

**UNIVERSIDAD TÉCNICA DE MACHALA**

**UNIDAD ACADÉMICA DE INGENIERÍA CIVIL**

**CARRERA DE INGENIERÍA DE SISTEMAS**

**TRABAJO DE TITULACIÓN**

**PREVIO A LA OBTENCIÓN DEL TÍTULO DE**

**INGENIERO DE SISTEMAS**

**TITULO:**

**“DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB INTEGRADA AL PROGRAMA PERSONAL COMMUNICATIONS ISERIES ACCESS ORIENTADA A LA GESTIÓN DEL SISTEMA COMERCIAL (SICO-CNEL) PARA LA AUTOMATIZACIÓN DE PROCESOS MANUALES DE CONIEL CIA.LTDA”**

**AUTOR:**

**SR. JHONSSON XAVIER CÓRDOVA DÁVILA**

**TUTOR:**

**ING. REDROVAN CASTILLO FAUSTO**

**CO-TUTOR:**

**ING. CARTUCHE CALVA JOFFRE**

**MACHALA - EL ORO – ECUADOR**

**2014**

**AGRADECIMIENTO**

En primer lugar doy gracias a Dios, por haberme dado fuerza y valor para culminar esta etapa de mi vida.

Agradezco también la confianza y el apoyo brindado por parte de mis padres, que sin duda alguna en el trayecto de mi vida me han demostrado su amor, corrigiendo mis fallas y celebrando mis triunfos.

A mi mami Livia, que con su demostración de una segunda madre para mí me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.

Al Ingeniero Fausto Redrovan por toda la colaboración brindada, durante la elaboración de este proyecto.

Gracias a todas las personas que ayudaron directa e indirectamente en la realización de este proyecto.

Jhonsson Córdova

**DEDICATORIA**

Dedico este trabajo principalmente a Dios, el que me ha dado fortaleza para continuar cuando a punto de caer he estado.

De igual manera a mis padres, que han sabido formarme con buenos sentimientos, hábitos y valores, lo cual me ha ayudado a salir adelante en los momentos más difíciles.

A mi familia que siempre ha estado junto a mí brindándome su apoyo.

Jhonsson Córdova

**INTRODUCCIÓN**

Actualmente los procesos dentro de una empresa deben estar optimizados con el fin de proveer respuestas eficientes. Es por esto que el presente trabajo de investigación pretende dar solución a un sin número de problemas que tiene la empresa CONIEL CIA.LTDA con el manejo de sus procesos que actualmente son llevados de manera manual.

Dichos procesos conllevan a la repetición de varios pasos, al sobrecargo de recursos, tiempo, etc.

Por lo que se hará uso de tecnologías de desarrollo web para la creación una aplicación que permita la simplificación de recursos, tiempo y pasos para la ejecución de un proceso.

**Objetivos**

**Objetivos General**

Implementar una aplicación web integrada al programa PERSONAL COMMUNICATIONS ISERIES ACCESS orientada a la gestión del sistema comercial (SICO-CNEL) para la automatización de procesos manuales de CONIEL CIA.LTDA utilizando la metodología UWE en conjunto con el Framework Django.

**Objetivos Específicos**

* Recopilar y analizar los requisitos y necesidades que forman parte de los procesos de tratamiento y administración de información en Coniel Cia. Ltda., a través de reuniones, diálogos y encuestas realizadas al personal encargado, con el fin de determinar un punto de inicio para el desarrollo de la aplicación.
* Diseñar las interfaces de usuario en función a los requisitos previamente sintetizados, mediante herramientas de diseño para la determinación de la estructura base de la aplicación.
* Codificar la funcionalidad de las interfaces tomando como punto de partida los prototipos y estructura, con el fin de constituir una aplicación dinámica y funcional, utilizando frameworks para desarrollo de aplicaciones web.
* Realizar pruebas de funcionamiento a la aplicación, utilizando herramientas de testeo de aplicaciones web, con la finalidad de retroalimentar el código escrito para lograr una aplicación robusta y estable.
* Implementar un servidor web interno mediante la utilización de tecnologías actuales para la puesta en producción de la aplicación.

**Hipótesis**

A través de la implementación de una aplicación web se logrará optimizar recursos, minimizar el tiempo de respuesta, disminuir la repetición de pasos en cada proceso, llevar un mejor control de las actividades realizadas, garantizar confiabilidad de información, fácil acceso, interfaz intuitiva, etc.

**CAPÍTULO I**

# MARCO REFERENCIAL

## Justificación

Actualmente los sistemas migran a la web por su facilidad de acceso, usabilidad e interacción con el usuario, con esto se logra un mejor desempeño en las aplicaciones ya que se reduce el uso de los recursos de hardware e incrementa el tiempo de vida de la misma en función de un solo equipo.

El desarrollo de aplicaciones web se realiza a través de tecnologías específicas de tal manera que se garantice mejores resultados que una aplicación de escritorio, tomando en cuenta los factores de tiempo de respuesta, trabajo en conjunto y disponibilidad. La compañía CONIEL CIA.LTDA realiza trabajos relacionados a la ingeniería eléctrica, por lo que hace uso del sistema actual que posee una serie de desventajas que no ayudan a obtener un máximo rendimiento en el desarrollo de los procesos.

Por este motivo la empresa necesita de implementar una aplicación web que garantice la optimización de los procesos, la disminución del tiempo de ejecución de las tareas y el fácil acceso.

## Problemas

CONIEL CIA.LTDA es una empresa dedicada a ejecutar trabajos de control de perdidas adjudicados por CNEL EP Unidad de Negocio El Oro a través del portal de compras públicas Sercop (www.compraspublicas.gob.ec), la misma encargada de gestionar información relevante del contrato al Sistema Comercial SICO, de la cual se debe guardar un registro en un sistema local para su posterior facturación, dicho sistema, es manual y es controlado por una persona.

Los contratos de Pérdidas y Control de Energía se deben llevar con un manejo de información y fotos estricto, ya que estos datos se entregan periódicamente de forma digital a CNEL EP, las fotos son organizadas en carpetas por código de abonado y la información es tabulada con la ayuda de archivos de Excel que facilitan el filtrado de información. Las actividades principales que se realizan en estos contratos son: cambios de medidores, mantenimientos, servicios nuevos y consultas de información de usuarios y medidores. El Sistema Comercial (SICO), es un sistema distribuido con un servidor AS/400 que trabaja bajo una terminal IBM Personal Communications, que a coste de brindar rapidez en tiempos de respuesta carece de una interfaz orientada al usuario final. El sistema SICO posee una complejidad muy alta a la hora de realizar una tarea, ya que las mismas se componen de varios pasos que deben ser ejecutados sin falta y en secuencia determinada, por lo que el usuario final debe someterse a capacitaciones extensas y que la mayoría de veces no brindan los resultados que se espera.

## Preguntas Científicas

* ¿Qué técnicas de investigación utilizar para la recopilación de requisitos?
* ¿Cómo realizar un documento de especificación de requisitos para diseñar interfaces gráficas en función de las necesidades del usuario?
* ¿Qué tecnologías utilizar para desarrollar aplicaciones web dinámicas y funcionales?
* ¿A través de que herramientas se podría realizar una gestión de pruebas a una aplicación web?
* ¿Cómo hacer uso de nuevas tecnologías para optimizar el acceso de la información?

## Alcance

Con el fin de automatizar los procesos que la empresa CONIEL Cia.Ltda emplea al habérsele adjudicado determinado proceso de control de pérdidas, se pretende construir un completo sistema administrable que gestione las actividades que son realizadas por el personal operativo (cambios de medidor, cambios de materiales, servicios nuevos y revisiones), así como también por el personal de oficina (entrega de reportes, manejo y administración de información, fotos y digitalizaciones) de tal manera que toda información requerida por determinado contrato esté disponible al menor tiempo posible para la empresa contratante (CNEL EP), para lo cual se implementará integración al programa Personal Communications de IBM que es un cliente para el Sistema SICO-CNEL en el cual se debe registrar toda información de actividades realizadas por la empresa. Todo lo anteriormente descrito será analizado e implementado a través de interfaz web con el framework DJANGO además se diseñara un esquema de base de datos POSTGRESQL que funcionará en el mismo servidor que la aplicación, el mismo que será adecuado y preparado para la puesta a producción al igual que la intranet y la capacitación al personal.

**CAPÍTULO II**

# MARCO TEÓRICO

## Antecedentes Históricos de la Empresa

### Reseña Histórica de la Compañía de Construcciones e Instalaciones Eléctricas CONIEL CIA.LTDA

La compañía de Construcciones e Instalaciones Eléctricas CONIEL CIA.LTDA, tiene sus inicios debido a la gran demanda de servicios eléctricos en nuestra Provincia. Fue constituida el 02 de julio del 2002, resultado de la asociación entre el Tnlg. Julio Loaiza y el Sr. Luis Pérez, para de esta manera dar paso al crecimiento de la empresa Privada cuyo único objetivo era ganar experiencia en el ámbito profesional y brindar servicios de calidad a la ciudadanía. Con el pasar de los años, la compañía es fuertemente constituida y cuenta con un amplio número de contratos firmados con la CNEL EP, los mismos que conforman su experiencia profesional. La empresa ha cumplido a cabalidad con las actividades de la prestación de los servicios adjudicado con la CNEL EP. La Compañía ha ido creciendo y ganando terreno en la Provincia y haciéndose acreedora a muy buenas referencias de trabajo por parte de la CNEL EP.

### Ubicación



**CONIEL CIA.LTDA**

*Figura 1*. Croquis de ubicación de la Empresa Coniel Cia. Ltda

***Fuente:*** https://www.google.com.ec/maps/@-3.2597728,-79.9635986,18z?hl=es-419

**Dirección:** Machala, Santa Rosa e/ Arízaga y Gral. Manuel Serrano.  
**Teléfono:** 2938-581

### Organigrama Estructural

**ORGANIGRAMA DE FUNCIONES**

**GERENTE**

(Planificación control en la ejecución de los trabajos)

**ABOGADO**

Persona encargada de los reglamentos, normas estatales que permiten la eficiencia y constitución de la misma.

**CONTADOR**

Ente regulador de las normas tributarias y responsable del control de la compañía.

**REPRESENTANTE TÉCNICO**

(Fiscalizador de los Trabajos a ejecutar)

**SECRETARIA**

(Recepción de información y transferencia de información, novedades, reportes de las mismas)

**DIGITADORES**

(Recepción de información, digitación a través del sistema SICO novedades, reportes de las mismas)

**SUPERVISORES**

(Enrutamiento de servicio, control del personal y de las acciones a cumplir de la gestión.)

**PERSONAL OPERATIVO**

**DE ESTA MANERA SE INTEGRAN LOS GRUPOS DE TRABAJOS**

**GRUPO 1**

**(4 personas)**

**(4 )**

**Bachilleres en Electricidad para ejecutar los trabajos de:**

* Cambio de Medidores - Acometidas
* Corte y Reconexión de Energía Eléctrica, etc.

Herramientas: Un radio para comunicación, una escalera, cinturones, guantes, cascos, conos, cable, pinzas, destornilladores, playo etc.

**Vehículo**

Cámara fotográfica, equipo de comunicaciones.

***Fuente:*** Imagen proporcionada por CONIEL CIA.LTDA

*Figura 2.* Estructura Organizacional de Coniel Cia.Ltda

## Antecedentes Conceptuales

### Aplicación Web

Es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (servidor web) y el protocolo mediante el que se comunica (HTTP) están estandarizados y no han de ser creados por el programados de aplicaciones.

Los entornos web en los que se ejecutan las aplicaciones web pueden ser: intranet y extranet. (Programación de aplicaciones web, 2012, pág. 47)

#### Ventajas de las aplicaciones web

* No requiere instalar software especial (en los clientes). En esencia, para acceder a un software web solo necesitamos disponer de un navegador de páginas web (Internet Explorer, Firefox, Opera, Chrome, etc.), los cuales suelen venir con el propio sistema operativo. No es necesario tener nada más. Debido a la arquitectura de las aplicaciones web, el navegador suele quedar relegado a mostrar la interfaz de usuario (menús, opciones, formularios, etc.), mientras que toda la compleja lógica de negocio se lleva en el lado del servidor.
* Bajo coste en actualizar los equipos con una nueva versión. Los navegadores web visualizan las páginas web que son servidas por el servidor web dinámicamente. En ese sentido, es el servidor quien ejecuta la mayor parte del código de la aplicación y suministra de forma centralizada las vistas (las páginas) a los navegadores conectados. En consecuencia, no hay que instalar nada en los puestos de trabajo, ya que la actualización se realiza en el servidor y automáticamente la ven todos los usuarios.
* Acceso a la última y mejor versión. Como consecuencia del punto anterior, se evita que pueda existir algún equipo que ejecute una versión diferente y desactualizada. Si existen ordenadores con distintas versiones del programa se pueden originar problemas de consistencia en la información o pérdida de funcionalidad.
* Información centralizada. En una aplicación web, no solamente la lógica de negocio está centralizada en el servidor, sino también los datos que se ubican en una base de datos centralizada (en ese servidor u otro destinado a tal fin). La centralización tiene la ventaja de facilitar el acceso a la misma.
* Seguridad y copias de seguridad. Este es un corolario del punto anterior, es decir, una consecuencia. Como disponemos de los datos centralizados es más fácil establecer y llevar el control de una política de copias de seguridad centralizada. Es más, al no ubicarse los datos en el puesto de trabajo, en caso de robo o incendio, la empresa no ha perdido información y puede desplegar rápidamente un nuevo puesto de trabajo (PC con un navegador web).
* Movilidad. Este es un concepto relativo y dependiente de la implantación concreta. Si el software está ubicado en un servidor web en Internet o bien disponemos de una intranet externalizada (extranet), cualquier usuario con un portátil y una conexión a Internet móvil podría acceder a la aplicación.
* Reducción de costes en los puestos cliente (mayor longevidad). Debido a que las páginas se ofrecen desde el servidor web (donde se suelen ejecutar la mayoría de los procesos y la lógica de negocio), el equipo cliente queda relegado a mostrar los resultados y formularios, para lo cual no es necesario un hardware potente en los puestos de trabajo, lo que se traduce en reducción de costes y una mayor longevidad en el uso de los mismos (no hay que cambiar el hardware de los puestos porque ahora se requieran operaciones más complejas).

#### Navegadores (browsers)

Permiten acceder a la web y visualizar en modo gráfico documentos

HTML (XHTML). Procesan también otros tipos de objetos: imágenes, sonidos, videos, scripts, etc.

* Pueden arrancar aplicaciones que traten los ficheros recibidos.
* Aceptan la instalación de plugins (módulos con funcionalidad extra)
* Para procesar ciertos tipos de documentos (p.ej, PDF)
* Presentaciones flash

**Navegadores más populares:**

* WorldWideWeb (Tim Berners-Lee, 1991)
* Mosaic (NCSA, 1993)
* Mozilla (1998)
  + Netscape Navigator (2000): Introduce JavaScript en la v2 (1995)
  + Firefox (2002)
* Microsoft Internet Explorer (1995): solo en Windows
* Safari (2003): en Mac OS X [Apple llegó bien tarde a la Web…]
* Chrome (2008): multiplataforma
* Opera (1996): el que más tecnologías soporta, multiplataforma. (Pavón Maestras, 2012)

### ¿Qué es Python ?

Es un lenguaje interpretado, interactivo y orientado a objetos que ofrece una gran cantidad de estructuras de datos de alto nivel por medio de un tipado dinámico y fuerte, además de estas características es multiparadigma y multiplataforma. Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. (Python para todos, 2010, pág. 7)

#### Ventajas de Python

* Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. (Python llega a ser considerado por muchos un lenguaje de programación de muy alto nivel.)
* Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación.
* Python ofrece un entorno interactivo que facilita la realización de pruebas y ayuda
* a despejar dudas acerca de ciertas características del lenguaje.
* El entorno de ejecucion de Python detecta muchos de los errores de programación
* que escapan al control de los compiladores y proporciona información muy rica
* para detectarlos y corregirlos.
* Python puede usarse como lenguaje imperativo procedimental o como lenguaje
* orientado a objetos.
* Posee un rico juego de estructuras de datos que se pueden manipular de modo
* Sencillo.

#### Elementos del Lenguaje de Python

Como en la mayoría de los lenguajes de programación de alto nivel, en Python se compone de una serie de elementos que alimentan su estructura. Entre ellos, podremos encontrar los siguientes:

##### Variables

Una variable es un espacio para almacenar datos modificables, en la memoria de un ordenador. En Python, una variable se define con la sintaxis:

nombre\_de\_la\_variable = valor\_de\_la\_variable

Cada variable, tiene un nombre y un valor, el cual define a la vez, el tipo de datos de la variable.

Existe un tipo de *variable*, denominada constante, la cual se utiliza para definir valores fijos, que no requieran ser modificados.

Utilizar nombres descriptivos y en minúsculas. Para nombres compuestos, separar las palabras por guiones bajos. Antes y después del signo =, debe haber uno (y solo un) espacio en blanco.

# Correcto

mi\_variable = 12

 # Incorrectos

MiVariable = 12

mivariable = 12

mi\_variable=12

mi\_variable = 12

Utilizar nombres descriptivos y en mayúsculas separando palabras por guiones bajos. Ejemplo:

MI\_CONSTANTE = 12

Para **imprimir un valor en pantalla**, en Python, se utiliza la palabra clave print:

mi\_variable = 15

print mi\_variable

Lo anterior, imprimirá el valor de la variable mi\_variable en pantalla. (Python para principiantes, 2012, pág. 30)

##### Tipo de Datos

Una variable (o constante) puede contener valores de diversos tipos. Entre ellos:

Cadena de texto (*string*):

mi\_cadena = "Hola Mundo!"

 mi\_cadena\_multilinea = """

Esta es una cadena

de varias líneas

"""

Número entero:

edad = 35

Número entero octal:

edad = 043

Número entero hexadecimal:

edad = 0x23

Número real:

precio = 7435.28

Booleano (verdadero / Falso):

verdadero = True

falso = False (Python para principiantes, 2012, pág. 31)

##### Operadores Aritméticos

Entre los operadores aritméticos que Python utiliza, podemos encontrar los siguientes: (Python para principiantes, 2012, pág. 32)

| **Símbolo** | **Significado** | **Ejemplo** | **Resultado** |
| --- | --- | --- | --- |
| + | Suma | a = 10 + 5 | a es 15 |
| - | Resta | a = 12 - 7 | a es 5 |
| - | Negación | a = -5 | a es -5 |
| \* | Multiplicación | a = 7 \* 5 | a es 35 |
| \*\* | Exponente | a = 2 \*\* 3 | a es 8 |
| / | División | a = 12.5 / 2 | a es 6.25 |
| // | División entera | a = 12.5 / 2 | a es 6.0 |
| % | Módulo | a = 27 % 4 | a es 3 |

*Tabla 1.* Operadores Aritméticos de Python

***Fuente:*** Libro Python para principiantes

#### Colecciones

##### Listas

La lista es un tipo de colección ordenada. Sería equivalente a lo que en otros lenguajes se conoce por arrays, o vectores.

Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos, y también listas. Crear una lista es tan sencillo como indicar entre corchetes, y separados por comas, los valores que queremos incluir en la lista:

l = [22, True, “una lista”, [1, 2]]

Podemos acceder a cada uno de los elementos de la lista escribiendo el nombre de la lista e indicando el índice del elemento entre corchetes. Ten en cuenta sin embargo que el índice del primer elemento de la lista es 0, y no 1:

l = [11, False]

mi\_var = l[0] # mi\_var vale 11

Si queremos acceder a un elemento de una lista incluida dentro de otra lista tendremos que utilizar dos veces este operador, primero para indicar a qué posición de la lista exterior queremos acceder, y el segundo para seleccionar el elemento de la lista interior:

l = [“una lista”, [1, 2]]

mi\_var = l[1][0] # mi\_var vale 1

También podemos utilizar este operador para modificar un elemento de la lista si lo colocamos en la parte izquierda de una asignación: (Python para todos, 2010, págs. 22-23)

l = [22, True]

# Ahora l valdrá [99, True]

l[0] = 99

##### Tuplas

Todo lo que hemos explicado sobre las listas se aplica también a las tuplas, a excepción de la forma de definirla, para lo que se utilizan paréntesis en lugar de corchetes.

t = (1, 2, True, “python”)

En realidad el constructor de la tupla es la coma, no el paréntesis, pero el intérprete muestra los paréntesis, y nosotros deberíamos utilizarlos, por claridad.

>>> t = 1, 2, 3

>>> type(t)

type “tuple”

Además hay que tener en cuenta que es necesario añadir una coma para tuplas de un solo elemento, para diferenciarlo de un elemento entre paréntesis.

>>> t = (1)

>>> type(t)

type “int”

>>> t = (1,)

>>> type(t)

type “tuple”

Para referirnos a elementos de una tupla, como en una lista, se usa el operador []:

mi\_var = t[0] # mi\_var es 1

mi\_var = t[0:2] # mi\_var es (1, 2)

Podemos utilizar el operador [] debido a que las tuplas, al igual que las listas, forman parte de un tipo de objetos llamados secuencias. Permitirme un pequeño inciso para indicaros que las cadenas de texto también son secuencias, por lo que no os extrañará que podamos hacer cosas como estas:

c = “hola mundo”

c[0] # h

c[5:] # mundo

c[::3] # hauo (Python para todos, 2010, pág. 24)

##### Diccionarios

Los diccionarios, también llamados matrices asociativas, deben su nombre a que son colecciones que relacionan una clave y un valor. Por ejemplo, veamos un diccionario de películas y directores:

d = {“Love Actually “: “Richard Curtis”,

“Kill Bill”: “Tarantino”,

“Amélie”: “Jean-Pierre Jeunet”}

El primer valor se trata de la clave y el segundo del valor asociado a la clave. Como clave podemos utilizar cualquier valor inmutable: podríamos usar números, cadenas, booleanos, tuplas, pero no listas o diccionarios, dado que son mutables. Esto es así porque los diccionarios se implementan como tablas hash, y a la hora de introducir un nuevo par clave-valor en el diccionario se calcula el hash de la clave para después poder encontrar la entrada correspondiente rápidamente. Si se modificara el objeto clave después de haber sido introducido en el diccionario, evidentemente, su hash también cambiaría y no podría ser encontrado.

La diferencia principal entre los diccionarios y las listas o las tuplas es que a los valores almacenados en un diccionario se les accede no por su índice, porque de hecho no tienen orden, sino por su clave, utilizando de nuevo el operador [].

d[“Love Actually “] # devuelve “Richard Curtis”

Al igual que en listas y tuplas también se puede utilizar este operador para reasignar valores.

d[“Kill Bill”] = “Quentin Tarantino”

Sin embargo en este caso no se puede utilizar slicing, entre otras cosas porque los diccionarios no son secuencias, si no mappings (mapeados, asociaciones). (Python para todos, 2010, págs. 25-26)

#### Control de Flujo

Las sentencias de control es uno de los primeros aspectos que deben ser abordados durante el aprendizaje de un lenguaje de programación. Entre las sentencias de las que dispone Python, las básicas son las que permiten crear condiciones y realizar iteraciones. Las principales sentencias de control son las siguientes:

##### If, else y elif

La sentencia *if/else* funciona evaluando la condición indicada, si el resultado es True se ejecutara la siguiente sentencia o sentencias, en caso negativo se ejecutara las sentencias que aparecen a continuación del *else.* Python utiliza la indentación para establecer sentencias que pertenecen al mismo bloque. Además, en el carácter dos puntos (:) indica el comienzo del bloque. Ejemplo:

X = 4

Y = 0

if x == 4:

y = 5

else:

y = 2

Obviamente, también es posible utilizar solo la sentencias *if* para comprobar si se cumple una determinada condición y actuar en consecuencia. Además, podemos anidar diferentes niveles de comprobación a través de *elif:* (Python3 al descubierto, 2014, pág. 52)

if x == 4:

y = 1

elif x == 5:

y = 2

elif x == 6:

y = 3

else:

y = 5

##### For

Para iterar contamos con dos sentencias que ayudaran a crear bucles, nos referimos a for y while. La primera de ellas aplica a una serie de sentencias sobre cada uno de los elementos que contiene el objeto sobre el que se aplicará la sentencia for. Python incorpora una función llamada range() que podemos utilizar para iterar sobre una serie de valores. Por ejemplo:

>>> for x in range(1 , 3):

… print ( x )

…

1

2

3

Es muy común iterar a través de *for* sobre los elementos de una tupla o de una lista:

>>> lista = [“uno”, “dos”, “tres”]

>>> cad = “”

>>> for ele in lista:

… cad += ele

…

>>> cad

“unodostres” (Python3 al descubierto, 2014, pág. 53)

### Introducción al Framework Django

Es un framework de desarrollo Web que ahorra tiempo y hace que el desarrollo Web sea divertido. Utilizando Django puedes crear y mantener aplicaciones Web de alta calidad con un mínimo esfuerzo.

En el mejor de los casos, el desarrollo web es un acto entretenido y creativo; en el peor, puede ser una molestia repetitiva y frustrante. Django te permite enfocarte en la parte creativa, la parte divertida de tus aplicaciones Web, al mismo tiempo que mitiga el esfuerzo de las partes repetitivas. De esta forma, provee un alto nivel de abstracción de patrones comunes en el desarrollo Web, atajos para tareas frecuentes de programación y convenciones claras sobre cómo solucionar problemas. Al mismo tiempo, Django intenta no entrometerse, dejándote trabajar fuera del ámbito del framework según sea necesario. (Django 1.0, 2010, pág. 21)

#### ¿Qué es Django?

Django es un **framework web** de código abierto escrito en Python que permite construir aplicaciones web **más rápido y con menos código**.

Fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD. Django se centra en automatizar todo lo posible y se adhiere al principio DRY (Don't Repeat Yourself). (Django 1.0, 2010, pág. 22)

#### Historia de Django

Django nació naturalmente de aplicaciones de la vida real escritas por un equipo de desarrolladores Web en Lawrence, Kansas. Nació en el otoño boreal de 2003, cuando los programadores Web del diario *Lawrence Journal-World*, Adrian Holovaty y Simon Willison, comenzaron a usar Python para crear sus aplicaciones. El equipo de The World Online, responsable de la producción y mantenimiento de varios sitios locales de noticias, prosperaba en un entorno de desarrollo dictado por las fechas límite del periodismo. Para los sitios incluidos LJWorld.com, Lawrence.com y KUsports.com los periodistas (y los directivos) exigían que se agregaran nuevas características y que aplicaciones enteras se crearan a una velocidad vertiginosa, a menudo con sólo días u horas de preaviso. Es así que Adrián y Simón desarrollaron por necesidad un framework de desarrollo Web que les ahorrara tiempo era la única forma en que podían crear aplicaciones mantenibles en tan poco tiempo.

En el verano boreal de 2005, luego de haber desarrollado este framework hasta el punto en que estaba haciendo funcionar la mayoría de los sitios World Online, el equipo de World Online, que ahora incluía a Jacob Kaplan-Moss, decidió liberar el framework como software de código abierto. Lo liberaron en julio de 2005 y lo llamaron Django, por el guitarrista de jazz Django Reinhardt. A pesar de que Django ahora es un proyecto de código abierto con colaboradores por todo el mundo, los desarrolladores originales de World Online todavía aportan una guía centralizada para el crecimiento del framework, y World Online colabora con otros aspectos importantes tales como tiempo de trabajo, materiales de marketing, y hosting/ancho de banda para el Web site del framework (http://www.djangoproject.com/).

Esta historia es relevante porque ayuda a explicar dos cuestiones clave. La primera es el "punto dulce" de Django. Debido a que Django nació en un entorno de noticias, ofrece varias características. La segunda cuestión a resaltar es cómo los orígenes de Django le han dado forma a la cultura de su comunidad de código abierto. Debido a que Django fue extraído de código de la vida real, en lugar de ser un ejercicio académico o un producto comercial, está especialmente enfocado en resolver problemas de desarrollo Web con los que los desarrolladores de Django se han encontrado y con los que continúan encontrándose. Como resultado de eso, Django es activamente mejorado casi diariamente. Los desarrolladores del framework tienen un alto grado de interés en asegurarse de que Django les ahorre tiempo a los desarrolladores, produzca aplicaciones que son fáciles de mantener y rindan bajo mucha carga. (Django 1.0, 2010, pág. 23)

#### Activar la interfaz de administración de Django

La interfaz de administración de Django es una de las características principales del framework, pero como no todos la necesitan, es una pieza opcional. Esto significa que hay que dar tres pasos para activar la interfaz de administración:

**1.** Agrega meta-datos de administración a tus modelos.

No todos los modelos pueden (o deberían) ser editables por los usuarios administradores, por lo que necesitas "marcar" los modelos que deberían tener una interfaz de administración. Esto lo hacemos añadiendo al modelo una clase interna Admin (junto con la clase Meta, si es que hay una). Así que, para agregar una interfaz de administración a nuestro modelo hacemos algo similar a lo siguiente:

class Book(models.Model):

title = models.CharField(maxlength=100)

authors = models.ManyToManyField(Author)

publisher = models.ForeignKey(Publisher)

publication\_date = models.DateField()

num\_pages = models.IntegerField(blank=True, null=True)

  def \_\_unicode\_\_(self):

return self.title

  class Admin:

pass

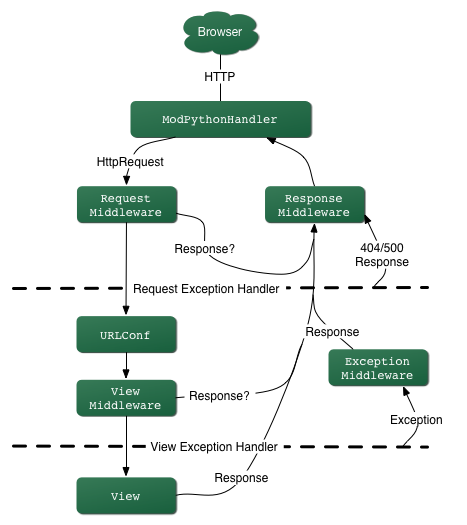
La declaración de Admin marca la clase como poseedora de una interfaz de administración. Hay una serie de opciones que podemos incluir bajo Admin, pero por ahora se limitará al comportamiento por defecto, así que se escribe pass para decirle a Python que la clase Admin está vacía.

1. Instalar la aplicación admin. Esto se hace agregando "django.contrib.admin" a tus INSTALLED\_APPS de su archivo de configuración settings.py.
2. Además, asegúrese de que las aplicaciones "django.contrib.sessions", "django.contrib.auth", y "django.contrib.contenttypes" no están comentadas, ya que la aplicación admin depende de ellas.
3. También descomente todas las líneas de MIDDLEWARE\_CLASSES configurando la tupla, y borre la definición de TEMPLATE\_CONTEXT\_PROCESSOR para permitir que tome los valores por defecto.
4. Ejecute python manage.py syncdb. Este paso instalará las tablas de la base de datos que la interfaz de administración necesita. (Django 1.0, 2010, pág. 87)

#### ¿Cómo procesa una petición Django?

Además del mapeo directo de URLs con funciones vista, Django nos provee un poco más de flexibilidad en el procesamiento de peticiones.

El flujo típico resolución de URLconf a una función de vista que retorna un HttpResponse puede ser corto-circuitado o *augmented* mediante middleware.



*Figura 3.* El flujo completo de una petición y respuesta Django

*Fuente: http:*//librosweb.es/libro/django\_1\_0

Cuando llega una petición HTTP desde el navegador, un *manejador* específico a cada servidor construye la HttpRequest, para pasarla a los componentes y maneja el flujo del procesamiento de la respuesta. El manejador luego llama a cualquier middleware de Petición o Vista disponible. Estos tipos de middleware son útiles para ***augmenting*** los objetos HttpRequest así como también para proveer manejo especial a determinados tipos de peticiones. En el caso de que alguno de los mismos retornara un HttpResponse la vista no es invocada.

Si una función de vista lanza una excepción, el control pasa al middleware de Excepción. Si este middleware no retorna un HttpResponse, la excepción se vuelve a lanzar. Django incluye vistas por omisión para respuestas amigables a errores 404 y 500.

Finalmente, el *middleware de respuesta* es bueno para el procesamiento posterior a un HttpResponse justo antes de que se envíe al navegador o haciendo una limpieza de recursos específicos a una petición. (Django 1.0, 2010, pág. 36)

#### Sistema de plantillas de Django

Una plantilla de Django es una cadena de texto que pretende separar la presentación de un documento de sus datos. Una plantilla define rellenos y diversos bits de lógica básica (esto es, etiquetas de plantillas) que regulan cómo debe ser mostrado el documento. Normalmente, las plantillas son usadas para producir HTML, pero las plantillas de Django son igualmente capaces de generar cualquier formato basado en texto. Vamos a sumergirnos en una simple plantilla de ejemplo:

<html>

<head><title>Ordering notice</title></head>

<body>

 <p>Dear {{ person\_name }},</p>

 <p>Thanks for placing an order from {{ company }}. It's scheduled to

ship on {{ ship\_date|date:"F j, Y" }}.</p>

 <p>Here are the items you've ordered:</p>

 <ul>

{% for item in item\_list %}

<li>{{ item }}</li>

{% endfor %}

</ul>

 {% if ordered\_warranty %}

<p>Your warranty information will be included in the packaging.</p>

{% endif %}

 <p>Sincerely,<br />{{ company }}</p>

 </body>

</html>

Esta plantilla es un HTML básico con algunas variables y etiquetas de plantillas agregadas. Explicando paso a paso:

* Cualquier texto encerrado por un par de llaves (por ej. {{ person\_name }}) es una *variable*. Esto significa "insertar el valor de la variable a la que se dio ese nombre".
* Cualquier texto que esté rodeado por llaves y signos de porcentaje (por ej. {% if ordered\_warranty %}) es una *etiqueta de plantilla*. Este ejemplo de plantilla contiene dos etiquetas: la etiqueta {% for item in item\_list %} (una etiqueta for) y la etiqueta {% if ordered\_warranty %} (una etiqueta if).

Una etiqueta for actúa como un simple constructor de bucle, dejándote recorrer a través de cada uno de los items de una secuencia. Una etiqueta if, como quizás esperabas, actúa como una cláusula lógica "if". En este caso en particular, la etiqueta comprueba si el valor de la variable ordered\_warranty se evalúa como True. Si lo hace, el sistema de plantillas mostrará todo lo que hay entre {% if ordered\_warranty %} y {% endif %}. Si no, el sistema de plantillas no mostrará esto. El sistema de plantillas también admite {% else %} y otras varias cláusulas lógicas.

* Finalmente, el segundo párrafo de esta plantilla tiene un ejemplo de un *filtro*, con el cual puedes alterar la exposición de una variable. En este ejemplo, {{ ship\_date|date:"F j, Y" }}, estamos pasando la variable ship\_date por el filtro date, tomando el filtro date el argumento "F j, Y". El filtro date formatea fechas en el formato dado, especificado por ese argumento. Los filtros se encadenan mediante el uso de un carácter pipe (|), como una referencia a las tuberías de Unix. (Django 1.0, 2010, pág. 46)

### ¿Qué es Postgres?

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS,s) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema:

* clases
* herencia
* tipos
* funciones

Otras características aportan potencia y flexibilidad adicional:

* Restricciones (Constraints)
* Disparadores (triggers)
* Reglas (rules)
* Integridad transaccional. (Gibert Ginestà & Pérez Mora, 2012)

#### Breve historia de Postgres

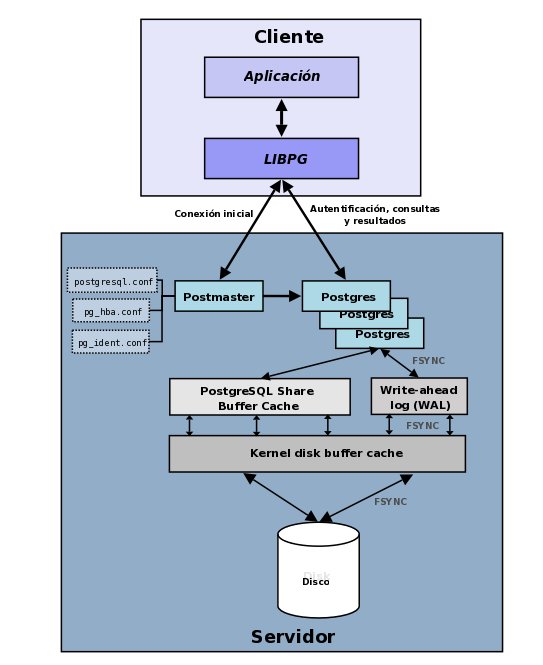
El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL (y brevemente llamado Postgres95) está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python). (Gibert Ginestà & Pérez Mora, 2012)

#### PostgreSQL

En 1996, se hizo evidente que el nombre “Postgres95” no resistiría el paso del tiempo. Elegimos un nuevo nombre, PostgreSQL, para reflejar la relación entre el Postgres original y las versiones más recientes con capacidades SQL. Al mismo tiempo, hicimos que los números de versión partieran de la 6.0, volviendo a la secuencia seguida originalmente por el proyecto Postgres.

Durante el desarrollo de Postgres95 se hizo hincapié en identificar y entender los problemas en el código del motor de datos. Con PostgreSQL, el énfasis ha pasado a aumentar características y capacidades, aunque el trabajo continúa en todas las áreas.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (Gibert Ginestà & Pérez Mora, 2012)



*Figura 4.* Componentes de un Sistema PostgreSQL

***Fuente:*** http://www.postgresql.org.es/sobre\_postgresql

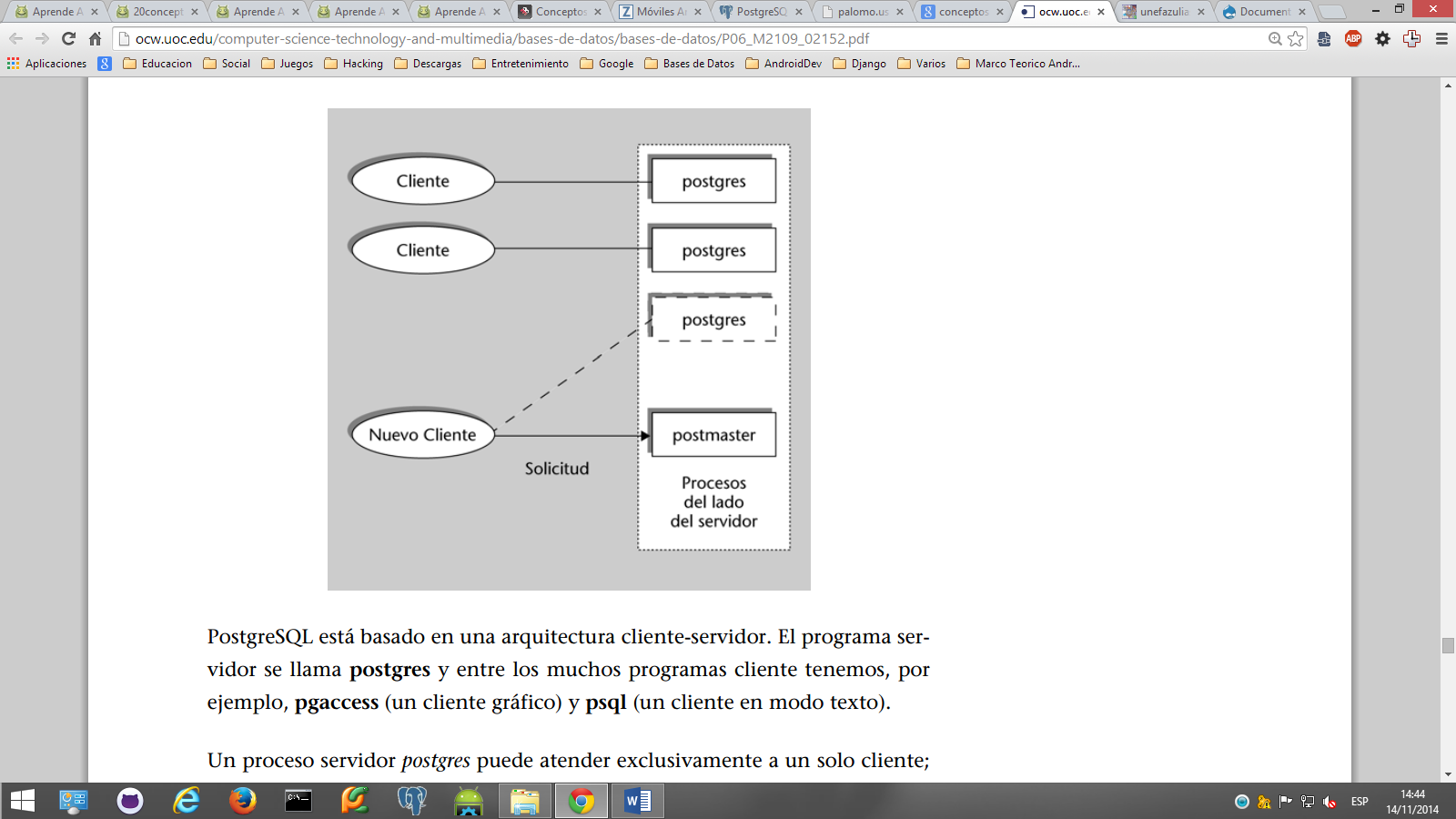
* **Aplicación cliente**: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir via TCP/IP ó sockets locales.
* **Demonio postmaster**: Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes
* **Ficheros de configuración**: Los 3 ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg\_hba.conf y pg\_ident.conf
* **Procesos hijos postgres**: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.
* **PostgreSQL share buffer cache**: Memoria compartida usada por POstgreSQL para almacenar datos en caché.
* **Write-Ahead Log (WAL)**: Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).
* **Kernel disk buffer cache**: Caché de disco del sistema operativo
* **Disco**: Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione. (PostgreSQL.org, 2010)

Las principales mejoras en PostgreSQL incluyen:

* Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg\_dump mientras la base de datos permanece disponible para consultas.
* Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).
* Se han añadido funcionalidades en línea con el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadena literal, conversión de tipos y entrada de enteros binarios y hexadecimales.
* Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.
* La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada. (Gibert Ginestà & Pérez Mora, 2012)

#### Arquitectura de PostgreSQL

El siguiente gráfico muestra de forma esquemática las entidades involucradas en el funcionamiento normal del gestor de bases de datos:



*Figura 5.* Arquitectura de PostgreSQL

***Fuente:*** http://ocw.uoc.edu/

PostgreSQL está basado en una arquitectura cliente-servidor. El programa servidor se llama **Postgres** y entre los muchos programas cliente tenemos, por ejemplo, **pgaccess** (un cliente gráfico) y **psql** (un cliente en modo texto).

Un proceso servidor  *postgres* puede atender exclusivamente a un solo cliente; es decir, hacen fala tantos procesos servidor *postgres* como clientes haya. El proceso **postmaster** es el encargado de ejecutar un nuevo servidor para cada cliente que solicite una conexión.

Se llama **sitio** al equipo anfitrión (*host*) que almacena un conjunto de bases de datos PostgreSQL. En un *sitio* se ejecuta solamente un proceso *postmaster* y múltiples procesos *postgres.* Los clientes pueden ejecutarse en el mismo sitio o en equipos remotos conectados por TCP/IP.Es posible restringir el acceso a usuarios o direcciones IP modificando las opciones del archivo pg\_hba-conf, que se encuentra en /etc/postgresql/pg\_hba.conf. Este archivo, junto con /etc/postgresql/postgresql.conf es particularmente importante, porque algunos de sus parámetros de configuración por defecto provocan multitud de problemas al conectar inicialmente y porque en ellos se especifican los mecanismos de autenticación que usará PostgreSQL para verificar las credenciales de los usuarios.

Para habilitar la conexión a PostgreSQL desde clientes remotos, debemos verificar el parámetro tcpip\_socket = true en el fichero */etc/postgresql/postgresql.conf*.

A continuación, para examinar los métodos de autenticación y las posibilidades de conexión de clientes externos, debemos mirar el fichero */etc/postgresql/pg\_hba.conf*, donde se explicita la acción que hay que emprender para cada conexión proveniente de cada *host* externo, o grupo de *hosts.* (Gibert Ginestà & Pérez Mora, 2012, pág. 65)

#### Acceso a un Servidor PostgreSQL

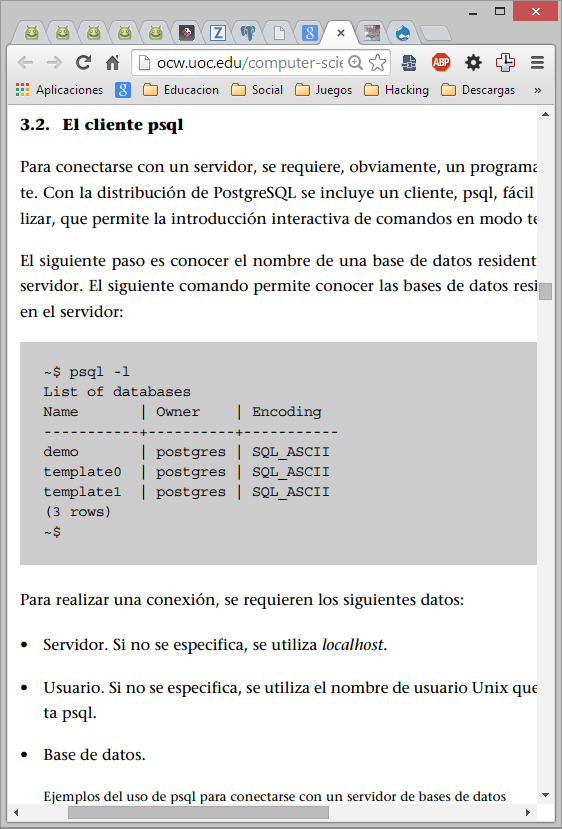
##### La conexión con el servidor

Antes de intentar conectarse con el servidor, debemos asegurarnos de que está funcionando y que admite conexiones, locales (el SGBD se está ejecutando en la misma máquina que intenta la conexión) o remotas. Una vez comprobado el correcto funcionamiento del servidor, debemos disponer de las credenciales necesarias para la conexión. Para simplificar, supondremos que disponemos de las credenciales del administrador de la base de datos (normalmente, usuario PostgreSQL y su contraseña). (Gibert Ginestà & Pérez Mora, 2012, pág. 20)

##### El cliente psql

Para conectarse con un servidor, se requiere, obviamente, un programa cliente. Con la distribución de PostgreSQL se incluye un cliente, psql, fácil de utilizar, que permite la introducción interactiva de comandos en modo texto.

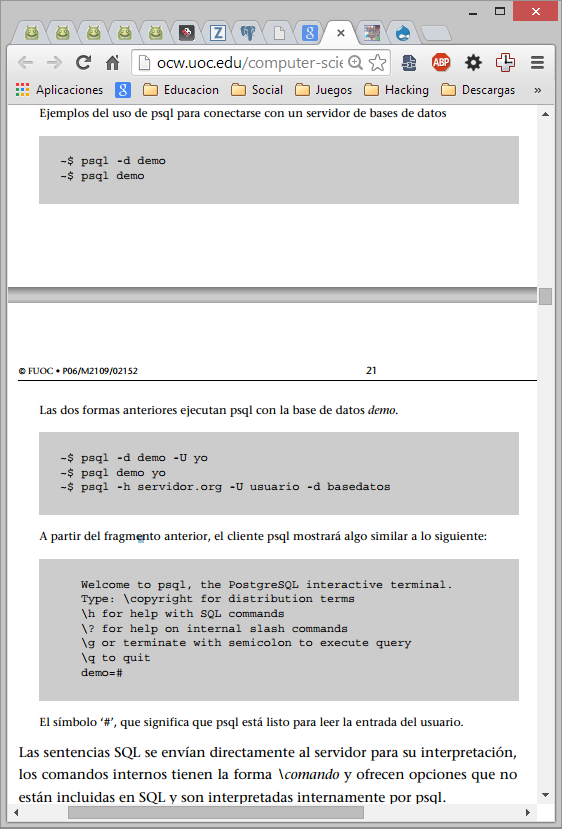
El siguiente paso es conocer el nombre de una base de datos residente en el servidor. El siguiente comando permite conocer las bases de datos residentes en el servidor:



*Figura 6.* Base de datos residentes en el servidor

***Fuente:*** http://ocw.uoc.edu/

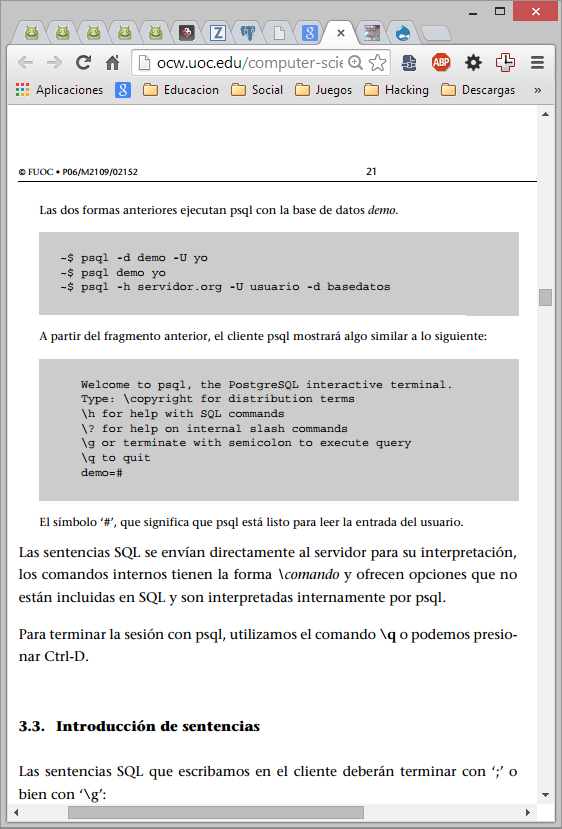
Para realizar una conexión, se requieren los siguientes datos:

* Servidor: Si no se especifica, se utiliza *localhost.*
* Usuario: Si no se especifica, se utilizar el nombre de usuario Unix que ejecuta psql.
* Base de datos.

*Figura 7.* Ejemplo del uso de psql para conectarse con un servidor de base de datos

***Fuente:*** http://ocw.uoc.edu/

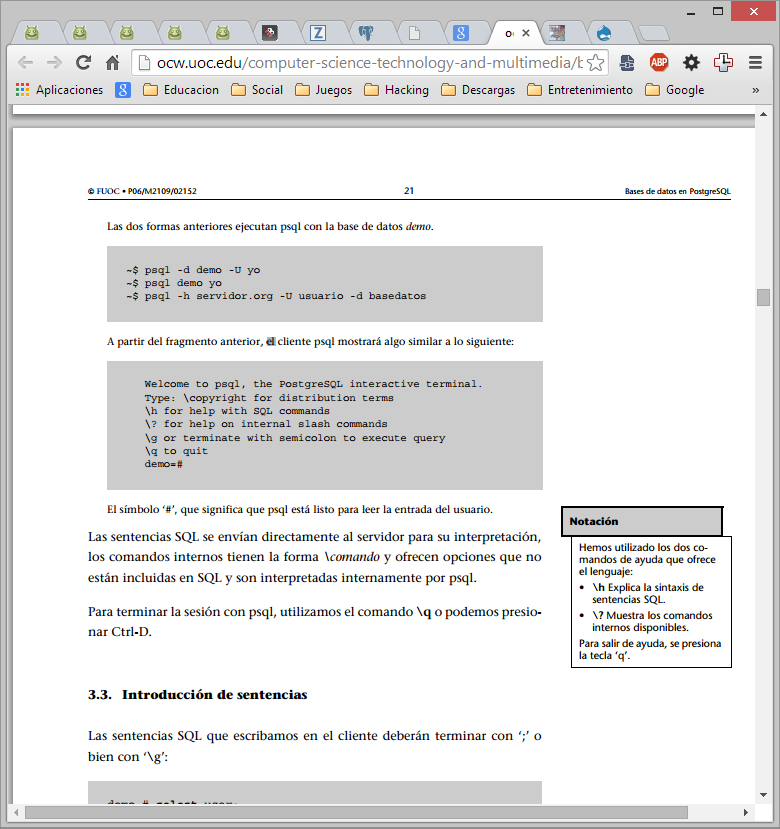
Las dos formas anteriores ejecutan psql con la base de datos *demo.*



***Fuente:*** http://ocw.uoc.edu/

*Figura 8.* Ejemplo de ejecución de psql con la base datos demo

A partir del fragmento anterior, el cliente psql mostrará algo similar a lo siguiente:



***Fuente:*** http://ocw.uoc.edu/

*Figura 9.* Resultado de la ejecución del comando anterior

El símbolo ‘#’, que significa que psql está listo para leer la entrada del usuario.

Las sentencias SQL se envían directamente al servidor para su interpretación, los comandos internos tienen la forma \*comando*  y ofrecen opciones que no están incluidas en SQL y son interpretadas internamente por psql.

Para terminar la sesión con psql, utilizamos el comando \q o podemos presionar Ctrl-D. (Gibert Ginestà & Pérez Mora, 2012, págs. 20-21)

### Introducción a HTML5

HTML5 es la actualización de HTML, el lenguaje en el que es creada la web. HTML5 también es un término de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de Javascript. La versión anterior y más usada de HTML, HTML4, carece de características necesarias para la creación de aplicaciones modernas basadas en un navegador. El uso fuerte de Javascript ha ayudado a mejorar esto, gracias a frameworks como jQuery, jQuery UI2, Sproutcore3, entre otros.

Flash en especial ha sido usado en reemplazo de HTML para desarrollar web apps que superaran las habilidades de un navegador: Audio, video, webcams, micrófonos, datos binarios, animaciones vecto­riales, componentes de interfaz complejos, entre muchas otras cosas. Ahora HTML5 es capaz de hacer esto sin necesidad de plugins y con una gran compatibilidad entre navegadores. (Vega & Vab Der Henst, 2011)

#### Componentes Básicos

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs (Interface de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y Javascript hace el resto que es extremadamente significativo.

La estructura sigue siendo parte esencial de un documento. La misma provee los elementos necesarios para ubicar contenido estático o dinámico, y es también una plataforma básica para aplicaciones. Con la variedad de dispositivos para acceder a Internet y la diversidad de interfaces disponibles para interactuar con la web, un aspecto básico como la estructura se vuelve parte vital del documento. Ahora la estructura debe proveer forma, organización y flexibilidad, y debe ser tan fuerte como los fundamentos de un edificio. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 18)

#### Estructura global de un documento HTML5

Los documentos HTML se encuentran estrictamente organizados. Cada parte del documento está diferenciada, declarada y determinada por etiquetas específicas. En esta parte del capítulo vamos a ver cómo construir la estructura global de un documento HTML y los nuevos elementos semánticos incorporados en HTML5.

**<!DOCTYPE>**

En primer lugar necesitamos indicar el tipo de documento que estamos creando. Esto en HTML5 es extremadamente sencillo: <!DOCTYPE html>

Esta línea debe ser la primera línea del archivo, sin espacios o líneas que la precedan. De esta forma, el modo estándar del navegador es activado y las incorporaciones de HTML5 son interpretadas siempre que sea posible, o ignoradas en caso contrario.

**<html>**

Luego de declarar el tipo de documento, debemos comenzar a construir la estructura HTML. Como siempre, la estructura tipo árbol de este lenguaje tiene su raíz en el elemento <html>. Este elemento envolverá al resto del código:

<!DOCTYPE html>

<html lang=”es”>

</html>

El atributo lang en la etiqueta de apertura <html> es el único atributo que necesitamos especificar en HTML5. Este atributo define el idioma humano del contenido del documento que estamos creando, en este caso es por español. HTML usa un lenguaje de etiquetas para construir páginas web. Estas etiquetas HTML son palabras clave y atributos rodeados de los signos mayor y menor (por ejemplo, <html lang="es">). En este caso, html es la palabra clave y lang es el atributo con el valor es. La mayoría de las etiquetas HTML se utilizan en pares, una etiqueta de apertura y una de cierre, y el contenido se declara entre ellas. En el ejemplo, <html lang="es"> indica el comienzo del código HTML y </html> indica el final. Compare las etiquetas de apertura y cierre y verá que la de cierre se distingue por una barra invertida antes de la palabra clave (por ejemplo, </html>). El resto de código será insertado entre estas dos etiquetas: <html> ... </html>.

**<head>**

El código HTML insertado entre las etiquetas <html> tiene que ser dividido entredós secciones principales. Al igual que en versiones previas de HTML, la primera sección es la cabecera y la segunda el cuerpo. El elemento <head> va primero, por supuesto, y al igual que el resto de los elementos estructurales tiene una etiqueta de apertura y una de cierre:

<!DOCTYPE html>

<html lang="es">

<head>

</head>

</html>

La etiqueta no cambió desde versiones anteriores y su propósito sigue siendo exactamente el mismo. Dentro de las etiquetas <head> definiremos el título de la página web, se declara el set de caracteres correspondiente, se provee información general acerca del documento y se incorpora los archivos externos con estilos, códigos Javascript o incluso imágenes necesarias para generar la página en la pantalla. Excepto por el título y algunos íconos, el resto de la información incorporada en el documento entre estas etiquetas es invisible para el usuario. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 19)

**<body>**

La siguiente gran sección que es parte principal de la organización de un documento HTML es el cuerpo. El cuerpo representa la parte visible de todo documento y es especificado entre etiquetas <body>. Estas etiquetas tampoco han cambiado en relación con versiones previas de HTML:

<!DOCTYPE html>

<html lang="es">

<head>

</head>

<body>

</body>

</html>

HTML es un lenguaje de etiquetas, un listado de elementos que usualmente se utilizan en pares y que pueden ser anidados (totalmente contenidos uno dentro del otro). Entre las etiquetas <html> insertamos otras etiquetas especificando dos importantes partes de la estructura básica: <head> para la cabecera y <body> para el cuerpo del documento.

**<meta>**

Algunos cambios e innovaciones fueron incorporados dentro de la cabecera, y uno de ellos es la etiqueta que define el juego de caracteres a utilizar para mostrar el documento. Ésta es una etiqueta <meta> que especifica cómo el texto será presentado en pantalla:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset=”iso-8859-1”>

</head>

<body>

</body>

</html>

La innovación de este elemento en HTML5, como en la mayoría de los casos, es solo simplificación. La nueva etiqueta <meta> para la definición del tipo de caracteres es más corta y simple. Por supuesto, podemos cambiar el tipo iso-8859-1 por el necesario para nuestros documentos y agregar otras etiquetas <meta> como description o keywords para definir otros aspectos de la página web, como es mostrado en el siguiente ejemplo:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="iso-8859-1">

<meta name=”description” content=”Ejemplo de HTML5”>

<meta name=”keywords” content=”HTML5, CSS3, Javascript”>

</head>

<body>

</body>

</html>

Hay varios tipos de etiqueta <meta> que pueden ser incluidas para declarar información general sobre el documento, pero esta información no es mostrada en la ventana del navegador, es solo importante para motores de búsqueda y dispositivos que necesitan hacer una vista previa del documento u obtener un sumario de la información que contiene. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 20)

**<title>**

La etiqueta <title>, como siempre, simplemente especifica el título del documento:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="iso-8859-1">

<meta name="description" content="Ejemplo de HTML5">

<meta name="keywords" content="HTML5, CSS3, JavaScript">

<title>Este texto es el título del documento</title>

</head>

<body>

</body>

</html>

El texto entre las etiquetas <title> es el título del documento que estamos creando. Normalmente este texto es mostrado en la barra superior de la ventana del navegador.

**<link>**

Elemento que va dentro de la cabecera del documento es <link>. Este elemento es usado para incorporar estilos, códigos Javascript, imágenes o iconos desde archivos externos. Uno de los usos más comunes para <link> es la incorporación de archivos con estilos CSS:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="iso-8859-1">

<meta name="description" content="Ejemplo de HTML5">

<meta name="keywords" content="HTML5, CSS3, JavaScript">

<title>Este texto es el título del documento</title>

<link rel=”stylesheet” href=”misestilos.css”>

</head>

<body>

</body>

</html>

En HTML5 ya no se necesita especificar qué tipo de estilos estamos insertando, por lo que el atributo type fue eliminado. Solo se necesitan dos atributos para incorporar en el archivo de estilos: rel y href. El atributo rel significa “relación” y es acerca de la relación entre el documento y el archivo que se está incorporando por medio de href. En este caso, el atributo rel tiene el valor stylesheet que le dice al navegador que el archivo misestilos.css es un archivo CSS con estilos requeridos para presentar la página en pantalla. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 22)

#### Estructura del cuerpo del documento HTML5

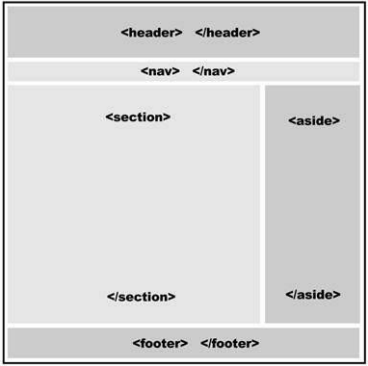
La estructura del cuerpo (el código entre las etiquetas <body>) generará la parte visible del documento. Este es el código que producirá nuestra página web. HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para este propósito fue <table>. Las tablas permitían a los diseñadores acomodar datos, texto, imágenes y herramientas dentro de filas y columnas de celdas, incluso sin que hayan sido concebidas para este propósito.

En los primeros días de la web, las tablas fueron una revolución, un gran paso hacia adelante con respecto a la visualización de los documentos y la experiencia ofrecida a los usuarios. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.

El elemento <div> comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML, CSS y Javascript, el uso de <div> se volvió una práctica común. Pero este elemento, así como <table>, no provee demasiada información acerca de las parte del cuerpo que está representando. Desde imágenes a menús, textos, enlaces, códigos, formularios, cualquier cosa puede ir entre las etiquetas de apertura y cierre de un elemento <div>. En otras palabras, la palabra clave div solo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene. Luego de la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, la identificación de cada parte del documento es una tarea que se ha vuelto más relevante que nunca.

HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo. En HTML5 las secciones más importantes son diferenciadas y la estructura principal ya no depende más de los elementos <div> o <table>.

HTML5 considera esta estructura básica y provee nuevos elementos para diferenciar y declarar cada una de sus partes. A partir de ahora podemos decir al navegador para qué es cada sección: (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 24)



*Fuente:* El gran libro de HTML5, CSS3 y JavaScript

*Figura 10.* Representación visual de un diseño utilizando elementos HTML5.

##### <header>

Hacer cosas como <div id=”header”> es un poco estúpido cuando el 99% de los proyectos web tienen una cabecera. <header> está diseñada para reemplazar la necesidad de crear divs sin signifi­cado semántico. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 27)

##### <nav>

Igual que <header>, <nav> está diseñado para que ahí coloques la botonera de navegación principal. Puedes colocar cualquier etiqueta dentro, aunque lo recomendado es usar listas <ul>. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 28)

##### <section>

Define un área de contenido única dentro del sitio. En un blog, sería la zona donde están todos los posts. En un video de youtube, habría un section para el video, uno para los datos del video, otro para la zona de comentarios. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 29)

##### <aside>

Cualquier contenido que no esté relacionado con el objetivo primario de la página va en un aside. En un blog, obviamente el aside es la barra lateral de información. En el home de un periódico, puede ser el área de indicadores económicos. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 29)

##### <footer>

Este es obvio. Es el pie de página y todo lo que lo compone. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 30)

#### Elementos dentro del cuerpo del documento HTML5

Los elementos HTML5 definidos hasta el momento ayudan a identificar cada sección del diseño y asignar un propósito intrínseco a cada una de ellas, pero lo que es realmente importante para el sitio web se encuentra en el interior de estas secciones.

La mayoría de los elementos fueron creados para construir una estructura para el documento HTML que pueda ser identificada y reconocida por los navegadores y nuevos dispositivos. Aprendimos acerca de la etiqueta <body> usada para declarar el cuerpo o parte visible del documento, la etiqueta <header> con la que se agrupa información importante para el cuerpo, la etiqueta <nav> que provee ayuda para la navegación del sitio web, la etiqueta <section> necesaria para contener la información más relevante, y también <aside> y <footer> para ofrecer información adicional de cada sección y del documento mismo. Pero ninguno de estos elementos declara algo acerca del contenido. Todos tienen un específico propósito estructural. Necesitamos poder diferenciar estos elementos y establecer una relación entre ellos dentro de la estructura. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 32)

**<article>**

Define zonas únicas de contenido independiente. En el home de un blog, cada post sería un article. En un post del blog, el post y cada uno de sus comentarios sería un <article>. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 33)

**<hgroup>**

Muchos headers necesitan múltiples títulos, como un blog que tiene un título y un tagline explicando el blog. <hgroup> permite colocar un h1, h2 y h3 dentro del header sin afectar el SEO, permitiendo usar otro h1 en el sitio. En el HTML actual, sólo puedes usar h1 una vez por sitio o el h1 pierde prioridad de SEO. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 34)

**<figure> y <figcaption>**

La etiqueta <figure> fue creada para ayudarnos a ser aún más específicos a la hora de declarar el contenido del documento. Antes de que este elemento sea introducido, no podíamos identificar el contenido que era parte de la información pero a la vez independiente, como ilustraciones, fotos, videos, etc. Cuando nos encontramos con esta clase de información, las etiquetas <figure> pueden ser usadas para identificarla:<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="iso-8859-1">

<meta name="description" content="Ejemplo de HTML5">

<meta name="keywords" content="HTML5, CSS3, JavaScript">

<title>Este texto es el título del documento</title>

<link rel="stylesheet" href="misestilos.css">

</head>

<body>

<header>

<h1>Este es el título principal del sitio web</h1>

</header>

<nav>

<ul>

<li>principal</li>

<li>fotos</li>

<li>videos</li>

<li>contacto</li>

</ul>

</nav>

<section>

<article>

<header>

<hgroup>

<h1>Título del mensaje uno</h1>

<h2>Subtítulo del mensaje uno</h2>

</hgroup>

<p>publicado 10-12-2011</p>

</header>

Este es el texto de mi primer mensaje

<figure>

<img src=”http://minkbooks.com/content/myimage.jpg”>

<figcaption>

Esta es la imagen del primer mensaje

</figcaption>

</figure>

<footer>

<p>comentarios (0)</p>

</footer>

</article>

<article>

<header>

<hgroup>

<h1>Título del mensaje dos</h1>

<h2>Subtítulo del mensaje dos</h2>

</hgroup>

<p>publicado 15-12-2011</p>

</header>

Este es el texto de mi segundo mensaje

<footer>

<p>comentarios (0)</p>

</footer>

</article>

</section>

<aside>

<blockquote>Mensaje número uno</blockquote>

<blockquote>Mensaje número dos</blockquote>

</aside>

<footer>

Derechos Reservados &copy; 2010-2011

</footer>

</body>

</html> (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 35)

### Introducción a CSS3

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación delos documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 42)

#### Breve Historia

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento exponencial del lenguaje HTML para la creación de documentos electrónicos. La guerra de navegadores y la falta de un estándar para la definición de los estilos dificultaban la creación de documentos con la misma apariencia en diferentes navegadores.

El organismo W3C (http://www.w3.org/) (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (*Cascading HTML Style Sheets*) y la SSP (*Stream-based Style Sheet Proposal*).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (*Cascading Style Sheets*).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1". A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS. El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando (la última actualización es del 19 de julio de 2007). La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. (Vega & Vab Der Henst, 2011)

#### Novedades de CSS3

* Aplicar imágenes a los bordes y añadir varios bordes,
* Crear fondos con degradados y con imágenes múltiples, usar la transparencia con los colores y con los Elementos,
* Aplicar sombras a los elementos (cajas, texto, etc.),
* Aplicar transformaciones, transiciones y animaciones a las propiedades y a los elementos,
* Insertar fuentes con caracteres diversos,
* Crear sitios web que se adapten a los distintos tamaños de pantalla. (Vega & Vab Der Henst, 2011)

#### Conceptos básicos sobre estilos

Antes de comenzar a insertar reglas CSS en el archivo de estilos y aplicar un modelo de caja, se debe revisar los conceptos básicos sobre estilos CSS que van a ser utilizados. Aplicar estilos a los elementos HTML cambia la forma en que estos son presentados en pantalla. Los navegadores proveen estilos por defecto que en la mayoría de los casos no son suficientes para satisfacer las necesidades de los diseñadores.

##### Estilos en línea

Una de las técnicas más simples para incorporar estilos CSS a un documento HTML es la de asignar los estilos dentro de las etiquetas por medio del atributo style. El siguiente código muestra un documento HTML simple que contiene el elemento <p> modificado por el atributo *style* con el valor *font-size: 20px.* Este estilo cambia el tamaño por defecto del texto dentro del elemento <p> a un nuevo tamaño de 20 pixeles.

<!DOCTYPE html>

<html lang="es">

<head>

<title>Este es el título del documento</title>

</head>

<body>

<p style=”font-size: 20px”>Mi texto</p>

</body>

</html> (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 45)

##### Estilos embebidos

Una mejor alternativa es insertar los estilos en la cabecera del documento y luego usar referencias para afectar los elementos HTML correspondientes:

<!DOCTYPE html>

<html lang="es">

<head>

<title>Este texto es el título del documento</title>

<style>

p { font-size: 20px }

</style></head>

<body>

<p>Mi texto</p>

</body>

</html>

El elemento <style> permite a los desarrolladores agrupar estilos CSS dentro del documento. En versiones previas de HTML era necesario especificar qué tipo de estilos serían insertados. En HTML5 los estilos por defecto son CSS, por lo tanto no necesitamos agregar ningún atributo en la etiqueta de apertura <style>. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 46).

#### Nuevas propiedades CSS3

CSS3 provee nuevas propiedades para crear efectos visuales y dinámicos que son parte esencial de la web:

##### border-radius:

Esta propiedad genera esquinas redondeadas para la caja formada por el elemento. Posee dos parámetros diferentes que dan forma a la esquina. El primer parámetro determina la curvatura horizontal y el segundo la vertical, otorgando la posibilidad de crear una elipsis. Para declarar ambos parámetros de la curva, los valores deben ser separados por una barra (por ejemplo, border-radius: 15px / 20px). Usando solo un valor determinaremos la misma forma para todas las esquinas (por ejemplo, border-radius: 20px). Un valor para cada esquina puede ser declarado en un orden que sigue las agujas del reloj, comenzando por la esquina superior izquierda.

##### box-shadow:

Esta propiedad crea sombras para la caja formada por el elemento. Puede tomar cinco parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, el valor de difuminación, y la palabra clave inset para generar una sombra interna. Los desplazamientos pueden ser negativos, y el valor de difuminación y el valor inset son opcionales (por ejemplo, box-shadow: #000000 5px 5px 10px inset).

##### text-shadow

Esta propiedad es similar a box-shadow pero específica para textos. Toma cuatro parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, y el valor de difuminación (por ejemplo, text-shadow: #000000 5px 5px 10px).

##### @font-face

Esta regla nos permite cargar y usar cualquier fuente que necesitemos. Primero, debemos declarar la fuente, proveer un nombre con la propiedad font-family y y especificar el archivo con src (por ejemplo, @font-face{ font-family: Mifuente; src: url('font.ttf') }). Luego de esto, podremos asignar la fuente (en el ejemplo Mi fuente) a cualquier elemento del documento.

##### linear-gradient(posición inicio, color inicial, color final)

Esta función puede ser aplicada a las propiedades background o background-image para generar un gradiente lineal. Los atributos indican el punto inicial y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, en porcentaje o usando las palabras clave top, bottom, left y right. El punto de inicio puede ser reemplazado por un ángulo para proveer una dirección específica para el gradiente (por ejemplo, linear-gradient(top, #FFFFFF 50%, #00669990%);).

##### ***radial-gradient(posición inicio, forma, color inicial, color final)***

Esta función puede ser aplicada a las propiedades background o background-image para generar un gradiente radial. La posición de inicio es el origen y puede ser declarado en pixeles, porcentaje o como una combinación de las palabras clave center, top, bottom, left y right. Existen dos valores para la forma: circle y ellipse, y puntos de terminación pueden ser declarados para cada color indicando la posición donde la transición comienza (por ejemplo, radial-gradient(center, circle, #FFFFFF 0%, #006699 200%);).

##### rgba()

Esta función es una mejora de rgb(). Toma cuatro valores: el color rojo (0-255), el color verde (0-255), el color azul (0-255), yla opacidad (un valor entre 0 y 1).

##### hsla()

Esta función es una mejora de hsl(). Puede tomar cuatro valores: el tono (un valor entre 0 y 360), la saturación (un porcentaje), la luminosidad (un porcentaje), yla opacidad (un valor entre 0 y 1).

##### outline

Esta propiedad fue mejorada con la incorporación de otra propiedad llamada outline-offset. Ambas propiedades combinadas generan un segundo borde alejado del borde original del elemento (por ejemplo, outline: 1px solid #000000; outline-offset: 10px;).

##### border-image

Esta propiedad crea un borde con una imagen personalizada. Necesita que el borde sea declarado previamente con las propiedades border o border-with, y toma al menos tres parámetros: la URL de la imagen, el tamaño de las piezas que serán tomadas de la imagen para construir el borde, y una palabra clave que especifica cómo esas piezas serán ubicadas alrededor del elemento (por ejemplo, border-image: url("file.png") 15 stretch;)

##### transform

Esta propiedad modifica la forma de un elemento. Utiliza cuatro funciones básicas: scale (escalar), rotate (rotar), skew (inclinar), y translate (trasladar o mover). La función scale recibe solo un parámetro. Un valor negativo invierte el elemento, valores entre 0 y 1 reducen el elemento y valores mayores que 1 expanden el elemento (por ejemplo, transform: scale(1.5);). La función rotate usa solo un parámetro expresado en grados para rotar el elemento (por ejemplo, transform: rotate(20deg);). La función skew recibe dos valores, también en grados, para la transformación horizontal y vertical (por ejemplo, transform: skew(20deg, 20deg);).

##### transition

Esta propiedad puede ser aplicada para crear una transición entre dos estados de un elemento. Recibe hasta cuatro parámetros: la propiedad afectada, el tiempo que le tomará a la transición desde el comienzo hasta el final, una palabra clave para especificar cómo la transición será realizada (ease, linear, ease-in, ease-out, ease-in.-out) y un valor de retardo que determina el tiempo que la transición tardará en comenzar. (El gran libro de HTML5, CSS3 y Javascript, 2013, págs. 62-66)

### Javascript

Javascript es un lenguaje interpretado usado para múltiples propósitos pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, Javascript fue expandido en relación con portabilidad e integración.Ala vez, interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores. La idea es hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web. (El gran libro de HTML5, CSS3 y Javascript, 2013, pág. 84)

#### Sintaxis básica

Comprensión de declaraciones, nombres de variables, espacios en blanco, y otras sintaxis básicas de JavaScript. Declaración simple de variable:

var foo = 'hola mundo';

Los espacios en blanco no tienen valor fuera de las comillas

var foo = 'hola mundo';

Los paréntesis indican prioridad

2 \* 3 + 5; // es igual a 11, la multiplicación ocurre primero

2 \* (3 + 5); // es igual a 16, por los paréntesis, la suma ocurre primero

La tabulación mejora la lectura del código, pero no posee ningún significado especial

var foo = function() {

console.log('hola');

};

#### Operadores

##### Operadores Básicos

Los operadores básicos permiten manipular valores.

###### Concatenación

var foo = 'hola';

var bar = 'mundo';

console.log(foo + ' ' + bar); // la consola de depuración muestra 'hola mundo'

###### Multiplicación y división

2 \* 3;

2 / 3;

###### Incrementación y decrementación

var i = 1;

var j = ++i; // incrementación previa: j es igual a 2; i es igual a 2

var k = i++; // incrementación posterior: k es igual a 2; i es igual a 3

##### Operaciones con Números y Cadenas de Caracteres

En JavaScript, las operaciones con números y cadenas de caracteres (en inglés strings) pueden ocasionar resultados no esperados.

###### Suma vs. Concatenación

var foo = 1;

var bar = '2';

console.log(foo + bar); // error: La consola de depuración muestra 12.Forzar a una cadena de caracteres actuar como un número

var foo = 1;

var bar = '2';

// el constructor 'Number' obliga a la cadena comportarse como un número

console.log(foo + Number(bar)); // la consola de depuración muestra 3

El constructor Number, cuando es llamado como una función (como se muestra en el ejemplo) obliga a su argumento a comportarse como un número. También es posible utilizar el operador de suma unaria, entregando el mismo resultado:

Forzar a una cadena de caracteres actuar como un número (utilizando el operador de suma unaria) console.log (foo + +bar); (Fundamentos de Jquery, 2013, pág. 8)

##### Operadores Lógicos

Los operadores lógicos permiten evaluar una serie de operandos utilizando operaciones AND y OR.

###### Operadores lógicos AND y OR

var foo = 1;

var bar = 0;

var baz = 2;

foo || bar; // devuelve 1, el cual es verdadero (true)

bar || foo; // devuelve 1, el cual es verdadero (true)

foo && bar; // devuelve 0, el cual es falso (false)

foo && baz; // devuelve 2, el cual es verdadero (true)

baz && foo; // devuelve 1, el cual es verdadero (true)

El operador || (OR lógico) devuelve el valor del primer operando, si éste es verdadero; caso contrario devuelve el segundo operando. Si ambos operandos son falsos devuelve falso (false). El operador && (AND lógico) devuelve el valor del primer operando si éste es falso; caso contrario devuelve el segundo operando. Cuando ambos valores son verdaderos devuelve verdadero (true), sino devuelve falso. (Fundamentos de Jquery, 2013, pág. 9)

##### Operadores de Comparación

Los operadores de comparación permiten comprobar si determinados valores son equivalentes o idénticos.

###### Operadores de Comparación

var foo = 1;

var bar = 0;

var baz = '1';

var bim = 2;

foo == bar; // devuelve falso (false)

foo != bar; // devuelve verdadero (true)

foo == baz; // devuelve verdadero (true); tenga cuidado

foo === baz; // devuelve falso (false)

foo !== baz; // devuelve verdadero (true)

foo === parseInt(baz); // devuelve verdadero (true)

foo > bim; // devuelve falso (false)

bim > baz; // devuelve verdadero (true)

foo <= baz; // devuelve verdadero (true) (Fundamentos de Jquery, 2013, pág. 10)

##### Declaración Switch

En lugar de utilizar una serie de declaraciones if/else/else if/else, a veces puede ser útil la utilización de la declaración switch. [Definición: La declaración Switch evalúa el valor de una variable o expresión, y ejecuta diferentes bloques de código dependiendo de ese valor.

Una declaración Switch

switch (foo) {

case 'bar':

alert('el valor es bar');

break;

case 'baz':

alert('el valor es baz');

break;

default:

alert('de forma predeterminada se ejecutará este código');

break;

}

Las declaraciones switch son poco utilizadas en JavaScript, debido a que el mismo comportamiento es posible obtenerlo creando un objeto, el cual posee más potencial ya que es posible reutilizarlo, usarlo para realizar pruebas, etc. Por ejemplo:

var stuffToDo = {

'bar' : function() {

alert('el valor es bar');

},

'baz' : function() {

alert('el valor es baz');

},

'default' : function() {

alert('de forma predeterminada se ejecutará este código');

}

};

if (stuffToDo[foo]) {

stuffToDo[foo]();

} else {

stuffToDo['default']();

}

(Fundamentos de Jquery, 2013, pág. 12)

##### Bucles

Los bucles (en inglés loops) permiten ejecutar un bloque de código un determinado número de veces.

Bucles

// muestra en la consola 'intento 0', 'intento 1', ..., 'intento 4'

for (var i=0; i<5; i++) {

console.log('intento ' + i);

}

###### Bucles Utilizando For

Un bucle utilizando for se compone de cuatro estados y posee la siguiente estructura:

for ([expresiónInicial]; [condición]; [incrementoDeLaExpresión])

[cuerpo]

El estado expresión Inicial es ejecutado una sola vez, antes que el bucle comience. Éste otorga la oportunidad de preparar o declarar variables. El estado condición es ejecutado antes de cada repetición, y retorna un valor que decide si el bucle debe continuar ejecutándose o no. Si el estado condicional evalúa un valor falso el bucle se detiene. El estado *incrementoDeLaExpresión* es ejecutado al final de cada repetición y otorga la oportunidad de cambiar el estado de importantes variables. Por lo general, este estado implica la incrementación o decrementación de un contador.

El cuerpo es el código a ejecutar en cada repetición del bucle.

Un típico bucle utilizando for

for (var i = 0, limit = 100; i < limit; i++) {

// Este bloque de código será ejecutado 100 veces

console.log('Actualmente en ' + i);

// Nota: el último registro que se mostrará

// en la consola será "Actualmente en 99"

} (Fundamentos de Jquery, 2013, pág. 13)

###### Bucles Utilizando While

Un bucle utilizando while es similar a una declaración condicional if, excepto que el cuerpo va a continuar ejecutándose hasta que la condición a evaluar sea falsa.

while ([condición]) [cuerpo]

Un típico bucle utilizando while

var i = 0;

while (i < 100) {

// Este bloque de código se ejecutará 100 veces

console.log('Actualmente en ' + i);

i++; // incrementa la variable i

}

Puede notar que en el ejemplo se incrementa el contador dentro del cuerpo del bucle, pero también es posible combinar la condición y la incrementación, como se muestra a continuación:

Bucle utilizando while con la combinación de la condición y la incrementación

var i = -1;

while (++i < 100) {

// Este bloque de código se ejecutará 100 veces

console.log('Actualmente en ' + i);

}

Se comienza en -1 y luego se utiliza la incrementación previa (++i). (Fundamentos de Jquery, 2013, pág. 14)

###### Bucles Utilizando Do-while

Este bucle es exactamente igual que el bucle utilizando while excepto que el cuerpo es ejecutado al menos una vez antes que la condición sea evaluada.

do [cuerpo] while ([condición])

Un bucle utilizando do-while

do {

// Incluso cuando la condición sea falsa

// el cuerpo del bucle se ejecutará al menos una vez.

alert('Hola');

} while (false);

Este tipo de bucles son bastantes atípicos ya que en pocas ocasiones se necesita un bucle que se ejecute al menos una vez. De cualquier forma debe estar al tanto de ellos. (Fundamentos de Jquery, 2013, pág. 15)

###### Break y Continue

Usualmente, el fin de la ejecución de un bucle resultará cuando la condición no siga evaluando un valor verdadero, sin embargo también es posible parar un bucle utilizando la declaración break dentro del cuerpo.

Detener un bucle con break

for (var i = 0; i < 10; i++) {

if (something) {

break;

}

}

También puede suceder que quiera continuar con el bucle sin tener que ejecutar más sentencias del cuerpo del mismo bucle. Esto puede realizarse utilizando la declaración continúe.

Saltar a la siguiente iteración de un bucle

for (var i = 0; i < 10; i++) {

if (something) {

continue;

}

// La siguiente declaración será ejecutada

// Si la condición 'something' no se cumple

console.log('Hola');

} (Fundamentos de Jquery, 2013, pág. 15)

#### Incorporando Javascript

Existen tres técnicas para incorporar código Javascript dentro de HTML. Sin embargo, al igual que en CSS, solo la inclusión de archivos externos es la recomendada a usar en HTML5.

##### En línea

Esta es una técnica simple para insertar Javascript en nuestro documento que se aprovecha de atributos disponibles en elementos HTML. Estos atributos son manejadores de eventos que ejecutan código de acuerdo a la acción del usuario. Los manejadores de eventos más usados son, en general, los relacionados con el ratón, como por ejemplo onclick, onMouseOver, u onMouseOut. Sin embargo, encontraremos sitios web que implementan eventos de teclado y de la ventana, ejecutando acciones luego de que una tecla es presionada o alguna condición en la ventana del navegador cambia (por ejemplo, onload u onfocus).

<!DOCTYPE html>

<html lang=”es”>

<head>

<title>Este texto es el título del documento</title>

</head>

<body>

<div id=”principal”>

<p onclick=”alert(‘hizo clic!’)”>Hacer Clic</p>

<p>No puede hacer clic</p>

</div>

</body>

</html>

Usando el manejador de eventos onclick, un código es ejecutado cada vez que el usuario hace clic con el ratón sobre el texto que dice “Hacer Clic”. Lo que el manejador onclick está diciendo es algo como: “cuando alguien haga clic sobre este elemento ejecute este código” y el código en este caso es una función predefinida en Javascript que muestra una pequeña ventana con el mensaje “hizo clic!”. Intente cambiar el manejador onclick por onMouseOver, por ejemplo, y verá cómo el código es ejecutado solo pasando el puntero del ratón sobre el elemento. El uso de Javascript dentro de etiquetas HTML está permitido en HTML5, pero por las mismas razones que en CSS, esta clase de práctica no es recomendable. El código HTML se extiende innecesariamente y se hace difícil de mantener y actualizar. Así mismo, el código distribuido sobre todo el documento complica la construcción de aplicaciones útiles.

##### Embebido

Para trabajar con códigos extensos y funciones personalizadas debemos agrupar los códigos en un mismo lugar entre etiquetas <script>. El elemento <script> actúa exactamente igual al elemento <style> usado para incorporar estilos CSS. Nos ayuda a organizar el código en un solo lugar, afectando a los elementos HTML por medio de referencias. Del mismo modo que con el elemento <style>, en HTML5 no debemos usar ningún atributo para especificar lenguaje. Ya no es necesario incluir el atributo type en la etiqueta <script>. HTML5 asigna Javascript por defecto.

<!DOCTYPE html>

<html lang="es">

<head>

<title>Este texto es el título del documento</title>

<script>

function mostraralerta(){

alert('hizo clic!');

}

function hacerclic(){

document.getElementsByTagName('p')[0].onclick=mostraralerta;

}

window.onload=hacerclic;

</script>

</head>

<body>

<div id=”principal”>

<p>Hacer Clic</p>

<p>No puede hacer Clic</p>

</div>

</body>

</html>

El elemento <script> y su contenido pueden ser posicionados en cualquier lugar del documento, dentro de otros elementos o entre ellos.

Actualmente existen tres métodos disponibles para referenciar elementos HTML desde Javascript:

**· getElementsByTagName** (usado en el Listado 4-2) referencia un elemento por su nombre o palabra clave.

**· getElementById** referencia un elemento por el valor de su atributo id.

**· getElementsByClassName** es una nueva incorporación que nos permite referenciar un elemento por el valor de su atributo class.

##### Archivos externos

Los códigos Javascript crecen exponencialmente cuando agregamos nuevas funciones y aplicamos algunas de las APIs mencionadas previamente. Códigos embebidos incrementan el tamaño de nuestros documentos y los hacen repetitivos (cada documento debe volver a incluir los mismos códigos). Para reducir los tiempos de descarga, incrementar nuestra productividad y poder distribuir y reusar el códigos en cada documento sin comprometer eficiencia, recomendamos grabar todos los códigos Javascript en uno o más archivos externos y llamarlos usando el atributo src:

<!DOCTYPE html>

<html lang="es">

<head>

<title>Este texto es el título del documento</title>

<script src=”micodigo.js”></script>

</head>

<body>

<div id=”principal”>

<p>Hacer Clic</p>

<p>No puede hacer Clic</p>

</div>

</body>

</html>

El elemento <script> en el texto carga los códigos Javascript desde un archivo externo llamado micodigo.js. De ahora en más, podremos insertar el código que deseemos en este archivo y luego incluir el mismo en cualquier documento del sitio web que lo necesite. Desde la perspectiva del usuario, esta práctica reduce tiempos de descarga y acceso al sitio web, mientras que para nosotros simplifica la organización y facilita el mantenimiento. (El gran libro de HTML5, CSS3 y Javascript, 2013, págs. 85-87)

### Ajax

#### Introducción

El método XMLHttpRequest (XHR) permite a los navegadores comunicarse con el servidor sin la necesidad de recargar la página. Este método, también conocido como Ajax (Asynchronous JavaScript and XML), permite la creación de aplicaciones ricas en interactividad.

Las peticiones Ajax son ejecutadas por el código JavaScript, el cual envía una petición a una URL y cuando recibe una respuesta, una función de devolución puede ser ejecutada la cual recibe como argumento la respuesta del servidor y realiza algo con ella. Debido a que la respuesta es asíncrona, el resto del código de la aplicación continua ejecutándose, por lo cual, es imperativo que una función de devolución sea ejecutada para manejar la respuesta.

A través de varios métodos, jQuery provee soporte para Ajax, permitiendo abstraer las diferencias que pueden existir entre navegadores Los métodos en cuestión son $.get(), $.getScript(), $.getJSON(), $.post() y $().load().

A pesar que la definición de Ajax posee la palabra “XML”, la mayoría de las aplicaciones no utilizan dicho formato para el transporte de datos, sino que en su lugar se utiliza HTML plano o información en formato JSON (JavaScript Object Notation). En general, Ajax no trabaja a través de dominios diferentes. Sin embargo, existen excepciones, como los servicios que proveen información en formato JSONP (JSON with Padding), los cuales permiten una funcionalidad limitada a través de diferentes dominios. (Fundamentos de Jquery, 2013, pág. 55)

#### Conceptos claves de Ajax

La utilización correcta de los métodos Ajax requiere primero la comprensión de algunos conceptos clave.

##### GET vs. POST

Los dos métodos HTTP más comunes para enviar una petición a un servidor son GET y POST. Es importante entender la utilización de cada uno. El método GET debe ser utilizado para operaciones no-destructivas es decir, operaciones en donde se está “obteniendo” datos del servidor, pero no modificando. Por ejemplo, una consulta a un servicio de búsqueda podría ser una petición GET. Por otro lado, las solicitudes GET pueden ser almacenadas en la cache del navegador, pudiendo conducir a un comportamiento impredecible si no se lo espera.

Generalmente, la información enviada al servidor, es enviada en una cadena de datos (en inglés query string).

El método POST debe ser utilizado para operaciones destructivas — es decir, operaciones en donde se está incorporando información al servidor. Por ejemplo, cuando un usuario guarda un artículo en un blog, esta acción debería utilizar POST. Por otro lado, este tipo de método no se guarda en la cache del navegador. Además, una cadena de datos puede ser parte de la URL, pero la información tiende a ser enviada de forma separada. (Fundamentos de Jquery, 2013, págs. 55-56)

##### Tipos de Datos

Generalmente, jQuery necesita algunas instrucciones sobre el tipo de información que se espera recibir cuando se realiza una petición Ajax. En algunos casos, el tipo de dato es especificado por el nombre del método, pero en otros casos se lo debe detallar como parte de la configuración del método:

text Para el transporte de cadenas de caracteres simples.

html Para el transporte de bloques de código HTML que serán ubicados en la página.

script Para añadir un nuevo script con código JavaScript a la página.

json Para transportar información en formato JSON, el cual puede incluir cadenas de caracteres, vectores y objetos.

Es recomendable utilizar los mecanismos que posea el lenguaje del lado de servidor para la generación de información en formato JSON.

jsonp Para transportar información JSON de un dominio a otro.

xml Para transportar información en formato XML.

A pesar de los diferentes tipos de datos de que se puede utilizar, es recomendable utilizar el formato JSON, ya que es muy flexible, permitiendo por ejemplo, enviar al mismo tiempo información plana y HTML. (Fundamentos de Jquery, 2013, pág. 57)

##### Asincronismo

Debido a que, de forma predeterminada, las llamadas Ajax son asíncronas, la respuesta del servidor no está disponible de forma inmediata. Por ejemplo, el siguiente código no debería funcionar:

var response;

$.get('foo.php', function(r) { response = r; });

console.log(response); // indefinido (undefined)

En su lugar, es necesario especificar una función de devolución de llamada; dicha función se ejecutará cuando la petición se haya realizado de forma correcta ya que es en ese momento cuando la respuesta del servidor esta lista.

$.get('foo.php', function(response) { console.log(response); }); (Fundamentos de Jquery, 2013, pág. 57)

#### Métodos Ajax de Jquery

Como se indicó anteriormente, jQuery posee varios métodos para trabajar con Ajax. Sin embargo, todos están basados en el método $.ajax, por lo tanto, su comprensión es obligatoria. A continuación se abarcará dicho método y luego se indicará un breve resumen sobre los demás métodos.

##### $.ajax

El método $.ajax es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar la petición. Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código. Puede visitar http://api.jquery.com/jQuery.ajax/ para consultar la documentación sobre las opciones disponibles en el método.

##### Utilizar el método $.ajax

$.ajax({

// la URL para la petición

url : 'post.php',

// la información a enviar

// (también es posible utilizar una cadena de datos)

data : { id : 123 },

// especifica si será una petición POST o GET

type : 'GET',

// el tipo de información que se espera de respuesta

dataType : 'json',

// código a ejecutar si la petición es satisfactoria;

// la respuesta es pasada como argumento a la función

success : function(json) {

$('<h1/>').text(json.title).appendTo('body');

$('<div class="content"/>')

.html(json.html).appendTo('body');

},

// código a ejecutar si la petición falla;

// son pasados como argumentos a la función

// el objeto jqXHR (extensión de XMLHttpRequest), un texto con el estatus

// de la petición y un texto con la descripción del error que haya dado el servidor

error : function(jqXHR, status, error) {

alert('Disculpe, existió un problema');

},

// código a ejecutar sin importar si la petición falló o no

complete : function(jqXHR, status) {

alert('Petición realizada');

}

}); (Fundamentos de Jquery, 2013, págs. 56-57)

##### Opciones del método $.ajax

El método $.ajax posee muchas opciones de configuración, y es justamente esta característica la que hace que sea un método muy útil. Para una lista completa de las opciones disponibles, puede consultar http://api.jquery.com/jQuery.ajax/; a continuación se muestran las más comunes:

**async:** Establece si la petición será asíncrona o no. De forma predeterminada el valor es true. Debe tener en cuenta que si la opción se establece en false, la petición bloqueará la ejecución de otros códigos hasta que dicha petición haya finalizado.

**cache:** Establece si la petición será guardada en la cache del navegador. De forma predeterminada es true para todos los dataType excepto para “script” y “jsonp”. Cuando posee el valor false, se agrega una cadena de caracteres anti-cache al final de la URL de la petición.

**complete:** Establece una función de devolución de llamada que se ejecuta cuando la petición es ta completa, aunque haya fallado o no. La función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHTTPRequest) y un texto especificando el estatus de la misma petición (success, notmodified, error, timeout, abort, o parsererror).

**context:** Establece el alcance en que la/las funciones de devolución de llamada se ejecutaran (por ejemplo, define el significado de this dentro de las funciones). De manera predeterminada this hace referencia al objeto originalmente pasado al método $.ajax.

**data:** Establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo foo=bar&baz=bim.)

**dataType:** Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME que posee la respuesta

**error:** Establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHTTPRequest), un texto especificando el estatus de la misma petición (timeout, error, abort, o parsererror) y un texto con la descripción del error que haya enviado el servidor (por ejemplo Not Found o Internal Server Error).

**jsonp:** Establece el nombre de la función de devolución de llamada a enviar cuando se realiza una petición JSONP. De forma predeterminada el nombre es “callback success Establece una función a ejecutar si la petición a sido satisfactoria. Dicha función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHTTPRequest), un texto especificando el estatus de la misma petición y la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma.

**timeout:** Establece un tiempo en milisegundos para considerar a una petición como fallada.

**traditional:** Si su valor es true, se utiliza el estilo de serialización de datos utilizado antes de jQuery 1.4.

**type:** De forma predeterminada su valor es “GET”. Otros tipos de peticiones también pueden ser utilizadas (como PUT y DELETE), sin embargo pueden no estar soportados por todos los navegadores.

**url:** Establece la URL en donde se realiza la petición. (Fundamentos de Jquery, 2013, págs. 58-59)

### JQuery

#### Concepto JQuery

Es un framework JavaScript, que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, framework son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar. (Álvarez, 2012)

#### Core de jQuery

El core de jQuery es la base sobre la que se trabaja para hacer cualquier cosa con jQuery. Contiene una serie de clases y métodos útiles para hacer tareas reiterativas, que vamos a necesitar en las aplicaciones. Integra desde funciones que serán útiles en cualquier script, por sencillo que sea, hasta funciones menos recurridas pero que nos facilitarán la vida a hora de hacer código limpio, corto y reutilizable.

Utilizaremos el Core para realizar cosas útiles con objetos, clases, datos, etc, pero realmente lo que más haremos será utilizar la función JQuery, que es el pilar fundamental sobre el que se basarán las aplicaciones.

Core tiene las funciones clasificadas en diversos apartados:

##### $() (La función JQuery):

Es la función principal de JQuery, que además tiene diversas utilidades según los parámetros que se le envíen. Su utilidad principal es obtener elementos de la página.

##### Accesorios de objetos:

Es una gama de funciones de diversa y variada utilidad, que sirven de utilidad para hacer cosas con objetos, tales como iterar con cada uno de sus elementos, saber su tamaño, longitud, el selector o contexto con el que se obtuvo, obtener todos sus elementos DOM que contenga, etc.

##### Trabajo con datos:

Unas funciones útiles para trabajar con datos y asociarlos a elementos, una forma de guardar información adicional a elementos de la página. También tiene diversas funciones para trabajar con colas y administrar la salida y entrada de sus elementos.

##### Plugins:

Funciones que permiten extender los elementos JQuery para incorporar nuevos métodos, algo que se utiliza habitualmente a la hora de crear plugins para añadir funcionalidades a JQuery.

##### Interoperabilidad:

Funciones que permiten que JQuery no tenga problemas de compatibilidad con otras librerías JavaScript que también suelen utilizar la función dólar $(). (Álvarez, 2012)

#### Función JQuery pasando un HTML

Una posibilidad de trabajar con la función jQuery es enviarle un string con un HTML. Esto crea esos elementos en la página y les coloca los contenidos que se indique en el string. Ojo, que el HTML tiene que estar bien formado para que funcione en cualquier navegador, esto es, que se coloquen etiquetas que se puedan meter en el BODY de la página y que todas las etiquetas tengan su cierre.

var nuevosElementos = $("<div>Elementos que creo en <b>tiempo de ejecución</b>.<h1>En

Ejecución...</h1></div>");

Esto nos creará en la variable nuevosElementos los elementos HTML que hemos especificado en el string. Luego podríamos hacer lo que queramos con ellos, como colocarlos en la página con el método appendTo(), por ejemplo de esta manera:

nuevosElementos.appendTo("body");

Veamos el código completo de una página que hace uso de este ejemplo:

<html>

<head>

<title>función jquery</title>

<script src="../jquery-1.3.2.min.js" type="text/javascript"></script>

<script>

$(document).ready(function(){

var nuevosElementos = $("<div>Estos elementos ..</b>.<h1>Título...</h1></div>");

nuevosElementos.appendTo("body");

});

</script>

</head>

<body>

<p>Esto es el cuerpo de la página, que tiene poca cosa...</p>

</body>

</html> (Álvarez, 2012)

#### Función JQuery pasando elementos

Otro posible valor que se le puede enviar a la función jQuery es un elemento o una jerarquía de elementos del DOM, para extenderlos con las funcionalidades que aporta el framework para los elementos. Por ejemplo:

var documento = $(document.body);

documento.css("background-color", "#ff8833");

Con la primera línea creamos una variable llamada documento, a la que asignamos el valor que devuelve el método $() enviando el parámetro document.body.

La variable document.body corresponde con un elemento del DOM de Javascript, que crea automáticamente el navegador y hace referencia al documento de la página. Con ello obtenemos un objeto que es el cuerpo de la página (document.body) al que le hemos agregado todas las funcionalidades del framework jQuery para los elementos. Así pues, en la línea siguiente, invocamos el método css() sobre la variable "documento", que es el propio documento de la página extendido. Por eso el método css(), que es de jQuery(), funciona sobre ese objeto.

Algo como esto no funcionaría porque estaríamos intentando lanzar un método de jQuery directamente sobre el objeto DOM sin haberlo extendido:

document.body.css ("background-color", "#ff8833");

No funcionará porque no podemos llamar a un método JQuery sobre un objeto DOM, si es que no lo hemos extendido con la función $(). (Álvarez, 2012)

#### Función jQuery pasando una función

En la función $() una última posibilidad es pasarle como parámetro una función y entonces lo que tenemos es una función callback que se invoca automáticamente cuando el DOM está listo.

En esa función podemos colocar cualquier código, que se ejecutará sólo cuando el DOM está listo para recibir comandos que lo modifiquen. Con ello, esta función nos vale perfectamente para hacer cualquier cosa dentro de la página que afecte al DOM.

Ejemplo:

$(function (){

//Aquí puedo hacer cualquier cosa con el DOM

});

Este callback con la función jQuery $() sería una abreviatura de otro método que hemos visto repetidas veces a lo largo de este manual para definir acciones cuando el DOM está listo:

$(document).ready(function(){

//Aquí puedo hacer cualquier cosa con el DOM

});

Incluso podemos hacer varios callback, para agregar distintas acciones a realizar cuando el DOM está listo, las veces que queramos, igual que cuando definíamos el evento ready() sobre el objeto document.

Podemos ver el código de una página que hace uso de la función dólar, pasando por parámetro una función.

<html>

<head>

<title>función jquery</title>

<scriptsrc="../jquery-1.3.2.min.js" type="text/javascript"></script>

<script>

$(function (){

var documento = $("p");

documento.css("background-color", "#ff8833");

});

$(function (){

var documento = $("b");

documento.css("color", "#fff");

});

//si colocase aquí este código, no funcionaría, porque el DOM no estaría listo para realizar acciones

//var documento = $("p");

//documento.css("background-color", "#ff8833");

</script>

</head>

<body>

<p><b>Párrafo</b>!!</p>

<p>Otro <b>párrafo</b></p>

</body>

</html> (Álvarez, 2012)

#### . $(document).ready()

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. JQuery permite detectar dicho estado a través de la declaración $(document).ready() de forma tal que el bloque se ejecutará sólo una vez que la página esté disponible.

El bloque $(document).ready()

$(document).ready(function() {

console.log('el documento está preparado');

});

Existe una forma abreviada para $(document).ready() la cual podrá encontrar algunas veces; sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

Forma abreviada para $(document).ready()

$(function() {

console.log('el documento está preparado');

});

Además es posible pasarle a $(document).ready() una función nombrada en lugar de una anónima:

Pasar una función nombrada en lugar de una función anónima

function readyFn() {

// código a ejecutar cuando el documento esté listo

}

$(document).ready(readyFn); (Fundamentos de Jquery, 2013, págs. 24-26)

#### Selección de Elementos

El concepto más básico de jQuery es el de “seleccionar algunos elementos y realizar acciones con ellos”. La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados. En http://api.jquery.com/category/selectors/ se puede encontrar una completa referencia sobre los selectores de la biblioteca.

A continuación se muestran algunas técnicas comunes para la selección de elementos:

Selección de elementos en base a su ID

$('#myId'); // notar que los IDs deben ser únicos por página

Selección de elementos en base al nombre de clase

$('div.myClass'); // si se especifica el tipo de elemento,

// se mejora el rendimiento de la selección

Selección de elementos por su atributo

$('input[name=first\_name]'); // tenga cuidado, que puede ser muy lento

Selección de elementos en forma de selector CSS

$('#contents ul.people li'); (Fundamentos de Jquery, 2013, pág. 27)

#### Pseudo-selectores

$('a.external:first'); // selecciona el primer elemento <a>

// con la clase 'external'

$('tr:odd'); // selecciona todos los elementos <tr>

// impares de una tabla

$('#myForm :input'); // selecciona todos los elementos del tipo input

// dentro del formulario #myForm

$('div:visible'); // selecciona todos los divs visibles

$('div:gt(2)'); // selecciona todos los divs excepto los tres primeros

$('div:animated'); // selecciona todos los divs actualmente animados (Fundamentos de Jquery, 2013, pág. 28)

#### Elección de Selectores

La elección de buenos selectores es un punto importante cuando se desea mejorar el rendimiento del código. Una pequeña especificidad, por ejemplo, incluir el tipo de elemento (como div) cuando se realiza una selección por el nombre de clase puede ayudar bastante. Por eso, es recomendable darle algunas “pistas” a jQuery sobre en qué lugar del documento puede encontrar lo que desea seleccionar.

Por otro lado, demasiada especificidad puede ser perjudicial. Un selector como #miTabla thead trth.especial es un exceso, lo mejor sería utilizar #miTabla th.especial.

JQuery ofrece muchos selectores basados en atributos, que permiten realizar selecciones basadas en el contenido de los atributos utilizando simplificaciones de expresiones regulares.

// encontrar todos los <a> cuyo atributo rel terminan en "thinger"

$("a[rel$='thinger']"); (Fundamentos de Jquery, 2013, pág. 29)

##### Comprobar Selecciones

Una vez realizada la selección de los elementos, querrá conocer si dicha selección entregó algún resul tado. Para ello, pueda que escriba algo así: if ($('div.foo')) { ... }

Sin embargo esta forma no funcionará. Cuando se realiza una selección utilizando $(), siempre es devuelto un objeto, y si se lo evalúa, éste siempre devolverá true. Incluso si la selección no contiene ningún elemento, el código dentro del bloque if se ejecutará.

En lugar de utilizar el código mostrado, lo que se debe hacer es preguntar por la cantidad de elementos que posee la selección que se ejecutó. Esto es posible realizarlo utilizando la propiedad JavaScript length. Si la respuesta es 0, la condición evaluará falso, caso contrario (más de 0 elementos), la condición será verdadera. (Fundamentos de Jquery, 2013, pág. 29)

##### Evaluar si una selección posee elementos

if ($('div.foo').length) { ... } (Fundamentos de Jquery, 2013, pág. 29)

##### Guardar Selecciones

Cada vez que se hace una selección, una gran cantidad de código es ejecutado. JQuery no guarda el resultado por sí solo, por lo tanto, si va a realizar una selección que luego se hará de nuevo, deberá salvar la selección en una variable. (Fundamentos de Jquery, 2013, pág. 29)

##### Guardar selecciones en una variable

var $divs = $('div'); (Fundamentos de Jquery, 2013, pág. 29)

#### Refinamiento y Filtrado de Selecciones

A veces, puede obtener una selección que contiene más de lo que necesita; en este caso, es necesario refinar dicha selección. JQuery ofrece varios métodos para poder obtener exactamente lo que desea.

##### Refinamiento de selecciones

$('div.foo').has('p'); // el elemento div.foo contiene elementos <p>

$('h1').not('.bar'); // el elemento h1 no posse la clase 'bar'

$('ul li').filter('.current'); // un item de una lista desordenada

// que posse la clase 'current'

$('ul li').first(); // el primer item de una lista desordenada

$('ul li').eq(5); // el sexto item de una lista desordenada (Fundamentos de Jquery, 2013, pág. 30)

##### Selección de Elementos de un Formulario

JQuery ofrece varios pseudo-selectores que ayudan a encontrar elementos dentro de los formularios, éstos son especialmente útiles ya que dependiendo de los estados de cada elemento o su tipo, puede ser difícil distinguirlos utilizando selectores CSS estándar.

:button Selecciona elementos <button> y con el atributo type='button'

:checkbox Selecciona elementos <input> con el atributo type='checkbox'

:checked Selecciona elementos <input> del tipo checkbox seleccionados

:disabled Selecciona elementos del formulario que están deshabitados

:enabled Selecciona elementos del formulario que están habilitados

:file Selecciona elementos <input> con el atributo type='file'

:image Selecciona elementos <input> con el atributo type='image'

:input Selecciona elementos <input>, <textarea> y <select>

:password Selecciona elementos <input> con el atributo type='password'

:radio Selecciona elementos <input> con el atributo type='radio'

:reset Selecciona elementos <input> con el atributo type='reset'

:selected Selecciona elementos <options> que están seleccionados

:submit Selecciona elementos <input> con el atributo type='submit'

:text Selecciona elementos <input> con el atributo type='text'

***Utilizando pseudo-selectores en elementos de formularios***

$('#myForm :input'); // obtiene todos los elementos inputs

// dentro del formulario #myForm (Fundamentos de Jquery, 2013, pág. 31)

### ¿Qué es Bootstrap?

Es una colección de herramientas gratuitas para crear sitios web y aplicaciones web, contiene HTML y CSS basados en plantillas de diseño para el texto, formas, botones, navegación y otros componentes, también contiene extensiones de JavaScript opcionales,es de código abierto y está disponible en GitHub

Bootstrap es el framework HTML, CSS y JS más popular para el desarrollo responsable y proyectos web movile-first(enfocados a dispositivos móvil). Bootstrap le ayuda iniciar en el desarrollo de aplicaciones web y sitios web. (Refsnes Data Inc, 2011)

#### Bootstrap Historia

Bootstrap fue desarrollado por Mark Otto y Jacob Thornton en Twitter como un marco para fomentar la coherencia entre los instrumentos internos. Antes de Bootstrap, se utilizaron varias bibliotecas para el desarrollo de la interfaz, lo que llevó a inconsistencias y una alta carga de mantenimiento.

Bootstrap fue lanzado como un producto de código abierto en agosto de 2011 en GitHub. En junio 2014 Manos a la Obra fue el proyecto No.1 en GitHub! (Refsnes Data Inc, 2011)

#### ¿Por qué utilizar Bootstrap?

**Enfoque mobile-first:** Desde Bootstrap 3, el framework consiste en estilos mobile-first a lo largo de toda la biblioteca

**Compatibilidad con navegadores:** Manos a la Obra es compatible con todos los navegadores populares

**Diseño web Responsive:** CSS sensible de Manos a la Obra se ajusta a ordenadores de sobremesa, tabletas y teléfonos móviles. El diseño web responsive es crear sitios web que se ajustan y alinea de acuerdo a la herramienta de comunicación que utiliza para la visualización de la misma.

**Fácil para empezar:** Con sólo el conocimiento de HTML y CSS cualquiera puede empezar a trabajar con Bootstrap. (Refsnes Data Inc, 2011)

#### ¿Qué incluye Bootstrap?

**Scaffolding:** Contiene el diseño, sistema de red, sistema de red fluida, y el diseño de respuesta.

**CSS Base:** Contiene clases de tipografía, tablas, formularios, botones, imágenes, y mucho más.

**Componentes:** Contiene componentes reutilizables: iconos, menús desplegables, NavBars, pan rallado, alertas y más.

**JavaScript Plugins:** Contiene más de una docena personalizados plugins JQuery. Puede incluir a todos, o uno a uno.

**Componentes personalizables:** Personalice componentes de Bootstrap, variables menos, y plugins JQuery para crear su propia versión. (Refsnes Data Inc, 2011)

### IBM Personal Communications

IBM Personal Communications es un paquete de emulación de terminal y comunicación con host para Microsoft Windows que presenta emulación de terminal virtual (VT), soporte de aplicaciones SNA (arquitectura de redes de sistemas), integración y conectividad. Personal Communications es un componente clave de IBM Host Access Client Package e IBM Rational Host Integration Solution.

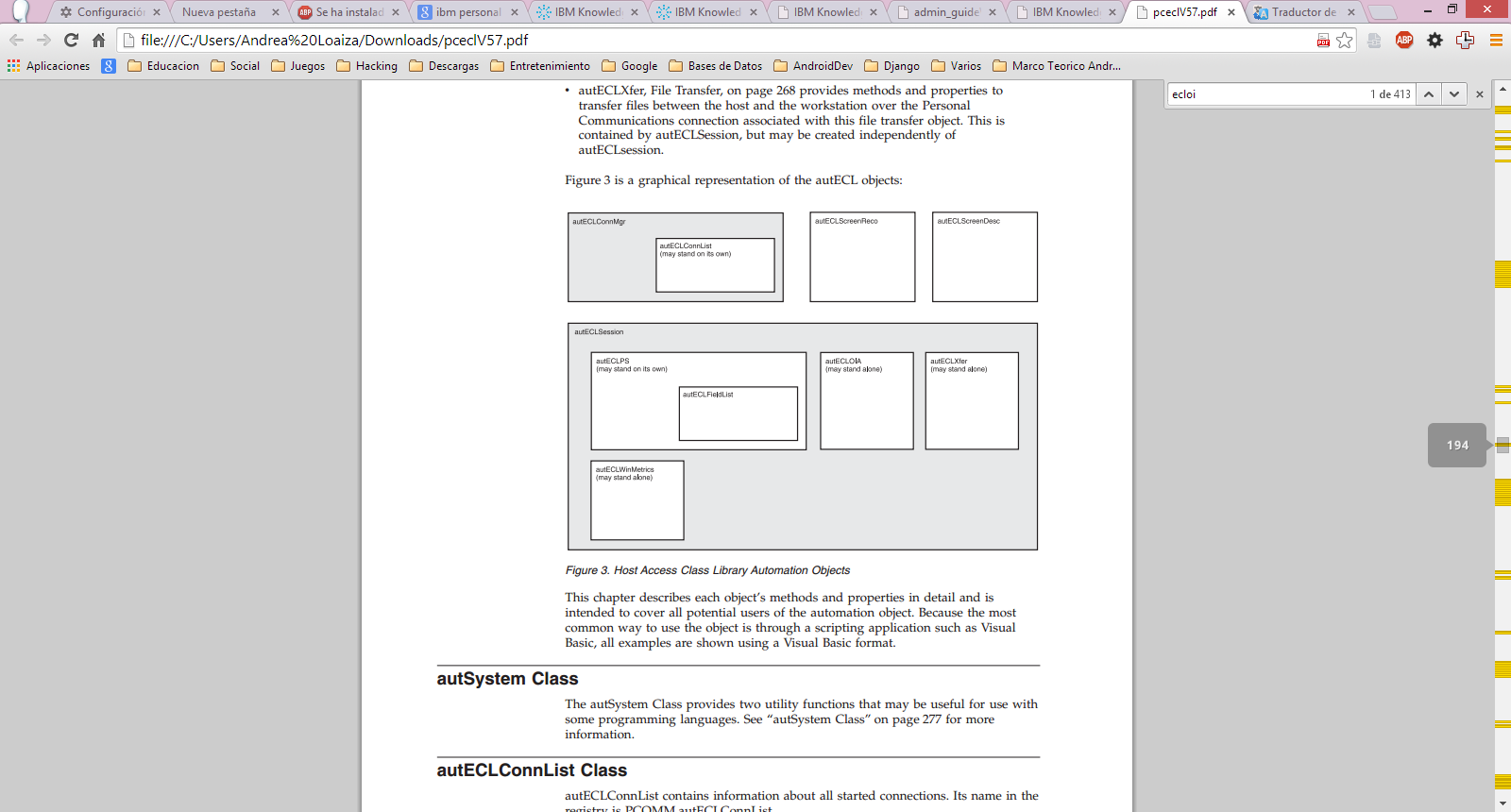
Personal Communications ofrece una plataforma de acceso tradicional a los datos y a las aplicaciones en distintos sistemas host.

* **Ofrece una solución para el acceso de host** a los sistemas empresariales de misión crítica.
* **Garantiza una red más segura** para la transmisión de datos importantes.
* **Proporciona soporte de desarrollo** de flexibilidad y ampliaciones añadidas de inversiones en redes.
* **Impulsa la productividad del usuario** gracias a programas de utilidades que permiten el ahorro de tiempo. (Host Access Class Library, 2003, pág. 14)

#### Host Access Class Automation Objects

El anfitrión Clase Acceso Biblioteca Automatización objetos permiten al personal Comunicaciones de productos de apoyo a la tecnología de automatización basada en COM Microsoft (anteriormente conocida como la automatización OLE). Los objetos ECL automatización son una serie de servidores de automatización que permiten a los controladores de automatización, por ejemplo, Microsoft Visual Básico, para acceder a los datos y la funcionalidad de programación de comunicaciones personales. Un ejemplo de esto sería el envío de claves para las comunicaciones personales espacio de presentación. Esto se puede lograr escribiendo manualmente las claves en la Ventana de Comunicaciones Personal, pero puede también ser automatizado a través del servidor adecuado de comunicaciones personales de automatización (autECLPS en este caso). Uso de Visual Basic puede crear el objeto autECLPS y luego llamar a los SendKeys método en el que objeto con la cadena que es para ser colocado en el espacio de presentación. En otras palabras, las aplicaciones que están habilitados para el control de la automatización protocolo (controlador de automatización) pueden controlar algunos de Comunicaciones Personales operaciones (servidor de automatización).

Personal Communications ofrece varios servidores de automatización para lograr esto. Estos servidores se implementan como en el mundo real, los objetos intuitivos con los métodos y propiedades que controlan la operatividad de comunicaciones personales. (Host Access Class Library, 2003, pág. 15)



*Figura 11.* Host Access Class Library Automation Objects

***Fuente:*** *Host Access Class Library*

Se describen los métodos y las propiedades de cada objeto en detalle y es destinada a cubrir todos los posibles usuarios del objeto de automatización. Debido a que la mayor parte forma común de utilizar el objeto es a través de una aplicación de scripting tales como Visual Básicos, todos los ejemplos se muestran utilizando un formato de Visual Basic.

##### autSystem Clase

La clase autSystem proporciona dos funciones de utilidad que pueden ser útiles para su uso con algunos lenguajes de programación.

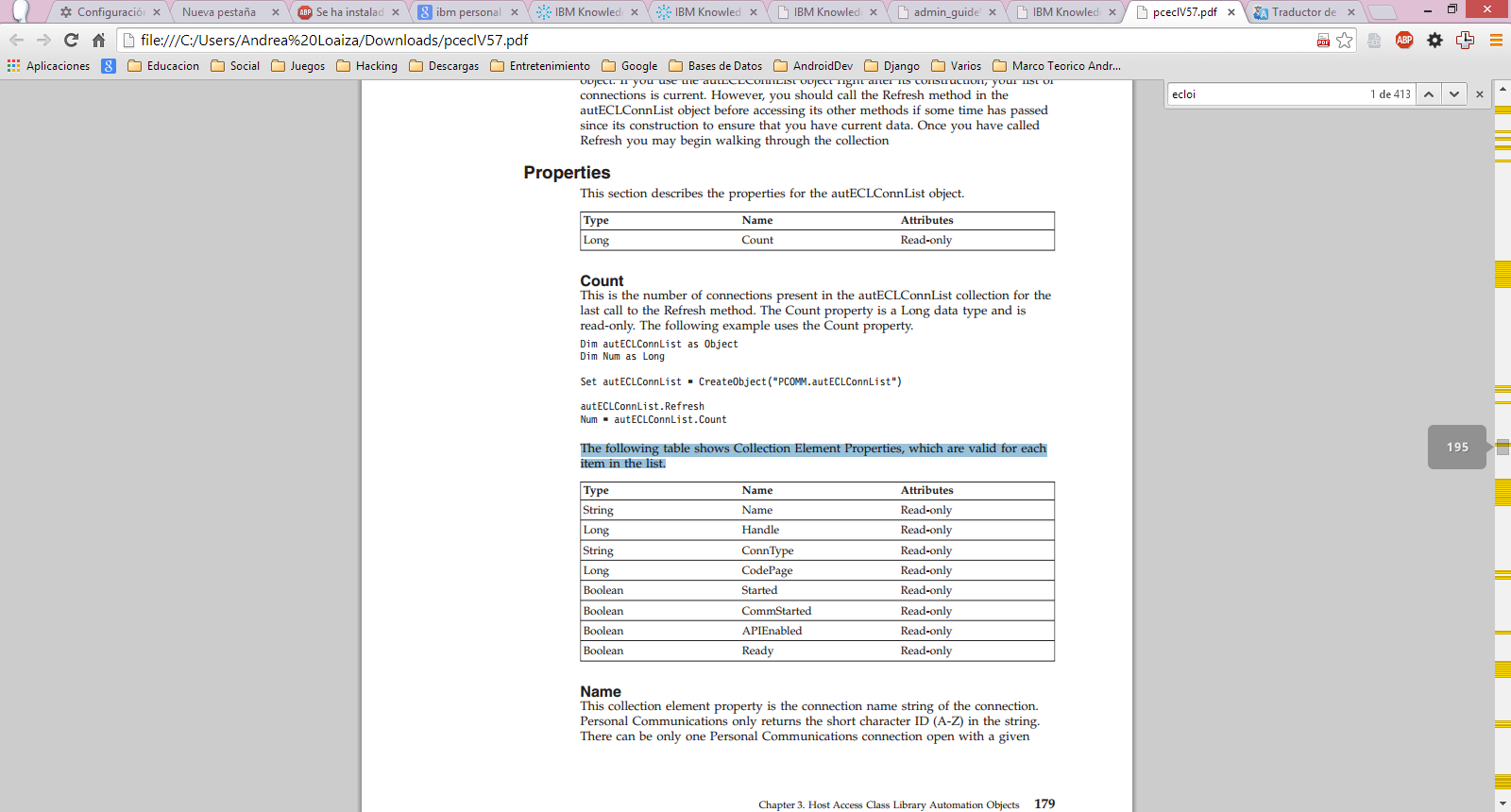
##### autECLConnList Clase

autECLConnList contiene información sobre las conexiones de todos los iniciados. Su nombre en el registro es PCOMM.autECLConnList.

El objeto autECLConnList contiene una recopilación de información acerca de las conexiones a un host. Cada elemento de la colección representa una única conexión (emulador ventana). Una conexión en esta lista puede estar en cualquier estado (por ejemplo, parado o desconectado). Todas las conexiones iniciadas aparecen en esta lista. El elemento de la lista contiene el estado de la conexión. Un objeto autECLConnList ofrece una instantánea estática de las conexiones actuales. La lista no se actualiza dinámicamente a medida que las conexiones se inician y detienen. El Método Refresh se llama automáticamente a la construcción de la autECLConnList objeto. Si utiliza el objeto autECLConnList justo después de su construcción, su lista de conexiones es actual. Sin embargo, usted debe llamar al método Refresh en el autECLConnList objeto antes de acceder a sus otros métodos si ha pasado algún tiempo desde su construcción para asegurarse de que usted tiene los datos actuales. Una vez que haya llamado Refrescar usted puede empezar a caminar a través de la colección. (Host Access Class Library, 2003, pág. 16)

###### Propiedades

Se describe las propiedades del objeto autECLConnList.



*Figura 12.* Propiedades de autECLConnList object

***Fuente:*** *Host Access Class Library*

Count

Este es el número de conexiones presente en la colección para la autECLConnList última llamada al método Refresh. La propiedad Count es un tipo de datos Long y es de sólo lectura. El ejemplo siguiente utiliza la propiedad Count:

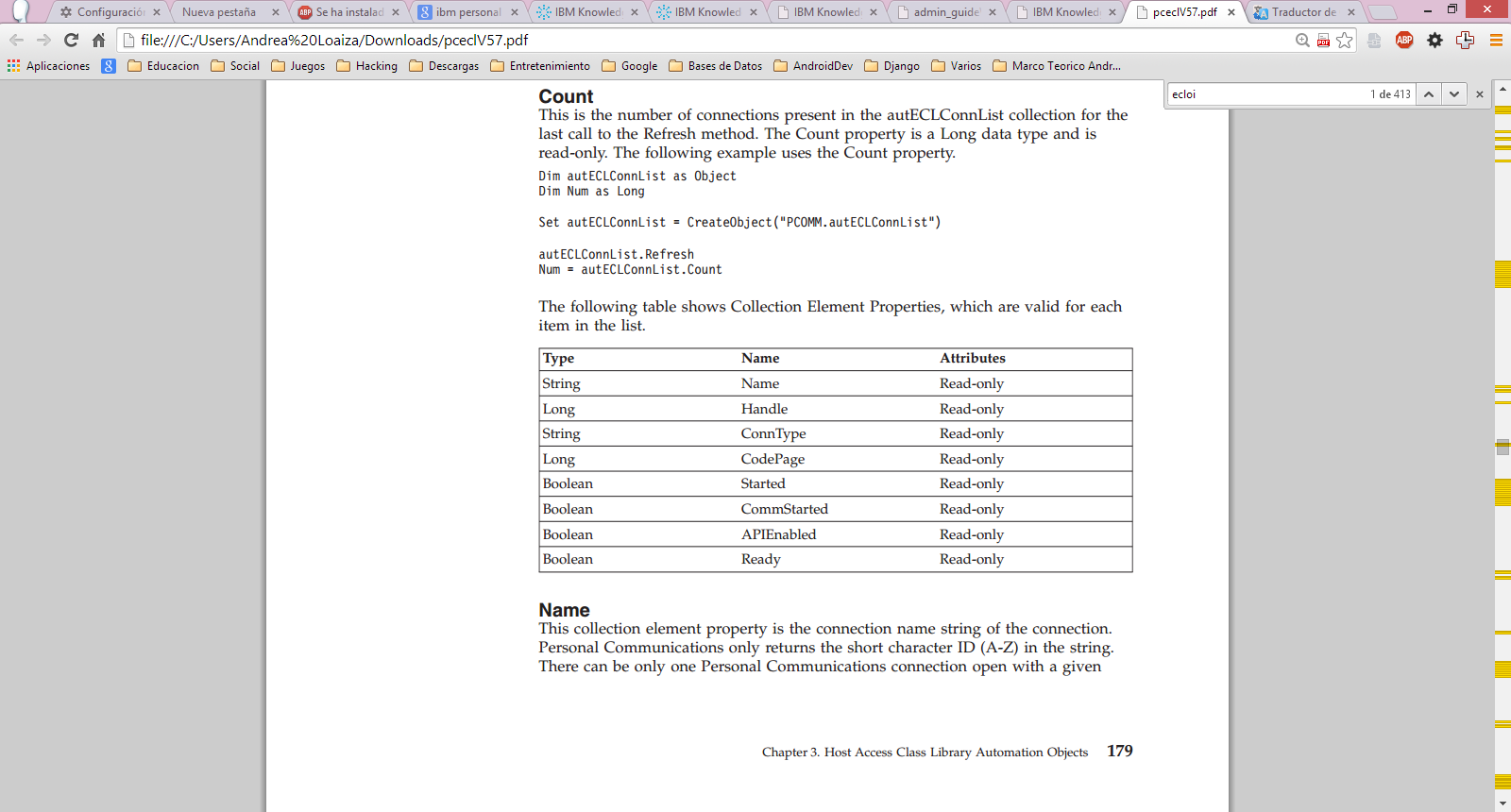
Dim autECLConnList as Object

Dim Num as Long

Set autECLConnList = CreateObject("PCOMM.autECLConnList")

autECLConnList.Refresh

Num = autECLConnList.Count

La siguiente tabla muestra Propiedades de elemento de la colección, que son válidos para cada elemento de la lista. (Host Access Class Library, 2003, pág. 17)

*Figura 13.* Elementos de la propiedad count

***Fuente:*** *Host Access Class Library*

Name

Este establecimiento situado en elemento de la colección es la cadena de nombre de la conexión de la conexión. Personal Communications sólo devuelve el ID de carácter corto (AZ) en la cadena. Sólo puede haber una conexión de comunicaciones personales abierto con un dado.

Por ejemplo, sólo puede haber una conexión "A" abierta a la vez. Nombre es un tipo de datos String y es de sólo lectura. El ejemplo siguiente utiliza el Nombre elemento de propiedad de colección.

Dim Str as String

Dim autECLConnList como objeto

Dim Num tan largo

Establecer autECLConnList = CreateObject ("PCOMM.autECLConnList")

autECLConnList.Refresh

Str = autECLConnList (1) .Name

Handler

Este establecimiento situado en elemento de la colección es el mango de la conexión. Sólo puede haber una conexión abierta con un mango dado Personal Communications.

Handle.

Dim autECLConnList como objeto

Mano Dim como Largo

Establecer autECLConnList = CreateObject ("PCOMM.autECLConnList")

autECLConnList.Refresh

Mano = autECLConnList (1) .Handle

ConnType

Este establecimiento situado en elemento de la colección es el tipo de conexión. Este tipo puede cambiar con el tiempo. ConnType es un tipo de datos String y es de sólo lectura. El ejemplo siguiente muestra la propiedad ConnType.

Tipo Dim as String

Dim autECLConnList como objeto

Establecer autECLConnList = CreateObject ("PCOMM.autECLConnList")

autECLConnList.Refresh

Tipo = autECLConnList (1) .ConnType

Tipos de conexión para la propiedad ConnType son:

Cadena Significado Devuelto

DISP3270 pantalla 3270

DISP5250 pantalla 5250

PRNT3270 3270 impresora

PRNT5250 5250 impresora

ASCII VT emulación

CodePage (Host Access Class Library, 2003, pág. 20)

# Bibliografía

Álvarez, M. (2012). *http://www.desarrolloweb.com/*. Obtenido de http://www.desarrolloweb.com/articulos/introduccion-jquery.html

Bahit, E. (2012). *Python para principiantes.* Autoedición.

Fernández Montoro, A. (2014). *Python3 al descubierto.* España: RC Libros.

Gauchat, J. (2013). *El gran libro de HTML5, CSS3 y Javascript.* Marcombo.

Gibert Ginestà, M., & Pérez Mora, O. (2012). *http://ocw.uoc.edu/.* Obtenido de http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06\_M2109\_02152.pdf

González Duque, R. (2010). *Python para todos.* España: Autoedición.

Holovaty, A., & Kaplan-Moss, J. (2010). *Django 1.0.* Autoedición.

IBM. (2003). *Host Access Class Library.*

Luján, S. (2012). *Programación de aplicaciones web.* Club Universitario.

Murphey, R. (2013). *Fundamentos de Jquery.* OpenLibra.

Pavón Maestras, J. (2012). *http://www.fdi.ucm.es/.* Obtenido de http://www.fdi.ucm.es/profesor/jpavon/web/11-Introduccion-TecnologiasWeb.pdf

PostgreSQL.org. (2 de 10 de 2010). *http://www.postgresql.org.es/*. Obtenido de http://www.postgresql.org.es/sobre\_postgresql

Refsnes Data Inc. (2011). *http://www.w3schools.com/*. Obtenido de http://www.w3schools.com/bootstrap/bootstrap\_intro.asp

Vega, J., & Vab Der Henst, C. (junio de 2011). *http://mlw.io/*. Obtenido de http://mlw.io/guia-html5/