se ha optado por mejorar la visualización en dispositivos móviles, para que en futuro sea mucho más sencillo continuar con el desarrollo de la plataforma, ya que la estructura HTML juntos con las hojas de estilo, definen la columna vertebral de una aplicación web como esta.

Figura 14. ejemplo de diseño responsive.

Dicho esto, vamos a describir brevemente en que consiste el framework utilizado.

Antes de nada, debemos incluir el fichero “bootstrap.min.css” en nuestro proyecto e incluirlo en él. Esto se hace en el header junto con el resto de ficheros que hemos utilizado:

```
<link href="/css/bootstrap.min.css" rel="stylesheet">
```

Código 27. include css de bootstrap.

En este proyecto, el framework bootstrap se ha utilizado para definir la parrilla (grid) de elementos y bloques de la plataforma. Esto es, por ejemplo, en el fichero form.php, disponemos en primer lugar de dos formularios, a continuación de una

gráfica, y por último de los tres iconos que aparecen en toda la plataforma (logos de los hospitales y de la universidad de Sevilla).

Bootstrap define una serie de clases de las cuales podemos hacer uso para situar los elementos a nuestro antojo. Hay que imaginar que el ancho de la pantalla se divide en 12 celdas y luego asignar el número de celdas que queramos a cada elemento. Por ejemplo, en el caso de los formularios, hemos dividido la pantalla en cuatro celdas y las hemos repartido de la siguiente manera:

La pimera celda va vacía para que así los formularios queden centrados. En la segunda celda hemos colocado el formulario para insertar datos. En la tercera hemos colocado el formulario para acceder a base de datos y obtener un histórico de consumo. Y por último, en la cuarta celda no hemos insertado nada, luego va vacía al igual que la primera.

Bootstrap define estas celdas como columnas, y podemos distibuirlas creando bloques que pertenezcan a una determinada clase u otra. Las clases tienen una estructura definida, y en este caso hemos usado la del tipo “col-md-X”.

Estas clases significan lo siguiente. Siempre comienzan con “col-“, a continuación se hace referencia al tipo de pantalla que vamos a utilizar (en este caso md significa médium desktop, pero hay muchos más como xs, lg, etc). Y con el número X asignamos la cantidad de columnas para ese bloque, por tanto en el caso de los formularios será col-md-4.

En la figura 15 se puede ver un resumen de diferentes parrillas de elementos.

Por último, simplemente añadir que un elemento puede tener varias clases y así definir diferentes comportamientos en función del dispositivo desde el cual se acceda a la plataforma.

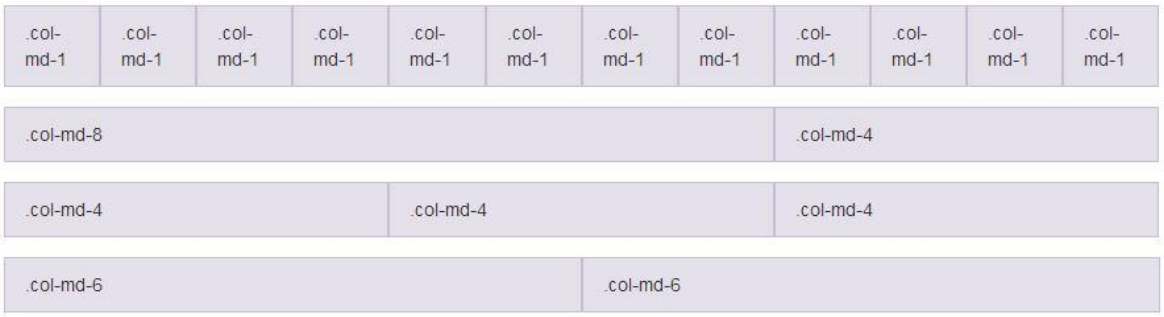


Figura 15. ejemplo de rejilla bootstrap con la escpicificación de las clases empleadas.

6. ANEXOS

6.1 MANUAL DE USUARIO.

El propósito de este manual no es otro que el de explicar, sin incurrir en aspectos demasiado técnicos, el uso de la plataforma desarrollada, de modo que sea de utilidad para el personal sanitario o para cualquier desarrollador que quiera continuar con la plataforma.

Como hemos mencionado en las líneas de introducción del capítulo cuatro, la aplicación se compone fundamentalmente de las siguientes partes:

1. Acceso de usuarios.
2. Menú principal.
3. Histórico de consumo.
4. Panel de control.
5. Subida de ficheros.
6. Cálculo del pedido óptimo.

Además de los citados arriba, existen otros módulos en la arquitectura, pero en principio no tendrán cabida en esta manual ya que el usuario no puede interaccionar directamente con ellos.

6.1.1 Acceso de usuarios.

Es el denominado *index* de la aplicación. Una vez el usuario accede al dominio, lo primero que se le presenta será un formulario donde se le pide que introduzca su usuario y contraseña.

Si el usario no dispone de una cuenta, deberá contactar con un administrador del sistema para que le cree una cuenta con un nuevo usuario y contraseña. En este proyecto, no se ha implementado la posibilidad de crear una cuenta de usuario si

no es a través de un administrador. Ésto se ha hecho así para una primera versión de la plataforma.
El administrador podrá crear cuantas cuentas quiera desde su panel de control. Se explicará más adelante en el apartado 6.1.4.

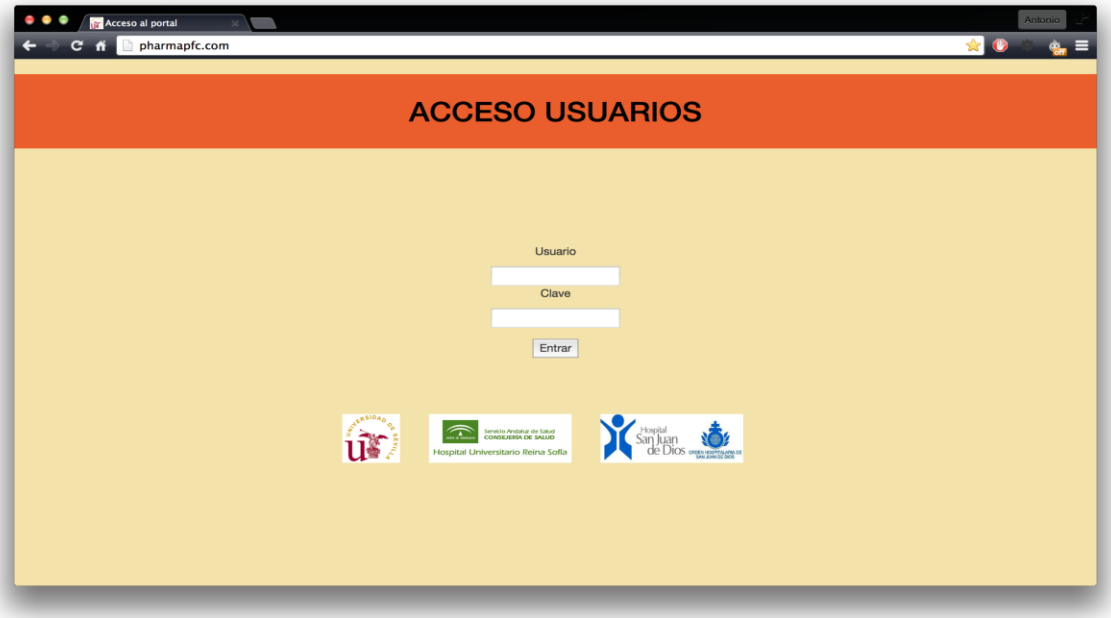


Figura 16. Acceso usuarios.

Si el usuario dispone de una cuenta, podrá acceder a la plataforma introduciendo sus credenciales (usuario y contraseña). Una vez introducidas, se comprobarán con las almacenadas en base de datos y, en el caso de que se introduzcan de manera errónea, nos aparecerá el error de la figura 17.

Por último, recordar también que no será posible acceder directamente a cualquier contenido de la página sin haber iniciado sesión, y que, aún habiendo iniciado sesión con anterioridad, si el navegador ha terminado la sesión, al intentar acceder a cualquier contenido de la web se nos redirigirá a la pantalla de login para que el usuario se identifique.

Figura 17. error en el acceso de un usuario.

6.1.2 Menú Principal.

Si el usuario y contraseña introducidos son correctos, la plataforma nos direccionará al menú principal de la aplicación. En él, podemos distinguir cuatro partes, tal y como se puede apreciar en la figura 19.

En la esquina superior derecha nos aparacerá un menú desplegable con el que podemos movernos por toda la aplicación. Éste menú aparecerá siempre, independientemente del módulo en el que nos encontremos, y su contenido dependerá de los permisos de los que disponga el usuario registrado.

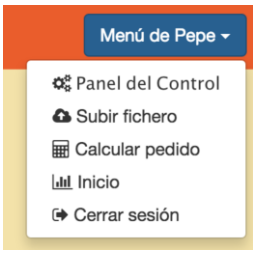


Figura 18. Menú principal.

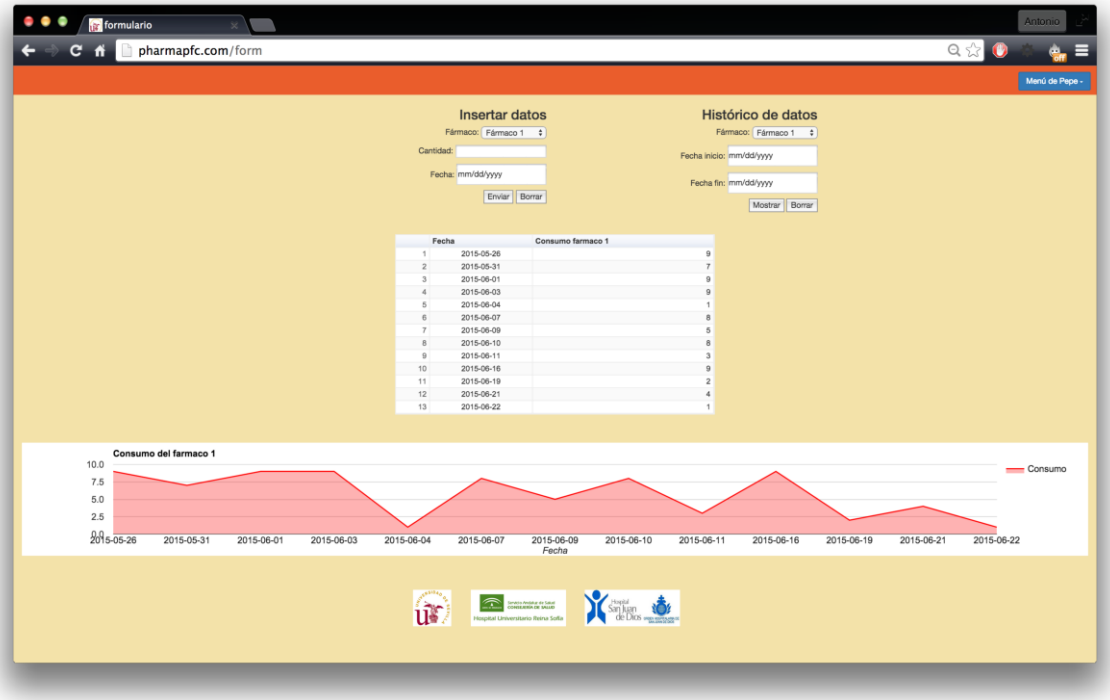


Figura 19. Pantalla principal.

A través del formulario de la figura 20, el usuario podrá guardar en base de datos el consumo de un determinado fármaco para una determinada fecha. Para ello, deberá seleccionar un fármaco de la lista desplegable. A continuación, introducirá la cantidad consumida en la fecha que corresponda a la tercera entrada del formulario. Las fechas deberán introducirse en notación ISO 8601, esto es, con

formato YYYY/mm/dd . La separación se hará mediante (/). También puede seleccionarse un día concreto haciendo click en el calendario desplegable que se nos ofrece. Es muy importante introducir correctamente el formato de la fecha para que los datos almacenados sean coherentes. El consumo se deberá considerar positivo, un consumo negativo significará que han entrado fármacos en el almacén.

Figura 20. Formulario de insercción de datos.

A la derecha de este formulario, tenemos otro que aparece bajo el título de *Histórico de datos*. A través de él, podremos hacer una consulta a base de datos donde se nos mostrará el consumo de un determinado fármaco.

Para ello, el usuario deberá de seleccionar el fármaco sobre el cuál realizar la consulta meidante el desplegable. También, deberá escoger un período de tiempo mediante los campos *Fecha inicio* y *Fecha fin.*
Cuando se haga click en mostrar, una nueva pestaña se abrirá en el navegador y apareceran los consumos para ese fármaco. En el apartado 6.1.3 se profundizará más sobre este módulo.

Figura 21. Formulario para la consulta del consumo.

Las siguientes dos partes que se presentan al usuario están relacionadas entre sí. Como se puede observar en la figura 19, se trata de una gráfica de consumo para el fármaco que se encuentre seleccionado en el selector del apartado *Insertar datos.* En dicha gráfica, se representa el consumo frente a la fecha de las últimas cuatro semanas. En la tabla situada encima de la gráfica, se encuentran, para mayor claridad, los datos representados en la gráfica.

6.1.3 Histórico de consumo.

Si hemos optado por rellenar el formulario de "histórico de consumo" en el menú principal, se nos abrirá una nueva petaña donde vemos varias gráficas representadas.

Esta ventana se divide en dos secciones. La primera de ellas nos muestra el histórico del consumo para ese fármaco. Es decir, obtiene todos los datos de consumo almacenados en base de datos y los presenta en dos formatos, una tabla en el margen izquierdo, y un calendario en el derecho.

Si posicionamos el ratón sobre un día concreto del calendario, se nos desplegará una pequeña ventana que nos indica el número de unidades consumidas de ese medicamento y la fecha en la que se consumieron.

La segunda sección consiste en una gráfica donde se representan las unidades consumidas frente al tiempo. A diferencia de la sección anterior, ésta gráfica sólo representa aquellos consumos que se hayan realizado en el período de tiempo indicado en el formulario. Además, se puede ver una línea roja que representa la media aritmética del consumo en el intervalo de tiempo seleccionado.

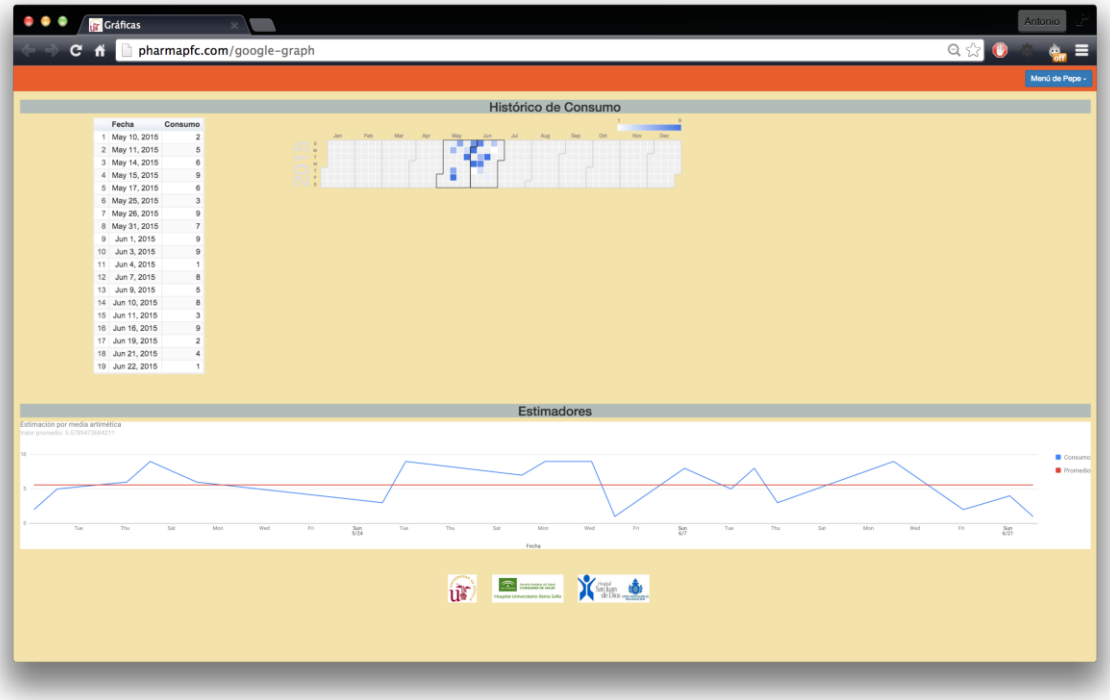


Figura 22. Histórico de consumo de un determinado fármaco.

6.1.4 Panel de control.

Éste módulo sólo estará presente para aquellos usuarios que tengan los privilegios de administrador de sistemas. Será accesible desde el menú desplegable de la esquina superior.

En él, el usuario podrá interactuar directamente con la base de datos a través de cuatro formularios independedientes entre sí. Mediante dichos formularios, el administrados podrá realizar cuatro acciones que son: añadir y eliminar usuarios, y añadir y eliminar fármacos.

Bastará con rellenar todos los campos del formulario en cuestión y pulsar sobre el botón correspondiente. Sólo habrá que rellenar el formulario para el cuál queremos realizar la acción. Es decir, si por ejemplo queremos añadir un nuevo fármaco a la base de datos, sólo hay que rellenar el tercer formulario (el correspondiente a *"Añadir fármaco"*) el resto podrán ir vacíos. Si se rellenan dos o más formularios, se realizarán también las acciones correspondientes a dichos formularios aunque el botón que se pulse no sea el correspondiente.

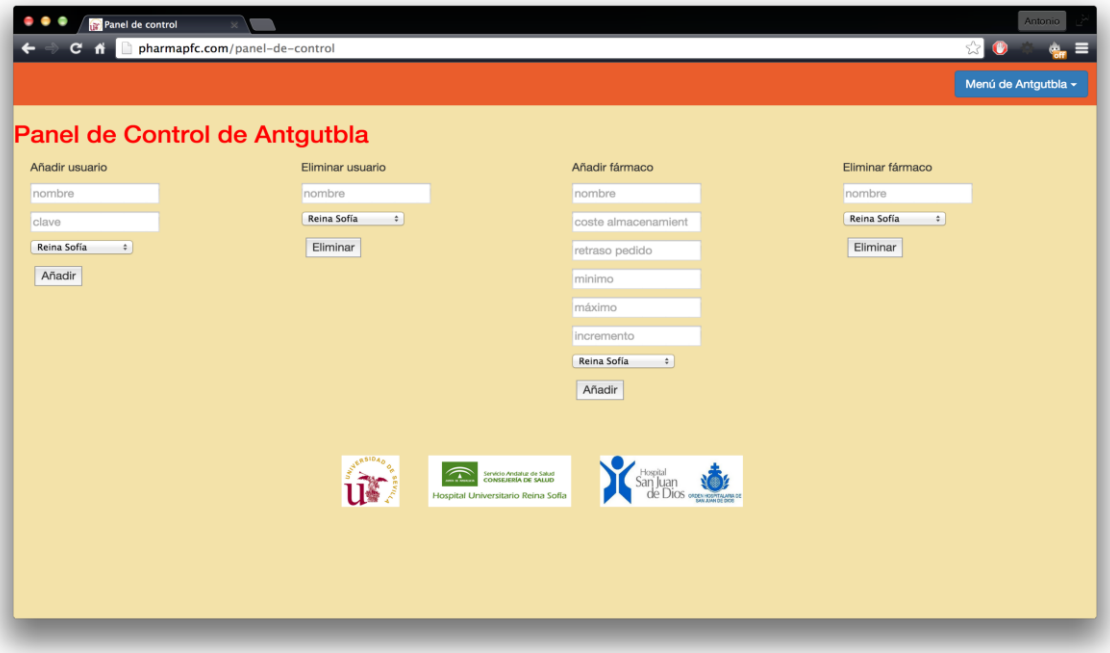


Figura 23. Panel de control de un administrador.

6.1.5 Subida de ficheros.

Para agilizar la tarea de insertar las cantidades consumidas de un determinado fármaco en muchas fechas a lo largo del tiempo, se ha visto necesaria la implementación de algún algoritmo que se encargo de automatizar el proceso de

guardado en base de datos, de modo que el usuario no tenga que introducir uno a uno estos consumos desde la pantalla principal de la aplicación.

Para que el citado módulo funcione correctamente, es necesario seguir las instrucciones que se explican a continuación. Si no se tiene en cuenta el formato requerido para el fichero a cargar, la subida se realizará con éxito pero los consumos no serán correctamente almacenados en la base de datos.
En la figura 24 se puede ver un ejemplo de fichero válido para la carga de datos. Como se puede observar, se insertará una línea por cada día en el que se haya registrado un cosumo mayor que cero del fármaco. A continuación, y a modo de separación entre la fecha y el consumo, se debe insertar una tabulación (\t) y finalmente, el valor del consumo.

Existe una gran diversidad de formatos que puede adoptar la fecha debido al preprocesado que se realiza posteriormente con PHP. Dichos formatos vienen definidos en el estándar del lenguaje de programación PHP y se pueden consultar en el siguiente enlace [XX](http://php.net/manual/es/datetime.formats.date.php)[4]. Cualquier otra fecha que no responda a alguno de los formatos ahí definidos causará incoherencias en la base de datos.

Una vez estemos seguros que el fichero posee el formato correcto, tan solo tenemos que pulsar sobre el botón de *"choose file"*, seleccionar el fichero de entre nuestros archivos, y pulsar en el botón *"Subir fichero*". A continuación nos aparecerá una ventana de control donde veremos si la subida ha fallado o no.

Para prevenir posibles errores en la subida de ficheros, éste modulo sólo es accesible para aquellos usuarios que dispongan de permisos de administrador del sistema.

[4] Comúnmente se suele emplear el estándar de la norma ISO 8601 para la representación de fechas. Dicho formato responde a la forma habitual YYYY/mm/dd

1/5/12	-3
1/10/12	-6
1/21/12	-7
2/1/12	-6
2/1/12	-9
2/7/12	-5
2/10/12	-7
2/16/12	-1
2/20/12	-9
2/24/12	-6
3/14/12	-3
3/16/12	-4
3/20/12	-3
3/29/12	-2

Figura 24. Formato válido para subida automática de consumos.

6.1.6 Cálculo del vector de pedido óptimo.

Bien se podría denominar el bloque objetivo del proyecto, ya que esta plataforma se ha desarrollado mayoritariamente para implementar esta función.

En esta ventana, el usuario podrá realizar el cálculo del vector de pedido óptimo para un determinado medicamento. Para ellos, deberá seleccionar primero un método de estimación de entre los desarrollados. A continuación, se selecciona el fármaco del cuál se quiere hacer el pedido. Finamente, se hace click en el botón de calcular para que nos muestre los resultados de la ejecución del algoritmo de optimización.

Si la ejecución se ha llevado a cabo con éxito y se ha optenido un vector de pedido válido, el pograma representará dicho vector en una gráfica como la de la figura XX. En la interpretación de la gráfica es la siguiente: en el eje x se representa la fecha en al que habrá que pedir las unidades correspondientes indicadas en el eje y. En azul se ha representado las unidades que se tienen que pedir para cumplir con las especificaciones de pedidos mínimos y días en el horizonte. En rojo se han representado las unidades a pedir para realizar un pedido óptimo, la diferencia es que dicho vector de pedido no tiene por qué cumplir con las especificaciones requeridas por el personal del hospital.

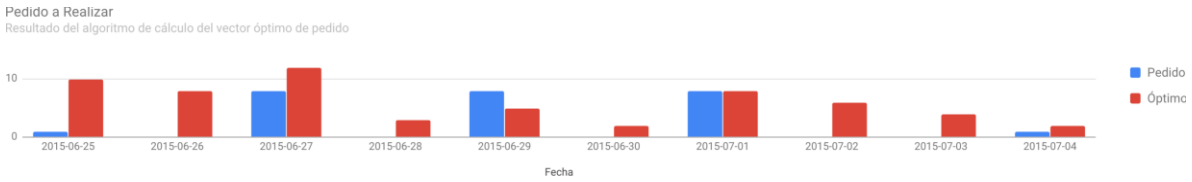


Figura 25. Representación del vector de pedido y véctor de pedido óptimo.

8. BIBLIOGRAFÍA

[1] Andy Harris: Programación con PHP6 y MySQL. Editorial ANAYA. Febrero 2009.

[2] PHP: http://php.net/

[3] Google Charts: https://developers.google.com/chart/

[4] W3School: http://www.w3schools.com/

[5] Web App schema: http://blog.tmimgcdn.com/wp-content/uploads/2013/07/gui.jpg?9d7bd4

[6] HTML: https://es.wikipedia.org/wiki/HTML , http://html.net/

[7] Responsive Design: http://www.purelybranded.com/wp-content/uploads/2012/09/responsive-web-design-a-quick-guide.gif

[8] Bootstrap grid: http://bootstrapbay.com/blog/wp-content/uploads/2014/09/bootstrap-grid-system.jpg

[9] GEO tutorial: Pronóstico de Demanda con Alisamiento Exponencial para distintos valores de Alfa . http://www.gestiondeoperaciones.net/proyeccion-de-demanda/pronostico-de-demanda-con-alisamiento-exponencial-para-distintos-valores-de-alfa/

[10] Alberto López Ramírez: Optimización de Estimación de la Demanda en Servicio Farmacéutico Hospitalario. Proyecto Fin de Carrera ESI. Marzo 2014.

[11] José López Quijado. Domine PHP y MySQL. 2º Edición 2010.

[12] César Hoyos Martín. Optimización de invertio en Farmacia Hospitalaria. Proyecto Fin de Carrea ESI. Junio 2015.

[12] JQuery: Ajax request. http://api.jquery.com/category/ajax/