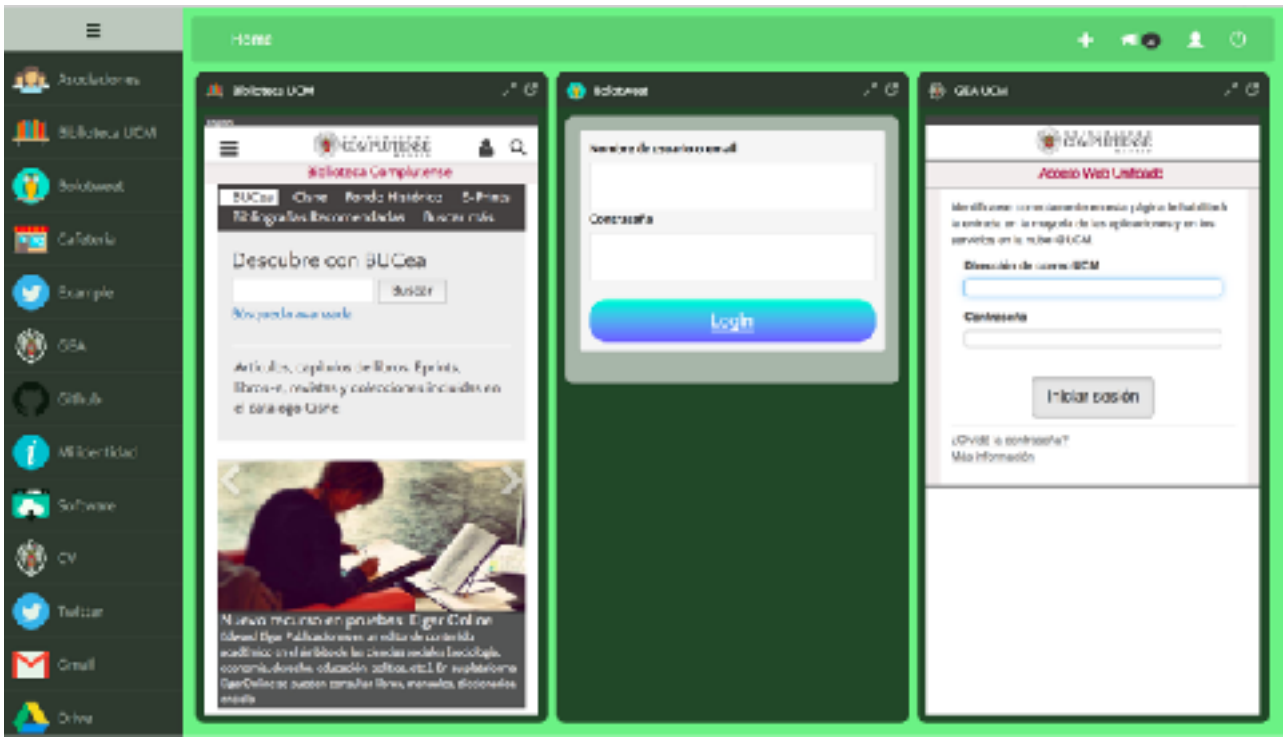
MEMORIA DEL PROYECTO

Hippocampus

Development : 22 de febrero - 4 de junio de 2017

HIPPOCAMPUS



Iván Gulyk
Blanca de la Torre
Marta Pastor Puente
Pablo García de los Salmones
Guillermo Monserrate Sánchez

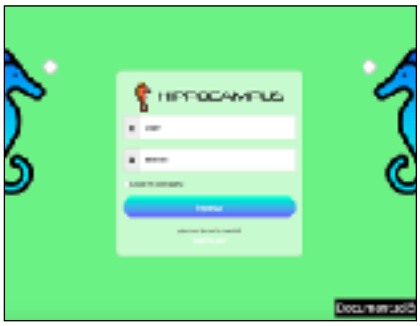
Índice

Introducción

Nuestra aplicación web busca conectar todos los servicios online ofertados por la universidad como pueden ser la plataforma Moodle, el correo electrónico académico, los srevicios de préstamo de la biblioteca o toda la información referente a los servicios disponibles en la facultad.

A través de Hippocampus, tanto alumnos como profesores podrán acceder a todos esos servicios sin necesidad de andar cambiando de pestaña en el navegador e iniciando sesión tan solo una vez.

Nuestra aplicación tiene como finalidad ofrecerle al usuario un servicio cómodo y sencillo, mostrando toda la información relevante de un vistazo, así como notificaciones actualizadas de los servicios a los que se encuentre conectado, para no perderse ninguna novedad.

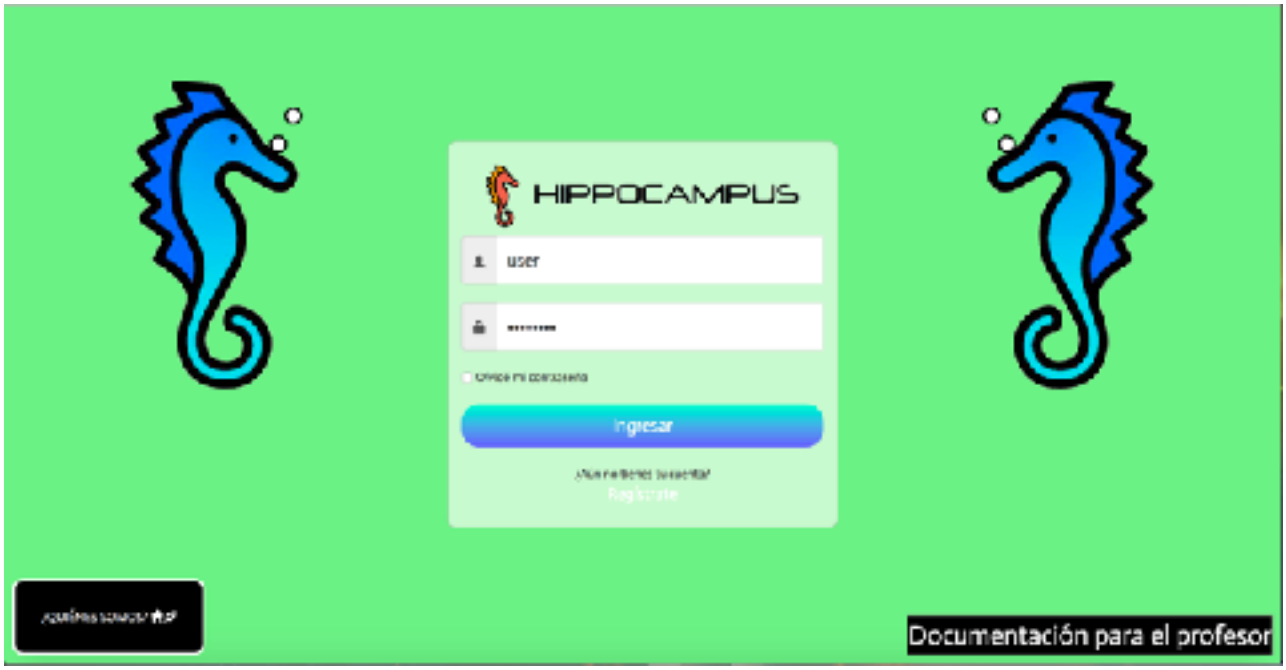


Anteriormente, Google poseía una aplicación que permitía algo parecido a la funcionalidad de nuestro proyecto. Esa es la aplicación que más se asemeja a nuestra página web. Además, el hecho de que Google cerrase dicha herramienta, hace que nuestra página web tenga bastante atractivo para aquellos que usaban la de Google.

Descripción detallada de la aplicación

Como hemos explicado, ésta aplicación conecta servicios online y te permite manejarlos a la vez a través de una misma interfáz. A continuación explicaremos con ayuda de imágenes las distintas funcionalidades que esto tiene:

Nuestra página cuenta con un formulario de login muy trabajado.



La misma página de inicio tiene una opción de “¿Quiénes somos?” en la que se muestran los miembros del grupo que ha desarrollado el proyecto.

Además, cuenta con otro enlace a la Documentación necesaria para el profesor.

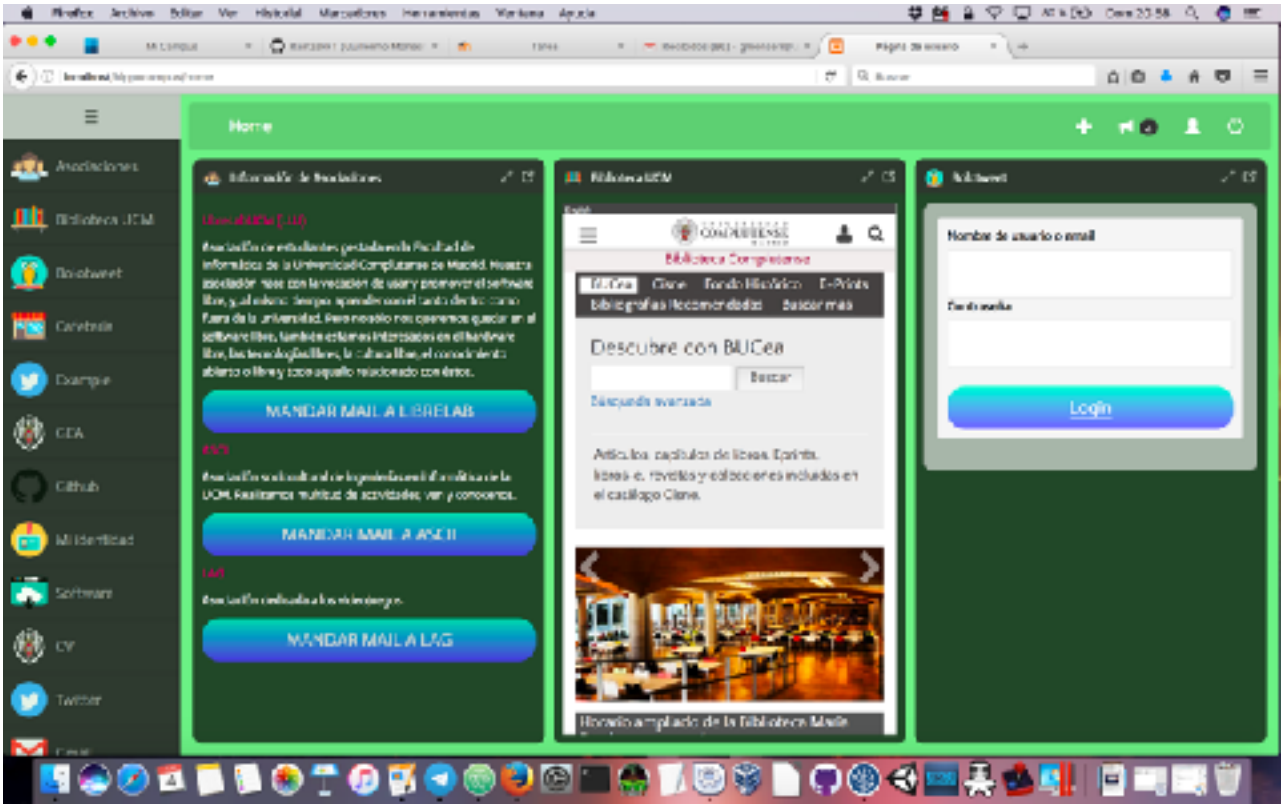
La aplicación cuenta con tres tipos de usuarios: root, admin y user.

El usuario root es el usuario de mayor privilegio, y nunca puede perder su rol.

El usuario admin tiene mismos privilegios que root, pero su rol es cambiable.

El usuario user no puede administrar la página como los otros dos usuarios, solamente se beneficia de sus funcionalidades.

Si entramos a la página como usuario user, tendremos la siguiente vista:

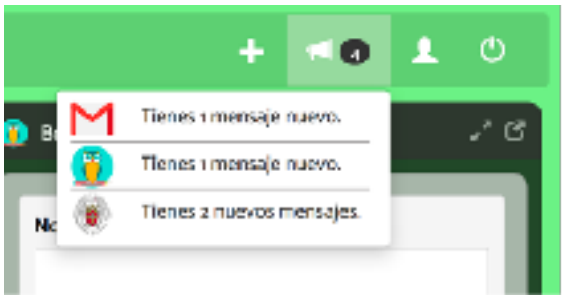


Como usuario user tenemos varias funcionalidades:

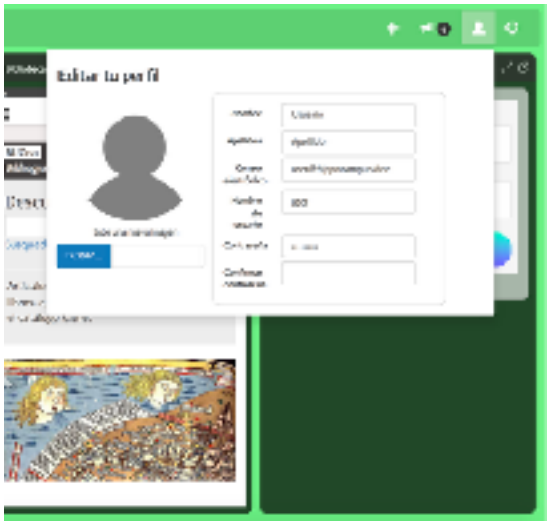
-Podemos cambiar entre los módulos que queremos que se muestren.

-Podemos cambiar cuentas columnas queremos que se muestren:

-Podemos mostrar las notificaciones:



-Podemos ver nuestro perfil de usuario y modificarlo:



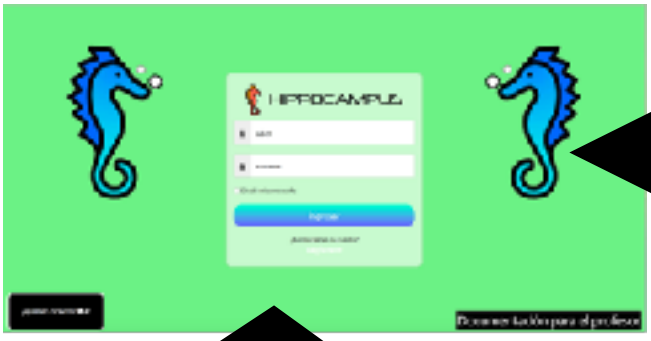
-Y podemos salir de nuestra cuenta en cualquier momento.

La funcionalidad de los usuarios admin y root es que pueden acceder al menú de administración:

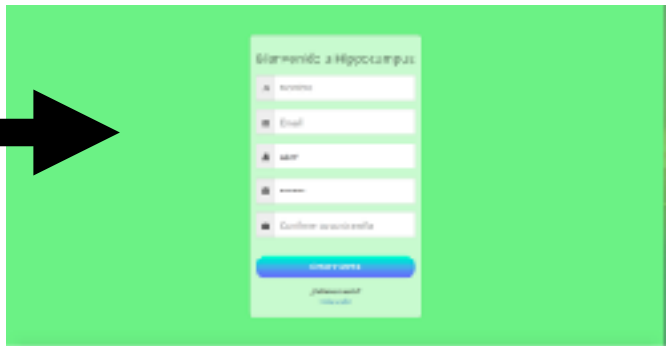
En dicho menú, el administrador podrá crear usuarios y editar usuarios.

Mapa de navegación del sitio:

Login

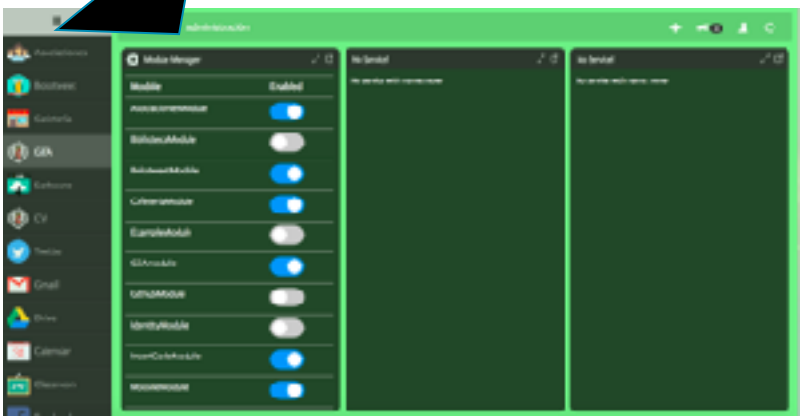


Registro



Miembros

Vista de usuario



Menu de administración

Sólo admin y root

Servicios conectados

Nuestra aplicación permite al usuario conectar con su cuenta de Hippocampus tantos servicios como quiera de la siguiente relación:

Gestión académica

- Gestión de identidad (gestión de datos personales, constraseñas y cuentas bancarias)
- Geanet (gestión de notas y calificaciones)
- Campus virtual (Moodle)
- Biblioteca

Google

- Gmail
- Google Drive
- Google Calendar
- Google Clasroom

Mensajería

- Bolotweet

Redes sociales

- Twitter

Aplicaciones informáticas

- Github
- Software (enlaces de descarga de programas con licencia ofrecida por la universidad)

Servicios de la Facultad

- Asociaciones
- Cafetería
- Secretaría

Otros

- Ajustes
- Sobre nosotros

Los módulos son el fuerte de la aplicación. Son los que ofrecen los servicios para los usuarios.

Se colocan en la carpeta *modules* situada en la raíz del proyecto.

Dentro de la carpeta *modules* hay una carpeta por cada módulo, que contine todos los archivos que el módulo pueda necesitar.

Dentro de la carpeta del módulo debe existir un archivo llamado exactamente igual que la carpeta, terminado en *.php*. Este archivo debe contener una clase que se debe llamar igual que la carpeta del módulo, y que debe extender de *HC_Module*.

El constructor de la clase principal del módulo recibe una instancia del objeto **Hippocampus**, que deberá pasarle al constructor de la clase padre.

Un ejemplo de un módulo:

modules/ExampleModule/ExampleModule.php:

```
<?php

class ExampleModule extends HC_Module {
  public function __construct($hc) {
    parent::__construct($hc);
  }
}
```

Ese sería el esquema básico de un módulo que no hace nada. Las funcionalidades de los módulos se añadirían en el constructor y en funciones específicas dentro de la clase.

Los módulos pueden acceder a la instancia de Hippocampus con *$this->hc*. Desde este objeto pueden obtener instancias a la base de datos, y a cualquier otra parte de la aplicación.

Añadir elementos al menú lateral

Los módulos pueden implementar la función *onCreatingSidebar*, que se llamará durante el proceso de creación del menú. Recibirá un parámetro con un array de elementos. Este parámetro se pasa por referencia, por lo que el módulo podrá editar el array, añadiendo nuevos elementos o borrando los existentes.

Un elemento consta de un array con 3 claves: *icon*, *text*, e *id*. En base a estos valores se creará posteriormente el elemento en el menú.

El id de un elemento del menú debe ser único (se recomienda incluir el nombre del módulo como parte del id para disminuir las posibilidades de choque entre id). El id se asociará posteriormente con una de las ventanas de la vista de usuario, en la que se mostrará el contenido adecuado según el id asociado.

Añadir código en la sección de la página

Implementando la función *onCreatingMetacode*, los módulos reciben un array por referencia con los elementos del código de la sección head de las páginas. En este array se pueden insertar elementos con referencias a archivos de javascript y de hojas de estilo. Además, se puede insertar código directamente, por ejemplo, insertando un elemento con el código entre las etiquetas *<script>* y *</script>*.

Aquí un módulo podría importar librerías o estilo css de fuentes externas, o de la propia carpeta del módulo.

Configurar contenido de las ventanas

La vista principal del usuario (el **userview**) tiene un número configurable de ventanas. Su contenido se rige por un identificador. Los módulos pueden registrarse creando un hook que será llamado cuando el contenido de una de estas ventanas se tenga que mostrar.

Para ello, en el contructor de la clase del módulo, se ejecutará el siguiente código:

```
$this->registerWindowCallback('window_identifier',
'callback_function');
```

Siendo **window_identifier** el identifiador (por ejemplo, el id de un elemento del menú lateral), y **callback_function** un string con el nombre de una función pública presente en esa misma clase que se llamará cuando se tenga que mostrar contenido en una ventana.

La función del callback puede, opcionalmente, recibir un array de campos, para personalizar más el contenido a mostrar al usuario.

Esta función debe devolver un array asociativo con 2 claves: *html* y *title*. El contenido se verá en el cuerpo y la cabecera de la ventana, respectivamente.

Para ampliar la funcionalidad de los módulos y facilitar la transición entre ventanas, se puede incluir un elemento html con el atributo *data-updatewindowboxservice*, dándole el valor de un identificador de ventana al que se cambiará cuando se haga clic en este elemento.

Un módulo puede configurar tantos callbacks como se desee, llamando varias veces a la función *registerWindowCallback*, indicando cada vez el identificador y el nombre de la función del callback. Por ello, un módulo puede configurar varias vistas distintas, y permitir al usuario navegar entre estas vistas creando elementos con el atributo *data-updatewindowboxservice*.

Para permitir una navegación más facil, se pueden añadir datos extra en forma de atributos de la forma *data-cbdata-Clave*. Clave será la clave del array de campos extra que se recibirá en la función del callback de la vista. Se pueden tener tantos valores como se desee, usando diferentes claves para cada valor.

Importante: la clave deberá empezar por mayúscula. Si no es así, la primera letra se convertirá a mayúscula automaticamente.

Por ejemplo:

```
<p data-updatewindowboxservice="newview" data-cbdata-Key1="value1"
data-cbdata-Key2="value2">Click to go to new view!</p>
```

Registro de llamadas a la api

Los módulos pueden capturar llamadas específicas a la api de la aplicación para procesarlas en funciones propias.

Incluyendo el siguiente código en el constructor de la clase del módulo:

```
$this->registerApiCallback('api_action', 'api_function_callback');
```

El módulo registrará la función **api_function_callback** para que procese todas las llamadas a la api con acción **api_action**. La función de callback deberá devolver un array con las opciones que la llamada a la api espere obtener en forma de un array en json.

La función registerApiCallback admite un tercer argumento, para pasar información adicional, que luego se recibirá en la función del callback.

La función del callback recibe 3 argumentos: **$identifier**, **$data** y **$cbdata**.

- **identifier**: identificador por el que esta función está procesando la llamada a la api. Es el valor del primer argumento de *registerApiCallback*. Útil si se usa una misma función para procesar varias acciones de la api.

- **$data**: los datos de la petición de la api en un array asociativo, obtenidos de $_POST.

- **$cbdata**: El contenido del tercer argumento de *registerApiCallback*.

Configuración inicial durante la instalación del módulo

En ocasiones es posible que un módulo necesite ejecutar un código la primera vez que se ejecuta. Por ejemplo, para configurar alguna clave, o para crear una tabla específica en la base de datos.

Para ello, los módulos tienen la posibilidad de implementar la función *setup*. Esta función debe ser pública y estática, ya que es llamada antes de instanciar la clase del módulo. Al ser estática, no tiene acceso a $this->hc, por lo que recibe la instancia de Hippocampus como argumento.

Esta función deberá devolver true para que el módulo cargue correctamente. Si esta función falla devolviendo false, el módulo será ignorado.

Arquitectura de la aplicación

Explicación de la Base de Datos

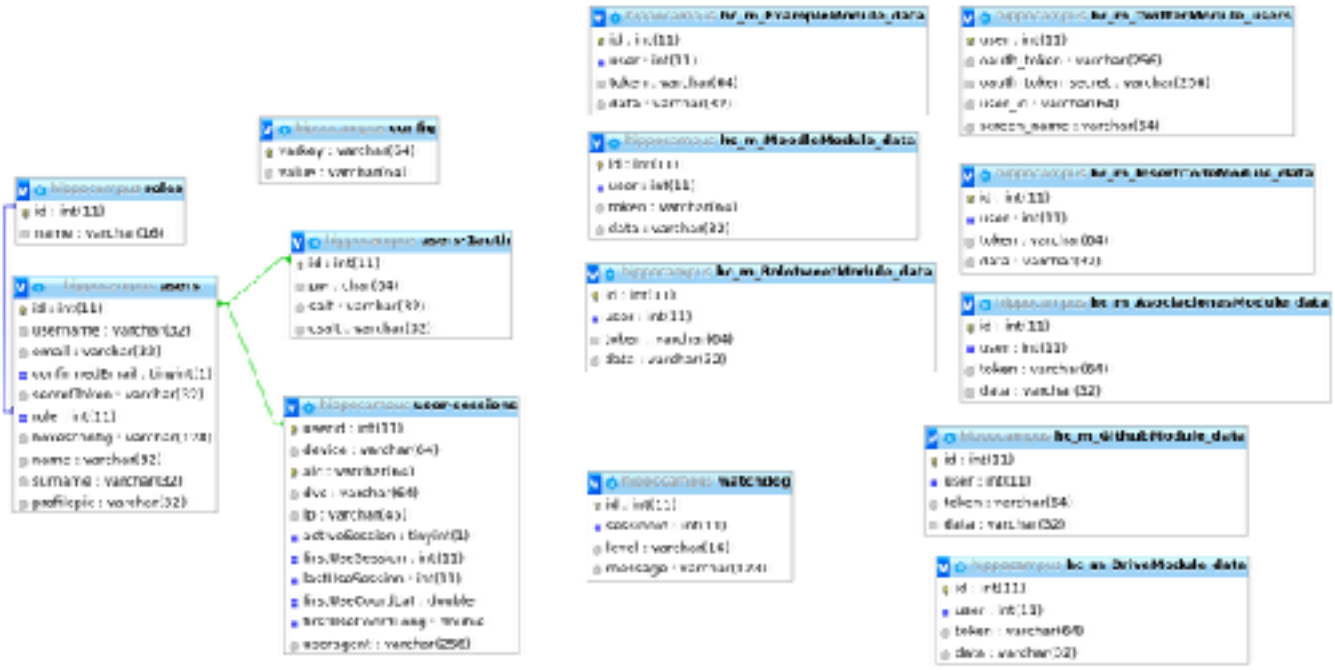


Tabla `config`

Usada para guardar datos de configuración en forma de clave/valor.

Las claves se nombran con un sistema de namespaces por puntos, categorizando cada clave en categorías y subcategorías.

Las categorías usadas actualmente son:

- site: configuración básica de la aplicación
- db: información de la base de datos. En `db.version` guardamos la versión de la base de datos desde que iniciamos el proyecto.
- module: información y configuración de los módulos.

La entrada `db.version` nos ha resultado muy útil durante la etapa de desarollo. Nos ha permitido trabajar todo el equipo con una misma base de datos, que podía ser exportada, incluida en el proyecto, y compartida a través del repositorio de git. Cuando el valor de `db.version` en la base de datos no coincidía con el presente en el código, el código actualizaba la base de datos usando el dump exportado previamente.

Para los módulos, siempre existen al menos 3 entradas por cada módulo (sustituyendo `[ModuleName]` por el nombre del módulo):

- `module.[ModuleName].module`: contiene el nombre del módulo, que debería ser igual a `[ModuleName]`.
- `module.[ModuleName].setup`: este valor se cambia a true una vez el módulo ha sido instalado correctamente, y se ha ejecutado la función `setup` del módulo (si la implementa). Si el módulo existe, pero el valor de este campo no es true, entonces se llama a la función setup del módulo para que realice su configuración inicial.
- `module.[ModuleName].enable`: su valor se alterna entre true y false, según si el módulo está activado y desactivado.

Si un módulo necesita guardar valores en formato clave/valor, se recomienda seguir la siguiente convención:

```
module.[ModuleName].[ConfigVariable]
```

Siendo `[ConfigVariable]` la clave del valor a guardar, evitando `setup`, `module` y `enable`.

Tabla `users`

En esta tabla se guardan los datos de los usuarios registrados. El `id` de usuario (PRIMARY_KEY) se genera automatimanete de forma incremental al insertar nuevos usuarios.

En esta tabla tenemos varias columnas que no estamos usando actualmente, pero que introdujimos en un principio, y que nos facilitarán en un futuro ampliar la funcionalidad de la web en este aspecto.

Tabla `users-lauth`

Los datos de autenticación de un usuario se guardan en esta tabla. Se guarda la id del usuario al que referencia eestos datos de sesión, la contraseña hasheada, un salt de servidor y otro salt de cliente. El proceso de login se explica con más detalle en Security.

Tabla `user-sessions`

Cuando un usuario se loguea en la web, se crea una entrada en esta tabla con los datos necesarios para mantener la sesión abierta para el usuario de forma segura.

Creando varias entradas para un mismo usuario en esta tabla, un usuario puede tener sesiones abiertas en varios dispositivos al mismo tiempo.

Algunas columnas de esta tabla no están siendo usadas actualmente, debido a temas de privacidad y recolección de datos.

Tabla `roles`

Los roles están definidos en esta tabla. Hay 3 roles: root, administrator y user. Los roles están identificados por una id incremental según la cantidad de permisos disminuye. Esto permite comprobar si un usuario tiene ciertos permisos, según si el id de su rol es menor o igual a cierto valor.

Tabla `watchdog`

Esta tabla se usa para guardar un registro de los cambios en la web, por ejemplo, cambios de configuración, activación y desactivación de módulos y temas, etc...

Tablas `hc_m_[ModuleName]_[Table]`

Los módulos que deseen crear tablas para guardar datos deberían seguir un patrón similar a `hc_m_[ModuleName]_[Table]` para evitar colisiones de nombres. Cada módulo que lo necesite creará la tabla en la función `setup` de su clase, con las colunas correspondientes.

Cambios realizados durante el proyecto

Durante el proyecto, nuestra idea de la aplicación siembre se ha mantenido.

Sin embargo, a medida que avanzaba el proyecto e íbamos aprendiendo cosas nuevas, nuestras ideas sobre ciertas funcionalidades iban evolucionando.

Por ejemplo, al principio pensabamos que lo mejor sería que el usuariose administrase sus propios módulos. Sin embargo, caímos en la cuenta de que un usuario administrador debería ser el encargado de mantener los módulos y administrarlos (o en caso de que hubiese algún error, arreglarlo).

También, gracias a jquery, javascript y ajax, pudimos crear los módulos que se actualizasen sin problema y que se manifestasen sólo cuando el usuario lo decidiese, creando así una aplicación web mucho más eficiente para el usuario.

Instruciones de instalación

Se requiere de un servidor web, PHP7, y un servidor MySql (o MariaDB). El servidor web deberá tener activada la opción `RewriteEngine`.

Se tendrá que crear una base de datos **hippocampus**, y un usuario **hippocampus** con contraseña **hippocampus** que tenga permisos para la base de datos anterior. Estos datos están en `core/config.php` en caso de querer ser cambiados.

Primero, clonar el repositorio en una carpeta accesible públicamente. Modificar en el archivo *.htaccess*, la linea

```
RewriteBase /AW/Hippocampus/
```

Cambiando `/AW/Hippocampus/` por la ruta sobre la que funcione la web (sin el dominio).

Por ejemplo, para la ruta: `localhost/HC`, RewriteBase tendría que tener el valor `/HC/`

Si en vez de Apache se usa otro navegador, se tendría que configurar una regla similar. Por ejemplo, en el caso de Caddyserver, el código a usar con la misma funcionalidad que el *.htaccess* sería:

```
rewrite {
  to {path} {path}/ /index.php?p={uri}
}
```

Tras tener todo configurado, accedemos con el navegador a la página. La primera vez que se acceda se verá un mensaje "*Database updated, please refresh*". La base de datos tendrá las tablas y las entradas necesarias para funcionar. Tras volver a entrar en la página, se verá la página principal de la web.

Inicialmente hay 3 usuarios con los que poder acceder: * root * admin * user

Los tres acceden con la misma contraseña: `Qwerty1!`

Se pueden registrar más usuarios siguiendo el botón para registrase.


HIPPOCAMPUS