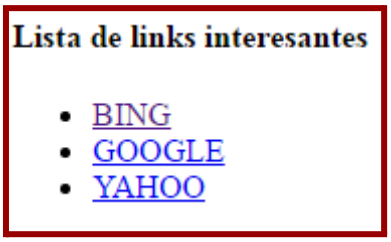
Ejercicios de Code Igniter

1. Hacer un “Hola mundo”
2. Hacer un “Hola <usuario>”, donde <usuario> es un dato que le viene (vía GET) al controlador a través del parámetro “nombre”
3. Desplegar una página con el siguiente aspecto:



En el que se muestre una lista de links, cuyos datos (tanto la URL como la etiqueta del link, vienen empaquetadas desde el controlador con un array asociativo creado “ad-hoc”. Utilizar la función “anchor” del helper “url” para hacer la tarea más sencilla.

4. Mejorar el ejercicio anterior para que la lista de url’s no provenga de datos “hard-coded” en el controlador, sino de los datos de una tabla, a la que se accederá a través de un modelo.
5. Crear una tabla **_MENUS(ID, IDP, NOMBRE, ACCION)**, destinada a contener la estructura de menús de una aplicación, donde.

ID: Es un número entero (exceptuando el cero), que es identificador único de cada menú o submenú, de nuestra aplicación.

IDP: Es el identificador del menú al que pertenece cada submenú. Si se trata de menús de primer nivel este IDP será 0,

NOMBRE: Es el nombre que aparecerá en la etiqueta de cada menú.

ACCION: Es la acción a realizar cuando se seleccione dicho menú.

6. Crear un modelo ***Menu***, con una única acción ***leerTodos()***, que lea los datos de la tabla anterior y devuelva dichos datos, en un array, con las filas de la tabla (a primer nivel), y en cada fila claves asociativas con el nombre de cada columna conteniendo el valor de las mismas.
7. Crear una función llamada “dibujarMenu($menu, $idMenu)” que recibiendo un array asociativo con la estructura definida anteriormente (en la variable $menu), y el identificador del menú que hay que dibujar, devuelva el código HTML, correspondiente a una lista del estilo:

```
<ul>
      <li>
            <a href=”accion_menu1”>EtiquetaMenu1</a>
            { … más submenús, si los contuviera }
      </li>
      <li>
            <a href=”accion_menu2”>EtiquetaMenu2</a>
            { … más submenús, si los contuviera }
      </li>
      ....etc.
</ul>
```

Cada elemento **<li>** serán los menús cuyo idp sean igual a $idMenu.

Para construir los submenús de cada uno de los demás (si los hubiera), se recomienda utilizar recursividad introduciendo otra estructura similar a ésta entre la marca **** y la marca **
** de cada menú.

PISTA1: Se recomienda utilizar ***recursividad***.

PISTA2: Utilizar un helper propio para ubicar la función

8. Crear una página de prueba que permita visualizar el menú anterior, utilizando el siguiente CSS

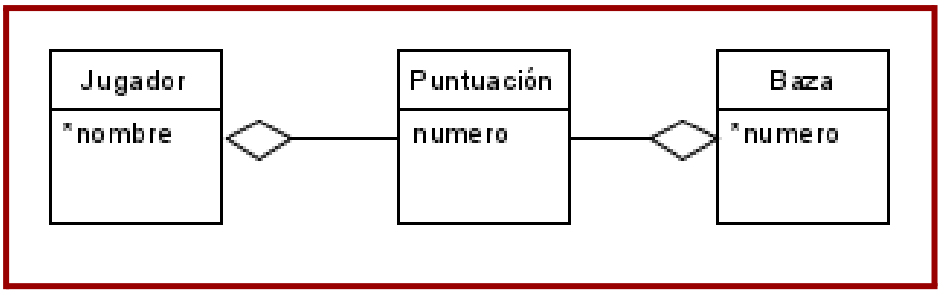
```
nav ul#navigation { margin: 0px auto; position: relative; float: left; border-left: 1px solid #c4dbe7; border-right: 1px solid #c4dbe7;}nav ul#navigation li { display: inline; font-size: 12px; font-weight: bold; margin: 0; padding: 0; float: left; position:relative; border-top: 1px solid #c4dbe7; border-bottom: 2px solid #c4dbe7;}nav ul#navigation li a { padding: 10px 25px; color:#616161; text-shadow: 1px 1px 0px #fff; text-decoration: none; display: inline-block; border-right: 1px solid #fff; border-left:1px solid #C2C2C2; border-top: 1px solid #fff; background: #f5f5f5; -webkit-transition: color 0.2s linear, background 0.2slinear; -moz-transition: color 0.2s linear, background 0.2s linear; -o-transition: color 0.2s linear, background 0.2s linear; transition: color 0.2s linear, background 0.2s linear;}nav ul#navigation li a:hover { background: #f8f8f8; color: #282828;}navul#navigation li a.first { border-left: 0 none;}nav ul#navigation li a.last { border-right: 0 none;}nav ul#navigationli:hover>a { background: #fff;}/* Drop-Down Navigation */nav ul#navigation li:hover>ul { /*these 2 styles are very important, being the ones which make the drop-down to appear on hover */ visibility: visible; opacity: 1;}nav ul#navigation ul, nav ul#navigation ulli ul { list-style: none; margin: 0; padding: 0; /*the next 2 styles are very important, being the ones which make the drop-down tostay hidden */ visibility: hidden; opacity: 0; position: absolute; z-index: 99999; width: 180px; background: #f8f8f8; box-shadow:1px 1px 3px #ccc; /* css3 transitions for smooth hover effect */ -webkit-transition: opacity 0.2s linear, visibility 0.2slinear; -moz-transition: opacity 0.2s linear, visibility 0.2s linear; -o-transition: opacity 0.2s linear, visibility 0.2s linear; transition: opacity 0.2s linear, visibility 0.2s linear;}nav ul#navigation ul { top: 43px; left: 1px;}nav ul#navigation ul li ul { top: 0; left: 181px; /* strong related to width:180px; from above */}nav ul#navigation ul li {clear: both; width: 100%; border: 0 none; border-bottom: 1px solid #c9c9c9;}nav ul#navigation ul lia { background: none; padding: 7px 15px; color: #616161; text-shadow: 1px 1px 0px #fff; text-decoration: none; display: inline-block; border: 0 none; float: left; clear: both; width: 150px;}
```

9. Realizar el mismo CRUD de empleados creado con el framework propio anterior, utilizando CodeIgniter
10. Realizar (siguiendo la guía de desarrollo sistemático) un CRUD completo de **empleados** en el que se registren los datos de nombre, apellidos, teléfono y **ciudad** de nacimiento (siendo esta última un bean diferenciado, y su borrado debe borrar en cascada todos los usuarios de dicha ciudad). Asimismo se deberán registrar los **lenguajes de programación** que el empleado conozca. Por sencillez, se asume que el nombre de cada empleado es único en el sistema.

    Mejorar el programa anterior para que sólo los usuarios autenticados (añadir un campo password al empleado), puedan hacer todas las operaciones CUD. Los usuarios anónimos sólo podrán ver y realizar las operaciones “R”. Desarrollar casos de uso “login” y “logout” a tal efecto.

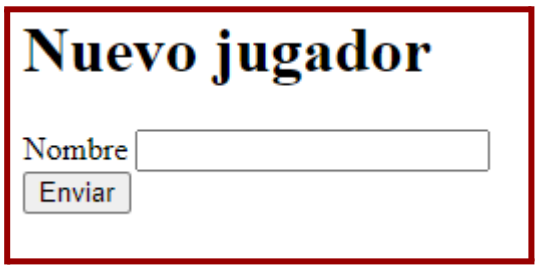
    Mejorar el programa para que se pueda indicar quién es el jefe de cada empleado. Cada empleado tendrá uno (y sólo un jefe), que será a su vez otro empleado. Modificar el caso “C” y el caso “R” de empleado, para que en este último se diga en una columna el nombre del jefe de un empleado. NOTA: Un empleado no puede ser jefe de sí mismo. (NOTA: Investigar los alias de RedBeanPHP y el método “form_dropdown(...) de CodeIgniter)

11. Dado el siguiente modelo.



Realizar un programa que permita anotar el desarrollo de una partida de cartas, en el que hay Jugadores de los que deseamos saber su nombre y bazas en las que cada jugador registra una puntuación que ha obtenido

- En cada baza se debe ver la puntuación actual y la puntuación acumulada (suma) de las bazas anteriores.
- El jugador que menos puntuación **tenga acumulada** en cada baza deberá estar resaltado (porque será el ganador provisional de esa baza)
- Si un jugador supera un número de puntos determinado / hard-coded (p.ej 200) también será resaltado (en otro color) porque habrá perdido la partida, y ya no debería aparecer en las siguientes bazas (ni en C baza, ni en las filas de R baza posteriores a las que fue eliminado.
- Desarrollar los siguientes casos de uso
  - C jugador



  - R total

	PEPE	ANA	JUAN	MARÍA
Baza1	10 / 10	8 / 8	0 / 0	20 / 20
Baza 2	150 / 160	0 / 8	10 / 10	5 / 25
Baza 3	50 / 210	20 / 28	0 / 10	100 / 125

  - C baza