Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación web para la gestión de un servicio de préstamo de material.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jose Boix Ruiz

Tutor: Vicente Luis Atienza Vanacloig

Diciembre 2014

Resumen

Se ha desarrollado una aplicación web en PHP (framework Yii) totalmente dinámica para la gestión de préstamo de material para la asignatura de Producción Multimedia. Los encargados de administrar la aplicación podrán crear categorías, donde se les dará la posibilidad de crear dinámicamente formularios y posteriormente registrar sus productos. En cambio, los alumnos podrán reservar los productos anteriormente registrados y llevar un amplio control de las reservas realizadas. Debido a los requisitos de comodidad y usabilidad que se plantean hoy en día, la aplicación podrá ser utilizada desde cualquier dispositivo conectado a internet, con el único requisito de disponer un navegador web.

Palabras clave: Aplicación, PHP, MySQL, Producción Multimedia, gestión, dispositivo.

Abstract

A fully dynamic web application in PHP has been developed towards the management of loan material for the Multimedia Production subject. The responsible of managing the web application can create categories, in which they will have the opportunity of creating forms dynamically and then register their products. By the way, students may reserve the products previously registered and take a major control over the products already booked. Due to comfort and usability requirements which arise nowadays, the application may be used from any internet connected device, with the only requirement of having a web browser.

Keywords: Application, PHP, MySQL, Multimedia Production, management, device.

Tabla de contenidos

Tabla de ilustraciones

1. Introducción

1.1 Motivación y Justificación

Las páginas web dinámicas son aquellas en las que la información presentada se genera a partir de una petición del usuario. A medida que la tecnología avanza, los hábitos de la sociedad evolucionan y van surgiendo nuevas necesidades. Internet nos proporciona la herramienta perfecta para satisfacer las nuevas necesidades que hoy en día se presentan, es por ello de la importancia de las páginas web dinámicas.

Satisfacer las necesidades de los alumnos y optimizar el proceso de gestión del material para la universidad, se han convertido en las principales motivaciones a la hora del desarrollo. Gracias a los requisitos que planteaba el proyecto se ha dado la posibilidad de aumentar los conocimientos en el framework Yii basado en el lenguaje PHP, muy demandado actualmente en el mundo laboral.

1.2 Objetivos del proyecto

El objetivo principal del proyecto es el desarrollo de una aplicación que permite a los alumnos reservar el material ofrecido por la asignatura. Aparte de esta descripción general del proyecto, se definieron una serie de objetivos primordiales que la aplicación debería satisfacer. Estos objetivos son:

- La posibilidad de crear usuarios con distintos roles y poder administrarlos.
- Permitir registrar objetos en la aplicación y poder agrupar los objetos en lotes.
- Posibilitar la opción de administrar dinámicamente los objetos registrados.
- Desarrollar un formulario de contacto para poder establecer una conexión con el administrador.
- Establecer un método de sanción según la infracción cometida en el periodo de reserva.

Una vez analizado los primeros objetivos en profundidad, se planteó con el tutor la posibilidad de mejorar el proyecto añadiéndole más objetivos para que no fuese una pequeña aplicación con exclusividad para una determinada asignatura. Es por ello que se añadieron los siguientes objetivos específicos:

- Estructurar los objetos en categorías para establecer un filtro de búsqueda.

- Poder crear formularios de manera dinámica, para registrar cualquier tipo de objeto.
- Establecer un sistema de tablas para la administración de reservas, objetos, lotes y usuarios.
- Desarrollar un sistema de búsqueda para cada elemento de la tabla de administración.
- La posibilidad de poder utilizar la aplicación en todo tipo de dispositivos, no solo en ordenador, como principalmente se había planteado.
- Utilizar algoritmos de encriptación para mejorar la seguridad de la aplicación.

Tanto los objetivos planteados por el tutor, como los objetivos específicos posteriormente planteados han sido desarrollados en su totalidad, conformando la base principal de la aplicación.

1.3 Estructura de la memoria

El contenido del documento está estructurado en ocho capítulos, en los cuales se pretende explicar la base del proyecto con el fin de comprender el trabajo realizado. En el primer capítulo consta la introducción, subdivido en tres apartados donde se explica cuál es la principal motivación y justificación por la cual se ha realizado el proyecto, cuales son los principales objetivos que se propusieron y como se ha estructurado el siguiente documento, con el fin de facilitar una rápida localización de los distintos apartados.

En el segundo capítulo se definen los principales conceptos básicos, para poder comprender en que entorno del proyecto nos encontramos. Para ello definiremos que es una página web dinámica y porqué se ha elegido este tipo de página web para el proyecto.

En el tercer capítulo explicaremos las tecnologías utilizadas y porque hemos hecho uso de ellas para la realización de este proyecto. Este apartado es fundamental debido a que una mala elección a la hora de utilizar una determinada tecnología puede desestructurar los cimientos de nuestro proyecto.

En el cuarto capítulo describiremos los escenarios contemplados para la realización del diseño de nuestra base de datos, con el fin de tener claro cuáles son las relaciones y atributos necesarios para almacenar de forma óptima los datos que incorporará nuestro proyecto.

El quinto capítulo es el más extenso, debido a que incorporamos el diseño e implementación de nuestra aplicación web. Para ello, se ha dividido este capítulo en distintos apartados entre los que destacamos: El diagrama de clases para el diseño de la base de datos donde se describirá el contenido de cada tabla, la metodología utilizada en el desarrollo, el diagrama de casos de uso y la interfaz de usuario.

En el sexto capítulo se demuestra que todos los objetivos del proyecto han sido cumplido y además se indican un serie de mejoras que se podrían desarrollar en la aplicación en un futuro.

En el séptimo capítulo se detalla las conclusiones obtenidas en el proyecto y en el octavo y último capítulo aparecen todas las referencias bibliográficas y los sitios webs visitados, que han aportado su granito de arena a la hora de realizar el proyecto.

Por último, en el anexo se incluye un manual. En el manual se detalla la instalación y configuración apropiada del proyecto mediante la herramienta XAMPP, para que el administrador no tenga problemas a la hora de hacer uso de la aplicación.

2. Conceptos Básicos

2.1 ¿Qué es una página web dinámica?

Las páginas webs dinámicas [1] son aquellas cuya información que presentan se genera a partir de una acción o petición del usuario de la página. Contrariamente a las páginas webs estáticas, en las que su contenido se encuentra predeterminado, en las páginas webs dinámicas la información aparece inmediatamente después de una solicitud hecha por el usuario.

Una página web dinámica permite visualizar la información contenida en una base de datos, así como almacenar y hacer actualizaciones de cierta información a través de un formulario. Para la creación de este tipo de páginas, además de etiquetas HTML es necesaria la utilización de algún lenguaje de programación que se ejecute del lado del servidor, así como la existencia de una base de datos.

2.2 ¿Por qué una página dinámica?

Si se analiza desde un punto más genérico el proyecto, se puede deducir que su base principal es la interacción del usuario con el sistema. Cada tipo de usuario va realizar distintas acciones, pero el dominador común de éstas, son las peticiones al servidor. Es por ello que las interacciones con el servidor y la posibilidad de utilizar un lenguaje de programación en el servidor, sean las principales razones por las que se ha optado por el modelo de una página web dinámica.

Además, se ha elegido una página web dinámica por las siguientes razones:

- Permite bastantes funcionalidades tales como bases de datos, contenido dinámico…
- El usuario puede alterar el diseño, contenido o presentación de la página a su gusto.
- Permite una gran cantidad de posibilidades en su diseño y desarrollo.
- Se puede actualizar de forma sencilla.
- Cuenta con soluciones prediseñadas de libre disposición.
- En su realización se utilizan diversos lenguajes y técnicas de programación.
- Existe una comunidad de programadores que ofrecen apoyo.

2.3 ¿Qué es SGBD?

Los Sistemas de Gestión de Base de Datos [2] son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

2.3.1 Ventajas de las bases de datos

- **Consistencia de datos:** Eliminando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

- **Mejora en la integridad de datos**: La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

- **Mejora en la productividad**: El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

- **Mejora en la seguridad**: La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

- **Compartición de datos**: En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece al

proyecto y puede ser compartida por todos los usuarios que estén autorizados.

- **Mejora en la accesibilidad a los datos**: Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

- **Control sobre la redundancia de datos**: Los sistema de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos. En los sistemas de bases de datos todos estos ficheros están integrados, por lo que se almacenan varias copias de los mismos datos. Sin embargo, es una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

- **Mantenimiento de estándares**: Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

- **Mejora en el mantenimiento**: En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de la aplicación que los manejan. Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados. Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

- **Aumento de la concurrencia**: En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero (como es nuestro caso), es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

3. Tecnologías Utilizadas

Al inicio del planteamiento del desarrollo se eligieron detalladamente las tecnologías que se iban a utilizar, con el fin de optimizar el rendimiento del proyecto. Para lograr la optimización del proyecto se buscaron únicamente tecnologías que se complementasen unas a otras, utilizándolas en un campo en concreto.

3.1 ¿Por qué HTML?

HTML [3] hace referencia al lenguaje de mercado para la elaboración de páginas web, que define una estructura básica y un código para la definición de contenido de una web, como texto, imágenes, etc.

Uno de los lenguajes fundamentales para nuestro proyecto ha sido HTML, debido a que ha sido el encargado de proporcionarnos la posibilidad de estructurar nuestra web en distintas capas, donde residirá su contenido. Debido al ser una página dinámica el uso de HTML solo ha sido frecuente en la elaboración de las vistas de nuestro framework, pero no por ello deja de ser un lenguaje fundamental debido a que sin él no hubiéramos podido realizar este proyecto.

3.2 ¿Por qué CSS, Bootstrap 3 y Media Queries?

Las siglas CSS [4] significan Cascading Style Sheets, que traducido al español es Hojas de Estilo en Cascada. CSS es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre el estilo y formato de las páginas web.

Bootstrap [5] es un framework de Twitter que permite crear interfaces web con CSS y Javascript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa. Debido a que uno de los objetivos que se marcaron fue la utilización del proyecto en distintos positivos, nos hemos decantado por Bootstrap el cuál automáticamente se adapta al tamaño de pantalla de todos los dispositivos.

En el desarrollo web, Media Queries [6] es un módulo CSS3 que permite adaptar la representación del contenido a características del dispositivo como la resolución de pantalla. La frecuencia de uso de Media Query en nuestro proyecto ha sido excepcional, debido a que se ha utilizado únicamente para

solventar algunas lagunas de adaptación de pantalla que tiene Bootstrap. El uso de esté modulo es muy sencillo, debido a que en nuestra hojas de estilo podemos indicar la resolución de pantalla con la que queremos trabajar y aplicar el estilo deseado con CSS.

3.3 ¿Por qué Javascript y JQuery?

Javascript [7] es un lenguaje de programación interpretado que permite incluir macros en páginas Web. Estas macros se ejecutan en el ordenador del visitante y no en el servidor. Debido a esta característica se ha utilizado Javascript en el proyecto, para realizar operaciones en la creación de formularios y para comprobar que los datos que introduce el usuario en el formulario antes de enviarlos sean correctos.

JQuery [8] es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML y manejar eventos, entre otros. Se ha utilizado JQuery en la creación de formularios por parte del usuario, para la interacción de crear un nuevo campo. La necesidad de añadir un nuevo campo al formulario de forma animada (intercalando en el código HTML) y poder almacenar en el lado cliente el formulario hasta su finalización, ha sido el principal motivo de hacer uso de está biblioteca.

3.4 ¿Por qué MySQL?

MySQL [9] es un sistema administrativo relacional de bases de datos. Este tipo de bases de datos pueden ejecutar desde acciones tan básicas como insertar registros o hasta realizar consultas tan complejas como nuestra aplicación necesite. MySQL es un servidor multi-usuarios rápido y robusto de ejecución de instrucciones en paralelo, es decir, que múltiples usuarios distribuidos a lo largo de una red local o internet podrán ejecutar distintas tareas sobre las base de datos de un servidor, justo lo necesario para un sistema de gestión de material.

Otra ventaja que ofrece MySQL frente a distintos sistemas de gestión de bases de datos es que es de código libre, por lo que podemos hacer uso de él sin ningún tipo de coste, garantizando seguridad y eficacia a nuestro proyecto.

3.5 ¿Por qué PHP?

PHP [10] es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. PHP fue uno de los primeros lenguajes de programación del lado del

servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos.

Se considera a PHP como uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hoy en día. Debido a la seguridad que garantiza operar sobre el lado del servidor y el alto rendimiento que ofrece, se decidió desde un primer momento utilizar este lenguaje para realizar el núcleo interno de nuestro proyecto.

Las principales razones para utilizar PHP [11] en nuestro proyecto son:

- Se basa en ser un lenguaje multiplataforma.
- Ofrece una capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la gran cantidad de módulos que posee.
- Permite las técnicas de programación orientada a objetos.
- Gran facilidad de aprendizaje.
- Se puede incrustar código PHP con etiquetas HTML

3.6 ¿Por qué Yii Framework?

Yii [12] es un framework PHP basado en componentes de alto rendimiento para desarrollar aplicaciones web de gran escala, destacándose de los distintos frameworks de PHP por su eficacia, gran cantidad de características y baja curva de aprendizaje. Una de las principales razones por las que se eligió utilizar este framework para el proyecto es su estructura modelo-vista-controlador, que proporciona una clara organización garantizando un alto rendimiento y una gran escalabilidad.

Esta estructura tiene como objetivo separar la lógica de negocio de las consideraciones de la interfaz de usuario a fin de que los desarrolladores puedan modificar cada parte más fácilmente sin afectar a otra. El modelo representa la información (los datos) y las reglas de negocio; la vista contiene elementos de la interfaz de usuario como textos, formularios de entrada; y el controlador administra la comunicación entre la vista y el modelo.

La ilustración 1 muestra la estructura estática de una aplicación en Yii:

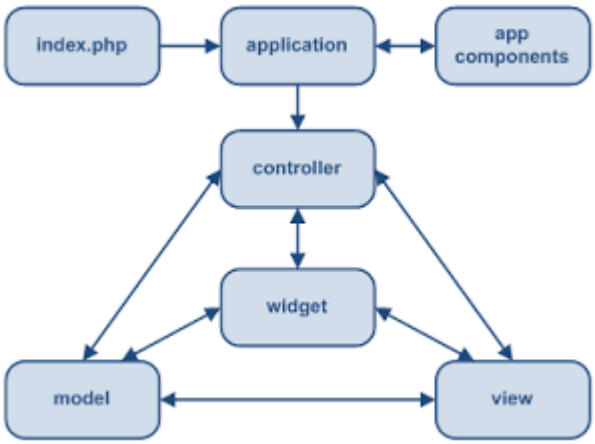

Ilustración 1: Estructura estática aplicación Yii

En el diagrama se observa como el nexo de unión que garantiza una comunicación entre el modelo y la vista es el controlador. Pero, ¿Cómo funciona Yii frente a una petición de un usuario? A continuación, en la ilustración 2 se muestra el flujo de tareas en una aplicación Yii cuando se trata de un pedido de usuario:

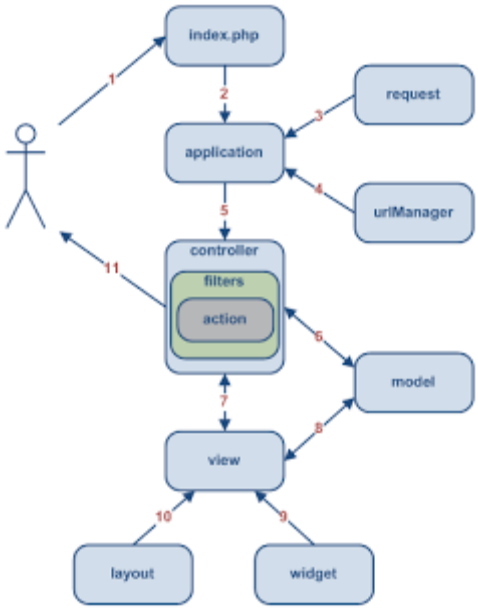

Ilustración 2: Interacción usuario con Yii

Este framework también nos facilita la unión de nuestra aplicación con la base de datos, proporcionando una seguridad a la hora realizar consultas SQL o insertar/eliminar elementos, garantizando la defensa sobre una posible vulneración en el sistema de reservas del material.

Las principales razones para utilizar Yii Framework en nuestro proyecto son:

- Patrón de diseño Modelo Vista Controlador, separando la interacción con la base de datos, el diseño y las acciones a realizar.
- Integración con JQuery. Como se indicó en el apartado 3.3 nuestro proyecto precisa de JQuery, es por ello que es fundamental elegir un framework que lo integre.
- Entradas de formulario y validación.
- Widgets de Ajax, como por ejemplo autocompletado de campos de texto.
- Personalización de aspectos y temas. Es fundamental para una página dinámica poder hacer un cambio de diseño modificando una simple línea de código, es por ello que Yii nos ofrece la posibilidad de crear distintos temas y a continuación elegir uno.
- El manejo de errores y logging. Los errores son manejados y personalizados. Además estos mensajes de error pueden ser categorizados y filtrados, por lo que se puede personalizar totalmente cada mensaje de error dependiendo de su causa.
- Las medidas de seguridad que ofrece Yii incluyen la prevención de cross-site scripting (XSS) y la prevención cross-site request forgery (CSRF), que previene la manipulación de cookies, etc.

Por último, destacar que Yii ofrece una herramienta llamada Gii, que será explicada a continuación.

3.6.1 ¿Por qué Gii?

Una vez creada la base de datos con sus correspondientes relaciones e indexada con Yii, se nos permite el uso de la extensión Gii. Este es el encargado de autogenerar el código proporcional a la base de datos y sus relaciones. También es capaz de generar el código de las acciones: crear, leer, actualizar y eliminar, comúnmente llamadas CRUD. Esto implica que automáticamente se generarán los modelos, las vistas y los controladores en base a nuestro proyecto, pudiendo luego operar sobre ellos. El ahorro de tiempo que nos proporciona esta herramienta a la hora de elaborar el proyecto ha sido la principal razón por la que hemos optado por ella.

3.7 ¿Por qué XAMPP?

XAMPP [13] es un servidor independiente de plataforma de software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. Como habíamos justificado en el punto 3.4, hemos optado por el uso de MySQL y al necesitar un servidor que pueda alojar páginas web dinámicas, encontramos esta herramienta capaz de

satisfacer nuestras necesidades y es por ello que ha sido escogido como servidor para la elaboración de nuestro proyecto.

Además incluye un módulo llamado phpMyAdmin, que nos proporciona un panel de administración para nuestra base de datos, muy intuitivo. Este panel de administración ha sido de gran ayuda a la hora de la creación o actualización de la base de datos, por lo que se ha hecho un gran uso de él. Por último indicar que la comodidad de tener todas las herramientas integradas en una aplicación ha tenido bastante peso a la hora de elegir el tipo de servidor.

3.8 ¿Por qué NetBeans y GitHub?

NetBeans [14] es un entorno de desarrollo integrado (IDE), modular, de base estándar, escrito en el lenguaje de programación Java. El proyecto NetBeans es un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación.

GitHub [15] es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se trata de un espacio virtual en internet para alojar los proyectos realizados o que están en proceso de elaboración, pudiendo guardar distintas versiones del proyecto o compartir el proyecto entero.

Debido a la importancia de hacer un backup diario de nuestro proyecto y la necesidad de encontrar un editor de texto que solventase los problemas de la visualización estructural que suelen tener los demás editores para grandes proyectos, se decidió utilizar NetBeans. Este entorno de desarrollo ofrece un plugin que se comunica con GitHub, con lo que cuando guardamos nuestro proyecto en el editor, automáticamente se realiza una copia de seguridad en GitHub, garantizando que no se pierda el código generado en nuestro proyecto.

4. Descripción del Escenario

Para establecer el diseño de la base de datos se debe estudiar todos los casos que el escenario y los objetivos nos plantean. Para ello, se debe estructurar de manera que la búsqueda de elementos en ella sea óptima y que todos ellos estén relacionados con un nexo común. En los siguientes párrafos se describirá el diseño de la base de datos relacionándolo con el funcionamiento que se dará al proyecto en realidad.

En primer lugar recordar que el objetivo principal del proyecto es crear un sistema de reserva para la petición de material. Uno de los objetivos que se plantearon fue el establecimiento de un sistema de filtros con su propio formulario, es por ello que primero se debe registrar en la aplicación las categorías a las que pertenecerá cada objeto. Cada categoría debe tener un identificador único y debido a que nos gustaría tener un sistema de herencia en el sistema de filtros, añadiremos un identificador para el padre de cada categoría en el caso que tuviese. También será necesario almacenar los atributos que generaremos al personalizar el formulario de cada categoría, por lo que se creará una nueva tabla donde albergará el identificador de la categoría, el nombre de cada atributo y su correspondiente tipo de contenedor para cada atributo del formulario. De esta manera, cada categoría irá relacionada con sus atributos y cada una contendrá su propio formulario personalizado.

Una vez creada la categoría correspondiente al objeto que deseamos registrar, este debe tener una serie de parámetros para garantizar que el sistema funcione. Para ello, los elementos principales son: un identificador único para cada objeto, el nombre del objeto, la categoría que pertenece el objeto, si está disponible el objeto y por último un parámetro que indique si el objeto pertenece a algún lote.

Como se planteó en los objetivos principales del proyecto, los objetos se deben poder organizar en una especie de “lotes” o “packs”, que se establecerán como unidad única en el sistema de reserva. Para ello cada vez que se registre un nuevo lote, los objetos que se añadan desaparecerán del sistema de reserva de objetos para pasar al del lote correspondiente. Principalmente los campos que debe contener los lotes son: un identificador único de cada lote relacionado con el identificador de lote que tiene cada objeto, el nombre del lote, si el lote está disponible, y una campo para una breve descripción del lote.

Una vez establecido las categorías, los objetos y los lotes en la base de datos, necesitamos crear los usuarios. Se acordó con el tutor, que nuestra aplicación debería tener un sistema de roles, por lo que gracias a una herramienta de Yii, podemos crear nuestros propios roles o acciones y almacenarlos en nuestra base de datos. Por otra parte, la tabla usuario debe contener su identificador único, el Nick único de cada usuario, la contraseña cifrada en SHA, el nombre, los apellidos, el teléfono, si el usuario está activo, la fecha de sanción, si el usuario está sancionado y la fecha de alta y baja del usuario.

Por último falta establecer el sistema de reserva. Para ello se debe llevar un riguroso control sobre las reservas realizadas, por eso se idearon varias fórmulas matemáticas, en las cuales necesitamos almacenar los siguientes datos:

- Un identificador único para cada reserva. Es muy importante que cada reserva tenga un único identificador que no pueda repetirse, debido a que uno de los objetivos del proyecto es almacenar un historial de las reservas realizadas.
- El identificador del usuario que ha realizado la reserva. De esta forma se podrá saber las reservas que ha realizado un usuario.
- El identificador del lote correspondiente en que se ha realizado la reserva. De esta forma, si en la reserva tenemos el identificador del lote, podemos consultar la reserva del lote en cuestión sin tener que mirar todos los lotes cada vez que se realice una reserva.
- El identificador el objeto, al igual que sucede con el identificador del lote, es conveniente almacenar la información del objeto a reservar.
- Un campo para orden. Debido a que cada objeto o lote puede reservarse aunque el objeto en sí lo esté disfrutando otro usuario, se ha ideado un sistema de cola, donde aparecerá en cada reserva el orden de prioridad y se irá modificando a medida que vayan entregando el material.
- La fecha de la reserva. Necesitamos tener un riguroso control de cuando el usuario ha realizado la reserva.
- El máximo de días que el usuario puede tener el objeto, antes de devolverlo.
- El máximo de días que un usuario puede ir a recoger el objeto.
- La fecha de recogida indica cuando el usuario ha ido a recoger el objeto que anteriormente había reservado.
- La fecha de recogida máxima. Es la fecha máxima en la cual el usuario puede ir a recoger el objeto. En el caso de que se supere esta fecha, se cancelará la reserva.
- La fecha de entrega que el usuario ha ido a devolver el objeto que había reservado y recogido.
- La fecha tope en la que el usuario puede ir a devolver el objeto.
- Los días de sanción equivalentes si el usuario no cumple las normas establecidas.

5. Diseño e Implementación

5.1 Esquemas de diseño

Para el diseño de nuestra aplicación y para establecer un mapa conceptual de los elementos visuales que albergaría nuestro proyecto se utilizó la herramienta Balsamiq Mockups, en el que podíamos plasmar todas las ideas principales que habían surgido hasta el momento. En los siguientes apartados se mostrarán únicamente los diseños previos de las ventanas que son imprescindibles y sus resultados (para distintos dispositivos), ya que la parte de administración se ha omitido por mantener una estructura muy similar a la que por defecto genera Gii.

5.1.1 Esquema página de inicio de sesión

Debido a que solo los usuarios registrados previamente por el administrador de la aplicación pueden acceder al sistema de reservas, tuvimos que diseñar una página de inicio de sesión. A diferencias de otras páginas web se diseñó el inicio de sesión de tal forma que el usuario no tuviera que indicar el dominio al que pertenecía, de esta forma el atacante no sabe cuántos dominios existen, ni como se llaman.

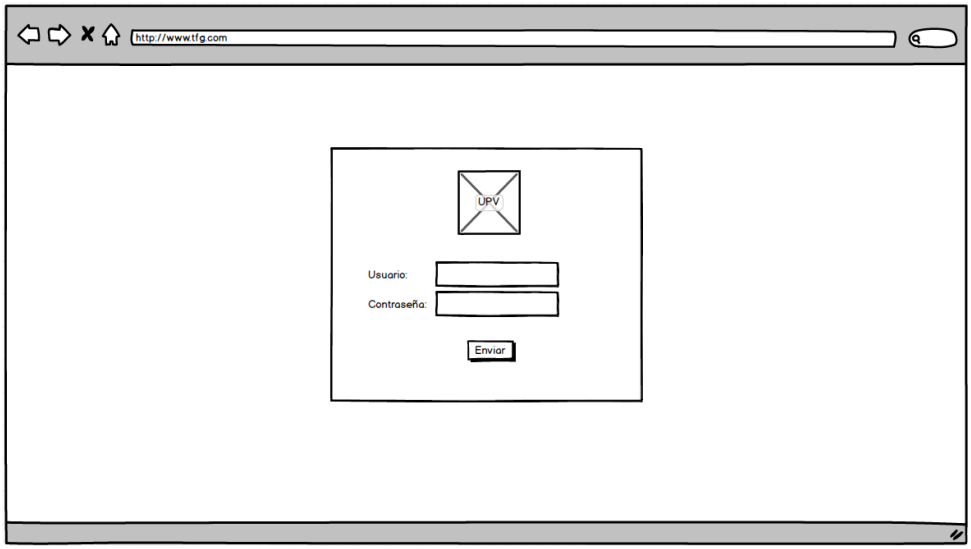

Ilustración 3: Diseño previo inicio

A medida que se fue implementando la página de inicio sesión, nos dimos cuenta de que la página web quedaba muy genérica y no plasmaba bien lo que era el proyecto, es por ello que se intercaló el mensaje “Servicio de préstamo de Material”.

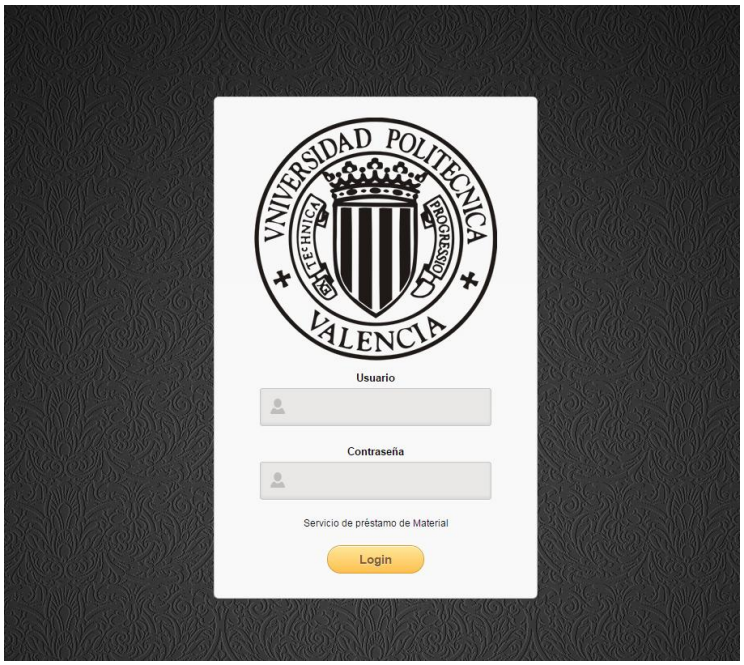

Ilustración 4: Resultado inicio en ordenador

Ilustración 5: Resultado inicio en móvil

5.1.2 Esquema página portada

Para garantizar la facilidad de uso de la aplicación se diseñó en su portada una breve descripción de las distintas secciones de la web. Para ello se decidió utilizar un slider que vaya indicando las secciones y de forma estática indicarlas en la parte inferior.

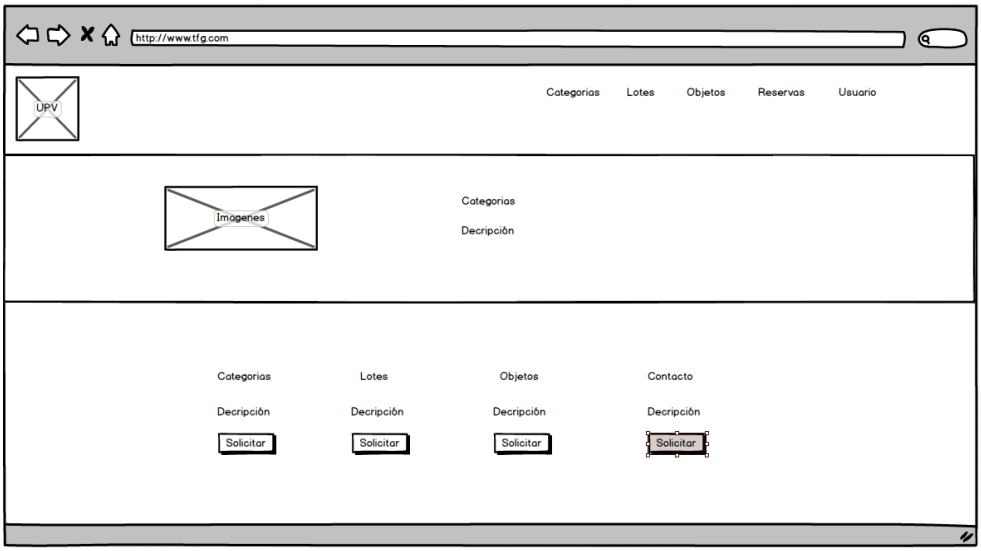

Ilustración 6: Diseño de portada

Debido a que se planteó como objetivo adicional una vez diseñado el proyecto, se añadió una sección nueva de contacto que no aparece en el diseño previo.

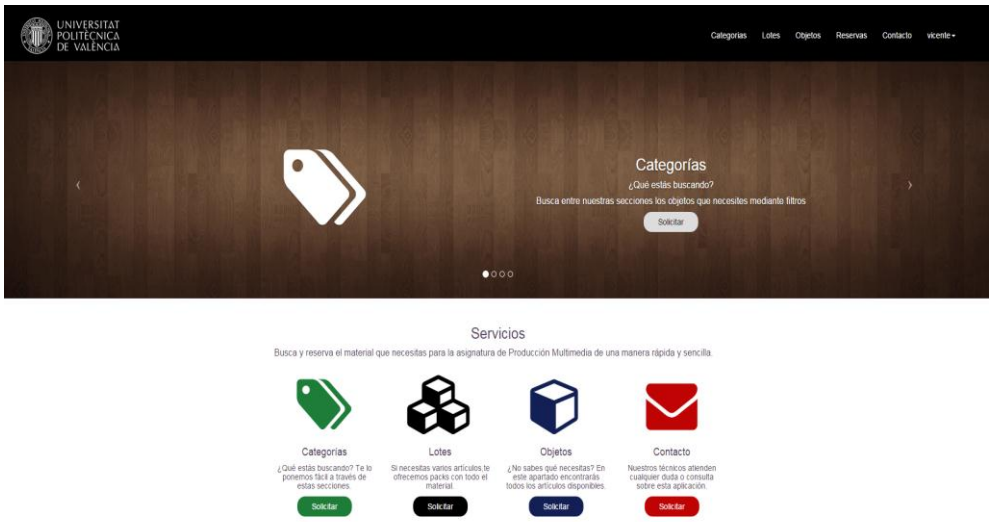

Ilustración 7: Resultado portada en ordenador

Uno de los principios universales de la usabilidad en versión móvil es centrar tanto las imágenes como el texto, garantizando una mayor facilidad de uso debido a las características de la pantalla del dispositivo.

Ilustración 8: Resultado portada en móvil

5.1.3 Esquema página categorías

Se propuso organizar los objetos en categorías como medida para reducir el tiempo de búsqueda por parte de un usuario mediante filtros. Para reducir más el tiempo de búsqueda del objeto, se ha creado un panel sencillo únicamente con la foto asociada a la categoría, evitando elementos que puedan dificultar su búsqueda. Este tipo de diseño también se ha aplicado a la página principal de lotes y objetos.

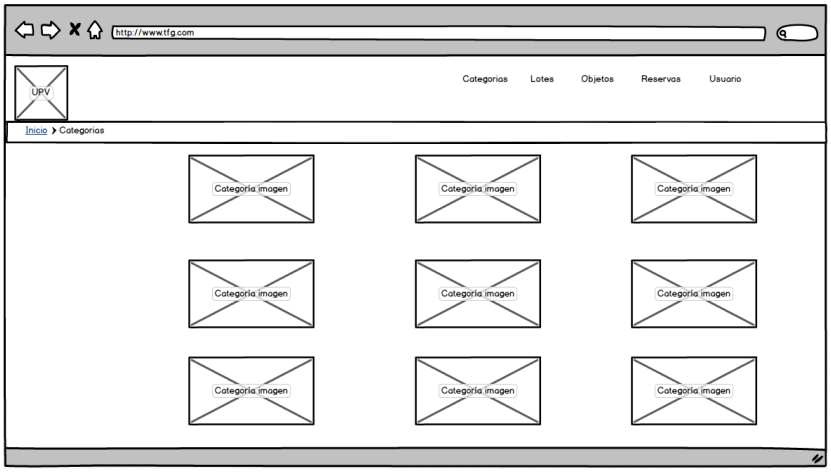

Ilustración 9: Diseño de categorías

A la hora de incorporar la imágenes a las categorías en la fase de desarrollo, se observó que algunas imágenes no definen correctamente a qué categoría corresponden los objetos que hay en ella, es por ello que se decidió incorporar el nombre de la categoría bajo de la imagen para no crear ningún tipo de duda al respecto. Debido a que la aplicación es totalmente dinámica el administrador puede ir incorporando categorías, por lo que se decidió que las categorías una vez creadas automáticamente caerán en cascada. Esta medida implica la anulación de un sistema de paginación en el sistema de selección de categorías, donde el usuario con el scroll del ratón podrá encontrar su categoría sin tener que buscar entre diversas páginas.

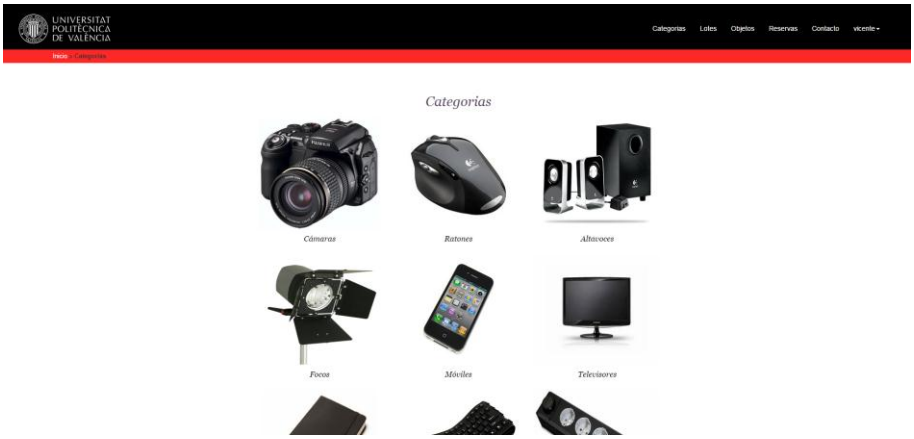

Ilustración 10: Resultado categorías en ordenador

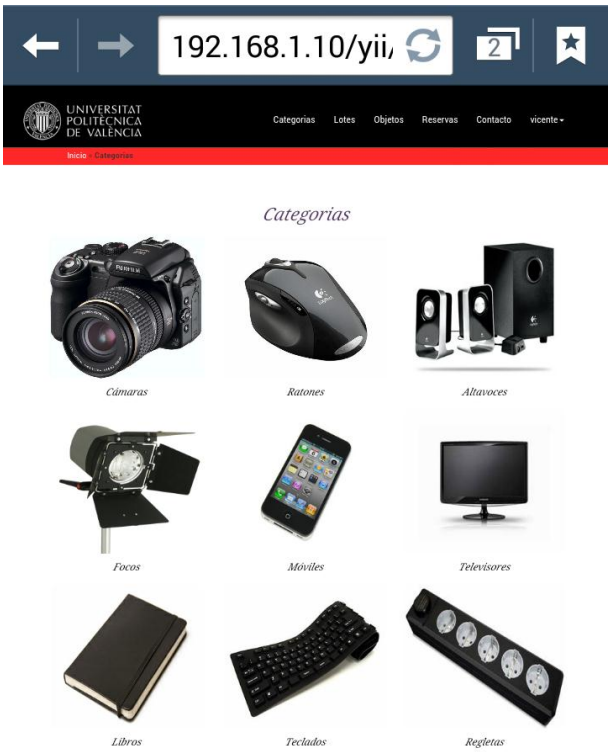

Ilustración 11: Resultado categorías en móvil

Una vez que se ha seleccionado la categoría, accedemos a los objetos que esta contiene. Como es de esperar, una categoría y más para un sistema de préstamo de material, puede albergar muchos objetos, es por ello que para lograr una visualización más conceptual se ha estructurado los objetos en cuatro filas, en vez de tres como las categorías. Para garantizar la usabilidad del sistema de filtros se ha añadido en la parte izquierda alineada con los objetos, un contenedor que contendrá las distintas categorías, de esta forma conseguimos que el usuario no tenga que salir de la propia categoría para poder acceder a otra distinta.

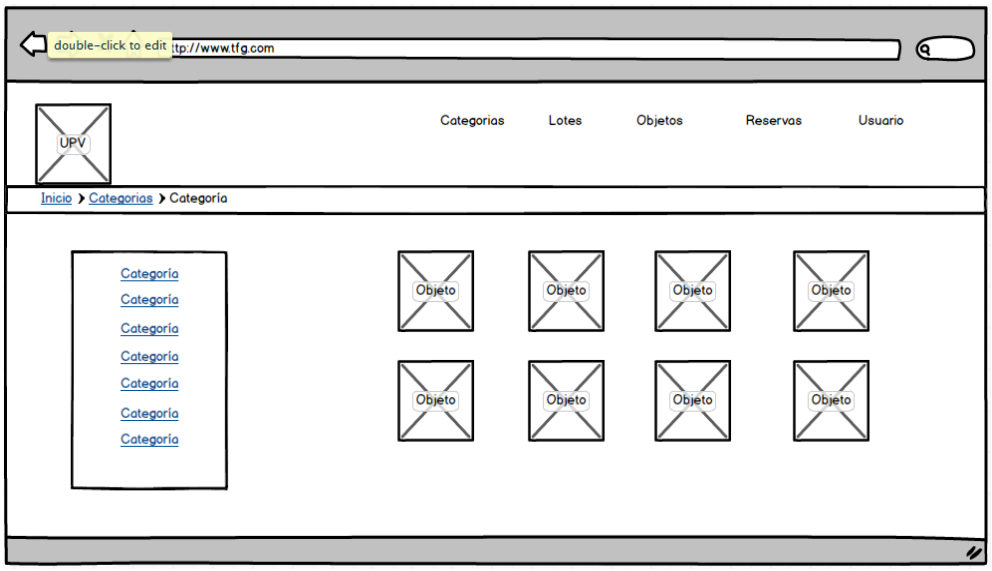

Ilustración 12: Diseño de categoría seleccionada

Al igual que sucedió con las categorías, lo objetos en un principio solo se iban a mostrar mediante una imagen y una vez dentro de él aparecería su nombre. Debido a que la imagen puede crear confusión o a que un objeto no se le haya añadido una imagen y salga la imagen por defecto, se decidió incorporar su nombre. También se ha desarrollado un algoritmo de ordenación de objetos, que consiste en dar prioridad a los objetos mediante el orden que aparecen en pantallas. Para ello se ha considerado tres factores.

1. Si el profesor/técnico ha indicado que esté disponible.
2. La cola de reserva del objeto.
3. El número de reservas que ha obtenido ese objeto (popularidad).

Cada factor está relacionado con el otro, por ejemplo si el objeto se considera popular por un histórico número de reservas y no está reservado pero si marcado como disponible por el profesor/técnico, tendrá la máxima prioridad lo que implica que será uno de los primeros objetos de la lista. En cambio si este objeto está reservado y tiene una cola de reserva alta, su prioridad se disminuye considerablemente, por lo que pasará a situarse en las últimas posiciones de la pantalla.

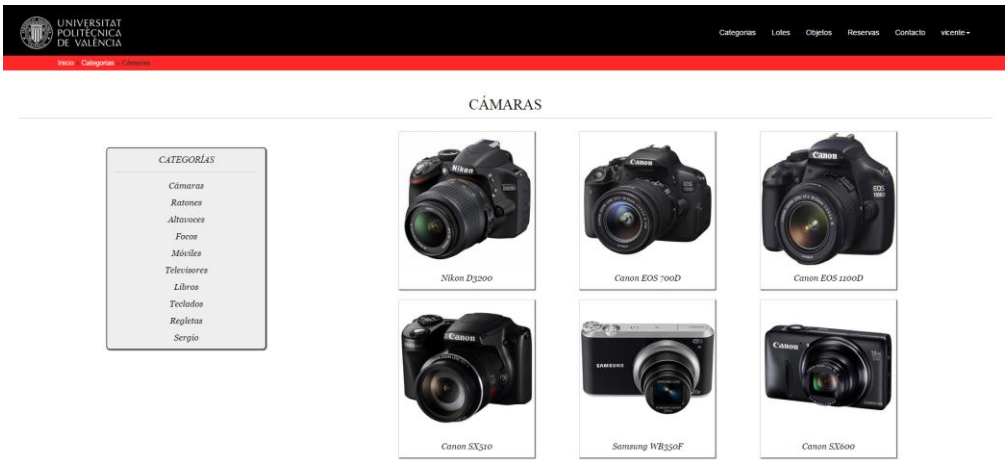

Ilustración 13: Resultado categoría seleccionada ordenador

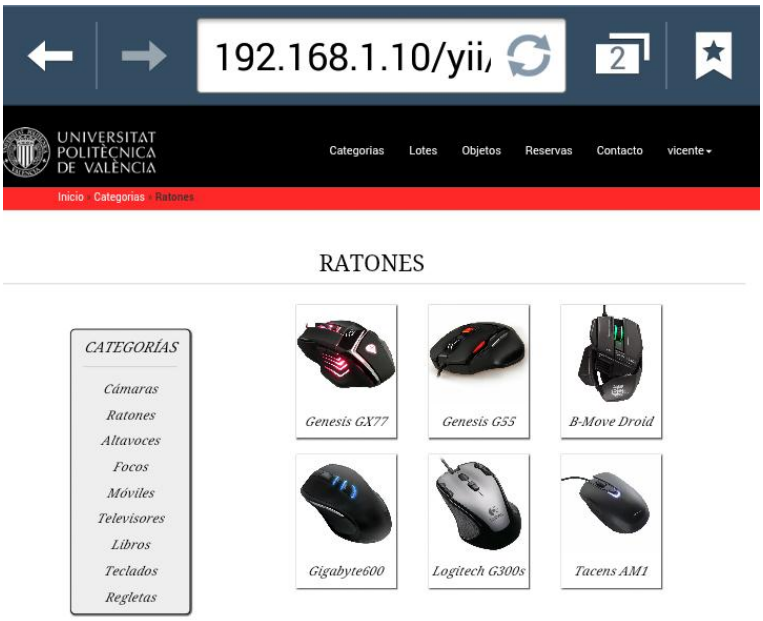

Ilustración 14: categoría seleccionada móvil

5.1.4 Esquema página objetos

Una vez localizado el objeto que se quiere reservar, deberemos mostrar sus correspondientes datos definidos previamente por el profesor/técnico que completó el formulario de la categoría asociada a ese objeto. Según los sistemas de gestión de material o negocio, es conveniente que la imagen del objeto deseado se encuentre en todo momento a la vista, es por ello que en la descripción también se ha decidido incorporar su foto sin reducir, situada en el lado superior izquierdo. Una vez incorporado la imagen del objeto, en la parte superior derecha se incorpora el formulario completado del objeto y posteriormente el botón reservar, que se seleccionará en el caso de que el usuario este de acuerdo hacer uso del objeto.

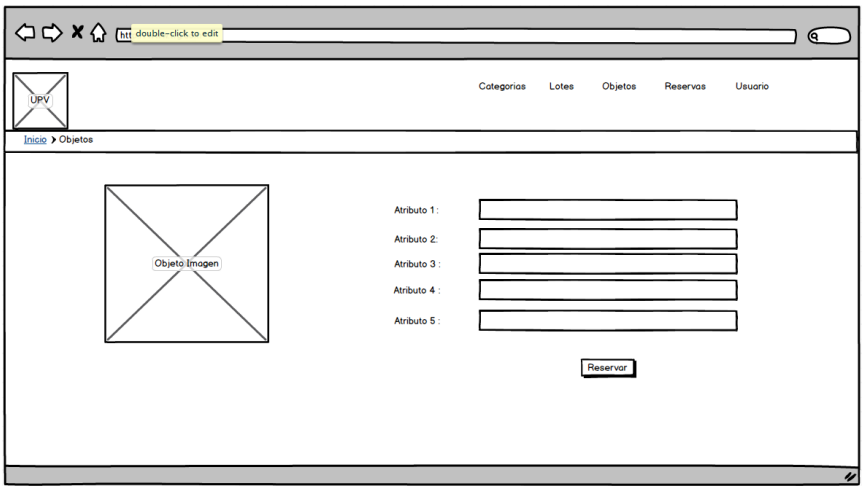

Ilustración 15: Diseño de objeto

Debido a la dinámica que nos ofrece Yii, en el objeto aparecerá automáticamente si está disponible marcado con un "Check" y si no está disponible marcado con una "X". También se ha incorporado por cada opción que se mostraba en el formulario un icono de desglose, para facilitar el inicio y final de cada campo.

Resultado versión ordenador:

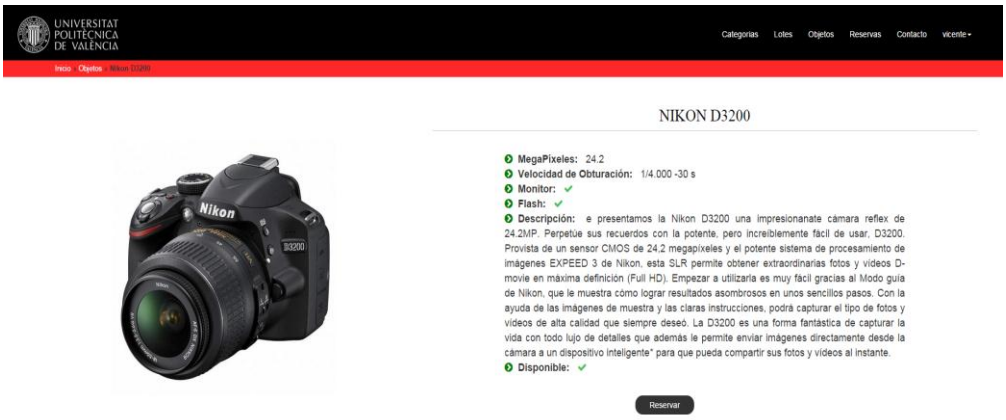

Ilustración 16: Resultado objeto en ordenador

En cuanto a la versión de móvil se observó que incorporar la imagen a la izquierda de la pantalla, el texto de su descripción a la derecha quedaba ilegible. Es por ello que se decidió por incorporar la imagen en la parte superior dejando un margen a los lados y a continuación presentar su descripción como texto alineado a la izquierda.

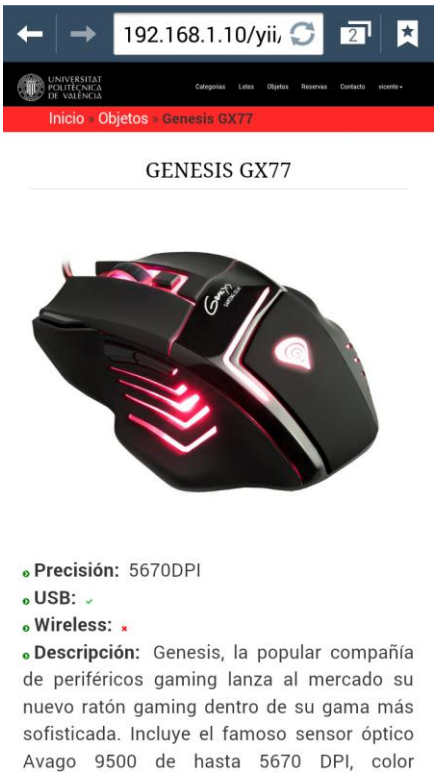

Ilustración 17: Resultado objeto en móvil

5.1.5 Esquema página reservar

Una vez el usuario haya realizado la reserva deberá ir a recoger el material al laboratorio donde se ejerce la asignatura. Desde la aplicación se debe ofrecer a cada usuario el historial de todas las reservas realizadas, así como cuando tiene que ir a recoger el objeto y cuando tiene que devolverlo. Para ello se habilita una sección reservas, donde mediante una tabla y un contador se indicará todas las reservas que se han realizado por el usuario indicando su estado (en vigor o en cola).

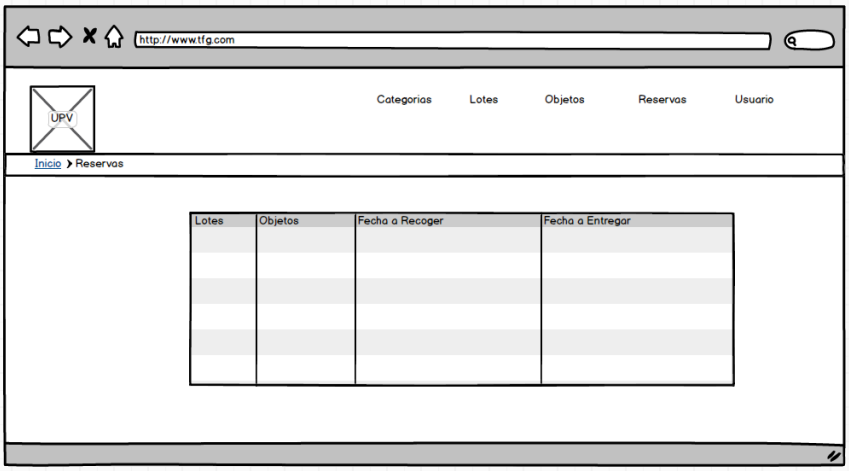

Ilustración 18: Diseño de tabla reservas

La estructura de la tabla permite indicar si está asignado un lote o un objeto, es por ello que como no se pueden reservar ambos a la misma vez se sustituyó el campo vacío por un guión, rellenando el hueco vacío para que no haya confusiones en el seguimiento de la tabla. Otra medida para facilitar la lectura de las reservas fue añadir un índice por cada reserva de la tabla, indicando a que fila pertenece.

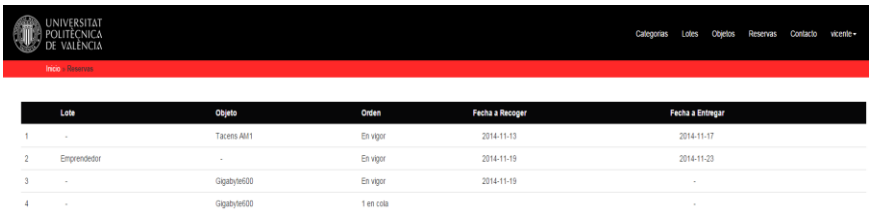

Ilustración 19: Resultado tabla reservas en ordenador

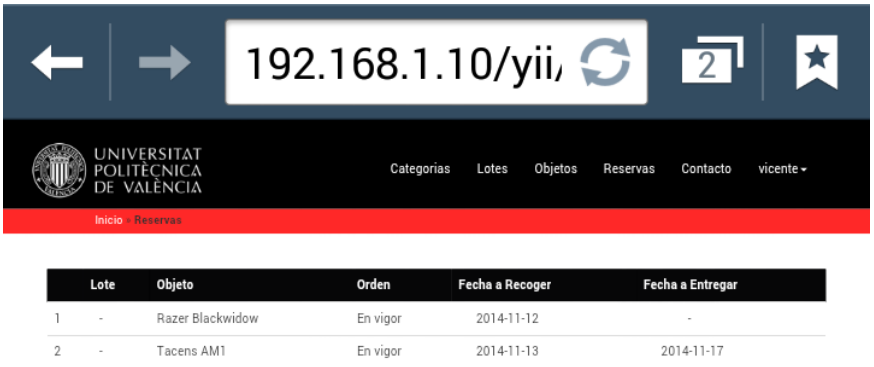

Ilustración 20: Resultado tabla reservas en móvil

5.1.6 Esquema página lote

Una vez registrados los objetos en la aplicación el profesor/técnico puede agruparlos en lotes. Un lote es una agrupación de objetos únicos, el cual solo se pueden reservar como unidad. La característica principal que debe tener un lote es mostrar que objetos lo contienen y una breve descripción sobre el lote. Como habíamos comentado anteriormente en el apartado de categorías, el filtro para seleccionar el lote y objeto es idéntico, es por ello que se muestra solamente el diseño del lote una vez seleccionado.

La idea de filtrar los lotes en categorías fue descartada desde un principio, debido a que no tiene sentido organizar los lotes por categorías ya que el lote en si puede ser formado por distintos objetos de diferentes categorías.

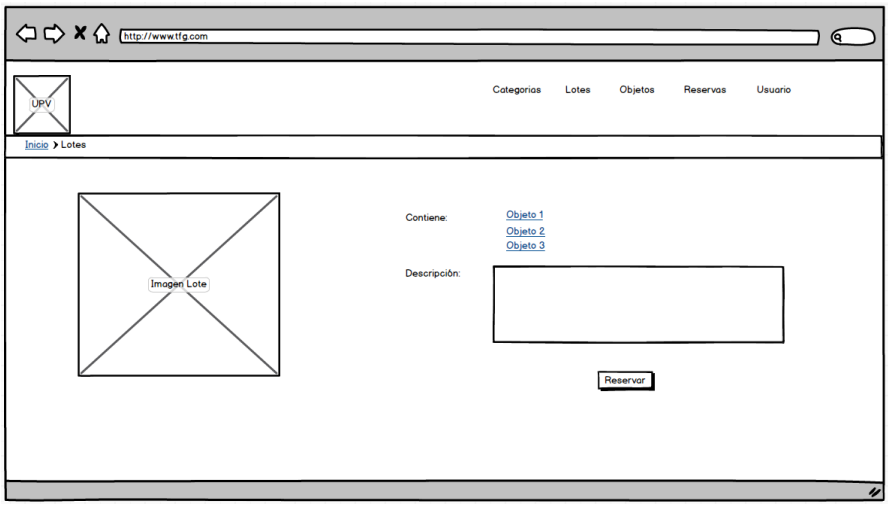

Ilustración 21: Diseño de lote

Como se puede observar en el diseño, los objetos que forman el lote son hipervínculos es decir, que si no recordamos con su nombre el objeto, podemos navegar hasta él y leer su descripción de manera detallada.

Ilustración 22: Resultado lote en ordenador

Al igual que sucede con los objetos, la imagen del lote es demasiado grande para la adaptación en versión móvil, es por ello que la imagen se sitúa en la parte superior y la descripción en la inferior, optimizando el desplazamiento del navegador.

Ilustración 23: Resultado lote en móvil

5.2 Diagrama y Diseño de la Base de Datos

Una base de datos es un "almacén" donde guardamos una colección o conjunto de informaciones (texto, imagen, sonido, video…) las cuales se encuentran relacionadas entre sí y pueden ser accesibles y consultadas en cualquier momento. Entre las principales características de los sistemas de bases de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación
- Acceso a través de lenguajes de programación estándar.

Existen distintos tipos de sistemas de gestión de bases de datos y como habíamos mencionado en el punto 3.4 de la memoria, nuestro SGBD es MySQL. Para poder administrar MySQL necesitamos la herramienta phpMyAdmin, la cual se encuentra dentro del módulo de XAMPP. Una vez aclarado varios conceptos, se explicará la creación de las tablas de nuestra base de datos representándolas con el diagrama UML.

5.2.1 Diagrama de casos de uso

Los diagramas de casos [16] de uso documentan el comportamiento de un sistema desde un punto de vista de usuario, es decir determinan los requisitos funcionales representando las funciones que un sistema puede ejecutar de forma sencilla. Un componente principal del diagrama de casos de uso es el papel del actor, que representa a un tipo de usuario en el sistema.

Como se definió desde un principio con el sistema de roles, en nuestra aplicación interactuarán tres tipos de actores. Cada uno de estos actores forma parte de un sistema de herencia, donde dependiendo del rol asignado podrá heredar las acciones de un actor y además podrá realizar sus propias acciones.

En primer lugar está el actor "Alumno", que representa a la mayor cantidad de usuarios registrados en la aplicación. Este actor es el que más restringido tiene sus acciones, debido a que el objetivo principal de su rol es poder realizar las reservas deseadas sin tener que realizar ninguna tarea de administración. En segundo lugar se encuentra el actor "Técnico", que hereda las acciones del actor "Alumno" y se le añaden permisos de administración. Es el encargado de realizar las tareas de administración necesarias sin la presencia del máximo responsable de la aplicación, en la cual puede realizar todas las operaciones del sistema excepto la administración de los usuarios. Por

último se define al actor “Profesor” como al máximo responsable de la aplicación que hereda todos los permisos de los demás actores y se incluyen las acciones de administración de los usuarios.

Cada actor representa una serie de objetivos a realizar mediante la interacción con el sistema y estas interacciones son una secuencia de pasos que se han construido mediante las siguientes relaciones [17].

- **Asociación:** Hay una asociación entre un actor y un caso de uso si el actor interactúa con el sistema para llevar a cabo el caso de uso. Por ejemplo nuestro actor “Alumno” tiene una asociación con la acción iniciar sesión.
- **Include:** Se puede incluir una relación entre dos casos de uso de tipo “include” si se desea especificar comportamiento común en uno o más caso de uso. Por ejemplo en el actor “Técnico” en el caso de uso de administrar aplicación, observamos que se definen diversos “includes” debido a que se observa un comportamiento común entre diversos casos de uso.
- **Extend:** Se puede incluir una relación entre dos casos de uso de tipo “Extend” si se desea especificar diferentes variantes del mismo caso de uso es decir, la relación “extend” implica que el comportamiento de un caso de uso es diferente dependiendo de ciertas circunstancias. Por ejemplo en el caso de uso de actualizar las reservas, existen diversos comportamientos diferentes como recoger o devolver, que depende del estado actual de la reserva.

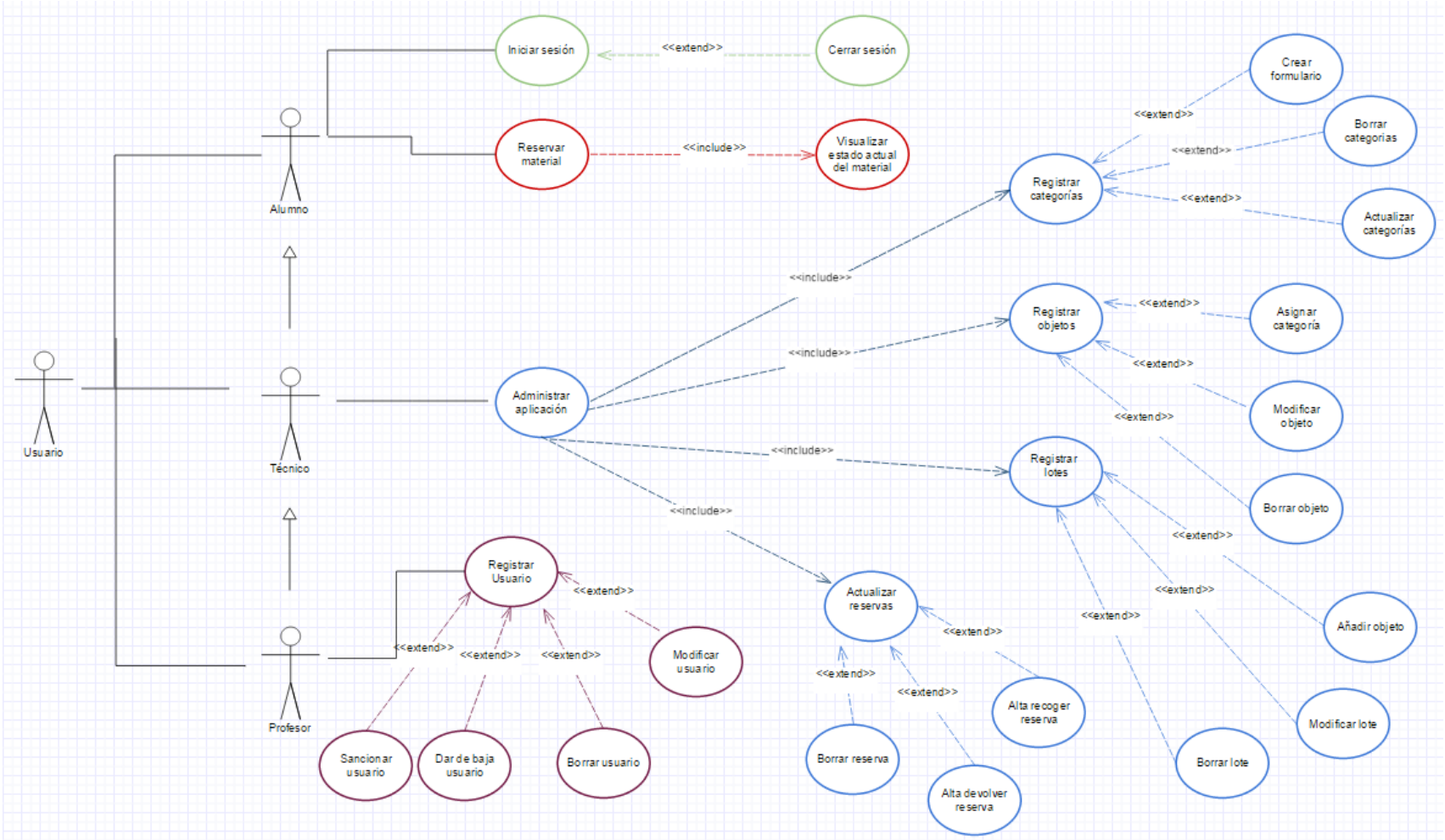

Ilustración 24: Diagrama casos de uso

5.2.2 Diagrama de clases

Los diagramas de clases modelan los conceptos del dominio de la aplicación, permitiendo visualizar las relaciones entre las clases que involucran el sistema componiéndose por: clases, relaciones y responsabilidades. Se ha utilizado la herramienta Gliffy editor para desarrollar el diagrama de clases debido a que es de código abierto donde de manera rápida y sencilla podemos crear el diagrama de clases de nuestra aplicación.

En la ilustración 25 se muestra las 10 clases que contiene nuestro proyecto con todos sus métodos, atributos y sus restricciones de integridad. En nuestro diagrama observamos distintos tipos de restricciones de seguridad como son:

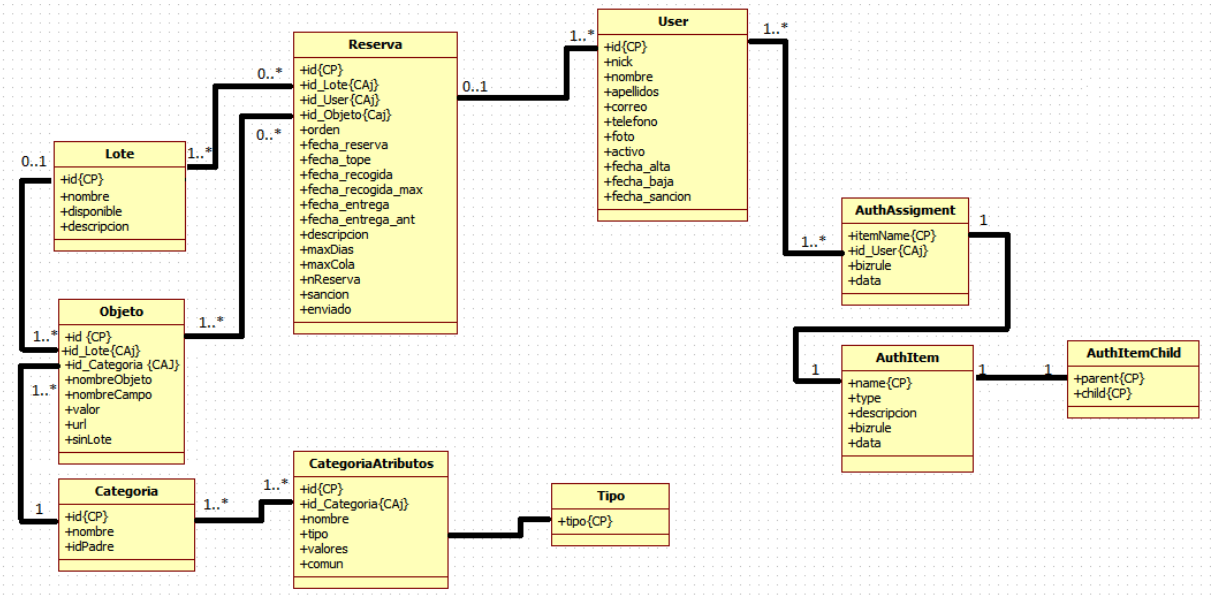

Ilustración 25: Diagrama de clases

5.2.3 Tabla Usuario

En primer lugar se diseñó la tabla de usuarios, pilar fundamental de nuestra aplicación, cuyos parámetros fueron consultados explícitamente con el tutor del proyecto para garantizar el almacenamiento y administración de los atributos de los usuarios que él creyese oportuno. Los siguientes atributos son:

- **ID**: identificador numérico único de cada usuario.
- **Nick**: nombre único de inicio de sesión del usuario.
- **Password**: contraseña cifrada en SHA-1 introducida por el usuario.
- **Nombre**: nombre del usuario.
- **Apellidos**: Apellidos de cada usuario, definido como cadena de caracteres.
- **Correo**: El correo electrónico facilitado por el usuario a la hora de crear el usuario.
- **Teléfono**: Sistema de localización opcional para la implementación de un futuro sistema de avisos vía mensajería telefónica o comunicación de sanción.
- **Foto**: Imagen facilitada por el usuario de forma voluntaria. En la base de datos se guardará únicamente el nombre y el formato de la imagen, los cuales utilizará la aplicación para buscar en su álbum e imprimir su foto.
- **Fecha_alta**: Fecha de creación del usuario.
- **Fecha_baja**: Fecha en que se da de baja a un usuario.
- **Activo**: Indica si el usuario está activado.
- **Fecha_sanción**: Fecha hasta la cuál un usuario está sancionado.
- **Sancionado**: Indica si el usuario está a fecha de hoy sancionado, sin tener que realizar una operación de comprobación por cada acción que realice el usuario.

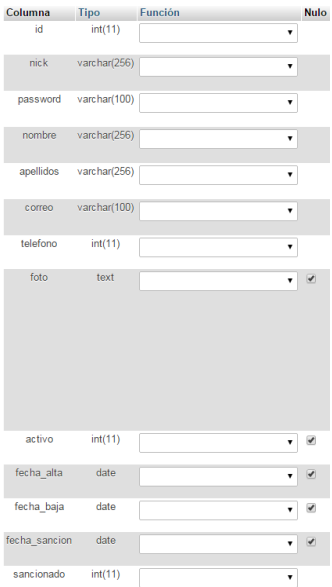

Ilustración 26: Tabla Usuario

Como se había mencionado a lo largo del proyecto, el framework Yii facilita un sistema de roles y sistema de herencias. Para ello, se ha de crear una serie de tablas para que este sistema pueda funcionar las cuáles se explicarán en los siguientes apartados.

5.2.3.1 Tabla AuthAssigment

Como indica su propio nombre, es la tabla en la cual se asigna el Nick único de un usuario a un rol o acción en concreto. Cada vez que el usuario intente realizar una operación, el sistema buscará el usuario en la siguiente tabla y comprobará que rol tiene asignado el usuario.

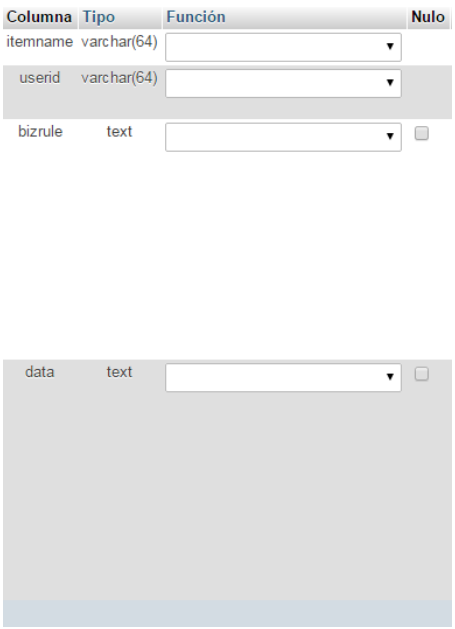

Ilustración 27: Tabla AuthAssigment

5.2.3.2 Tabla AuthItem

En la tabla AuthItem se definen los roles y las acciones que deseamos. Para distinguir si se quiere definir una acción o un rol se incluye el campo “Type”, que si almacena un “1” se indica que describe una acción y si almacena un “2” estaremos indicando la descripción de un rol. De esta manera podemos asignar a un usuario un rol y añadir acciones exclusivas a ese usuario sin la necesidad de crear un nuevo rol.

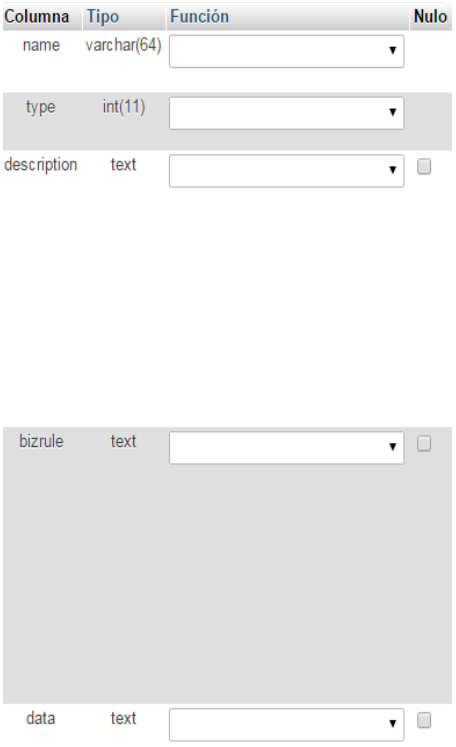

Ilustración 28: Tabla AuthItem

5.2.3.3 Tabla AuthItemChild

Si se quiere mantener un sistema de herencia necesitamos indicar al sistema el árbol genealógico que mantiene nuestra aplicación. Es por ello que por cada rol se le indicará su padre, pudiendo seguir una estructura en árbol.

Columna	Tipo	Nulo
parent	varchar(64)	No
child	varchar(64)	No

Ilustración 29: Tabla AuthItemChild

5.2.4 Tabla Categoría

En el planteamiento inicial del proyecto, la tabla categoría iba a albergar todos los atributos que la componen. Debido a que se planteó realizar un sistema de herencia dentro de las categorías, se decidió que hubiese dos tablas, uno para el sistema de herencia y otra para sus atributos. La tabla "Categoría" contiene:

- **ID**: identificador único de la categoría.
- **Nombre**: el nombre de la categoría en cuestión
- **ID_Padre**: el identificador de la categoría padre en el caso de que perteneciese a un sistema de herencia, por ejemplo el padre de la categoría "Sillas" puede ser la categoría "Muebles", formando un sistema de herencia.
- **Imagen**: El título de la imagen de la categoría para la visualización en su web.
- **Logo**: El título de una versión de la imagen reducida.

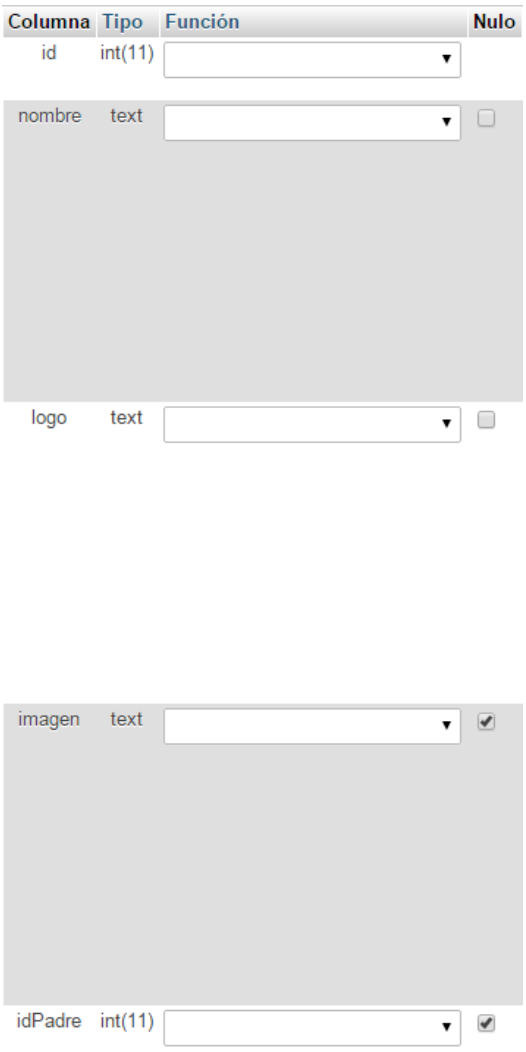

Ilustración 30: Tabla Categoría

5.2.4.1 *Tabla Categoría-Atributos*

Como habíamos comentado, se decidió albergar los atributos pertenecientes a las categorías en otra tabla. La siguiente tabla está compuesta por:

- **ID**: Cada atributo debe albergar un único número identificativo,
- **ID_Categoría**: El identificador de la categoría a la que pertenece este atributo.
- **Nombre**: El nombre del atributo, por ejemplo el atributo "Descripción".
- **Valores**: El valor que albergará el atributo.
- **Común**: Para conocer si un atributo ya ha sido creado para otra categoría.
- **Tipo**: El tipo de contenedor que albergará el atributo en el formulario. Por ejemplo en el atributo "Descripción" al precisar de bastantes caracteres el administrador almacenará el tipo "TextArea".

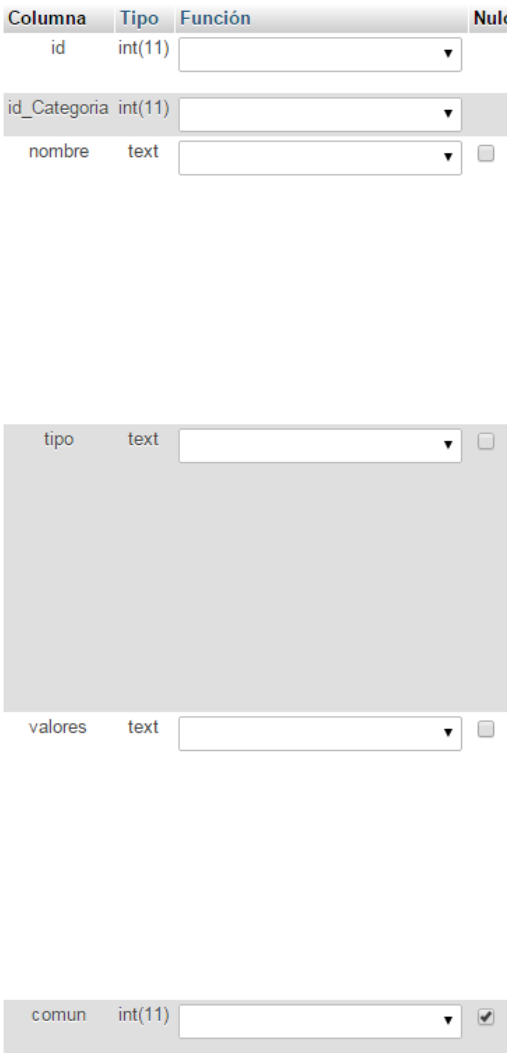

Ilustración 31: Tabla Categoría-Atributos

5.2.5 Tabla Objeto

El objeto que el administrador pretende registrar debe ser único pese a que puedan tener dos objetos iguales, distinguidos por un número de identificación. Además deberá contener a la categoría que pertenece dicho objeto y si pertenece a algún lote. Los atributos que definen al objeto son:

- **ID**: El identificador único de cada objeto representado por un elemento numérico.
- **ID_Categoría**: El identificador asociado a la categoría perteneciente.
- **ID_Lote**: Si el objeto pertenece a un lote, contendrá el identificador del mismo.
- **Nombre Objeto**: El nombre del objeto que figurará en su descripción.
- **Valor**: El valor de los atributos rellenados en el formulario del objeto.
- **URL**: Si el objeto contiene una guía vía web se podrá acceder almacenando su enlace.
- **Disponible**: Si el objeto está disponible para ser reservado.

- **Logo**: La imagen perteneciente al objeto.

5.2.6 Tabla Lote

Un lote puede ser compuesto por varios objetos es por ello que el identificador del objeto se ha establecido en la tabla objeto y no en la tabla lote para mejorar el sistema de búsqueda. Los atributos que se almacenan en la tabla lote son:

- **ID**: El identificador único de cada lote.
- **Nombre**: El nombre del lote.
- **Disponible**: Si el lote está disponible para realizar una reserva.
- **Descripción**: Debido a que un lote no pertenece a una categoría, si no a un conjunto de objetos, de añade el atributo descripción para que se pueda explicar la razón de la unión de los diversos objetos.
- **Logo**: La imagen del lote.

Ilustración 32: Tabla Lote

5.2.7 Tabla Reserva

Una vez establecida todas las demás tablas llegamos a la pieza central de nuestra aplicación, el sistema de reserva. Los atributos que alberga la siguiente tabla son:

- **ID**: Identificador único de cada reserva. Es muy importante destacar que este identificador debe ser único a pesar que se reserve el mismo material en sucesivas ocasiones en el mismo espacio de tiempo,
- **ID_Lote**: El identificador único del lote el cual se ha reservado. Este campo no es necesario que sea rellenado debido a que puede que se haya reservado solo un objeto.
- **ID_Objeto**: El identificado único del objeto reservado. Este campo puede considerarse como nulo si el campo ID_Lote contiene un identificador de lote, es decir, siempre habrá un solo valor para ambos campos.
- **ID_User**: El identificador del usuario que ha reservado el material. Es muy importante que el usuario figure en la reserva para poder llevar un claro control en el sistema.

- **Orden**: La cola de reserva para el material. Es decir, en el caso de que se reserve varias veces el mismo material en un limitado espacio de tiempo se establecerá un orden para hacer uso del material reservado.
- **Fecha Reserva**: Para llevar un control en la realización de cada reserva, se almacena en este atributo la fecha en la cual se produjo.
- **Fecha Tope**: Máxima fecha de devolución del material, en el caso de que hubiese un retraso se aplicaría la sanción fijada por el administrador.
- **Fecha Recogida**: Fecha en la cual el usuario ha ido a recoger el material.
- **Fecha Recogida Max**: Fecha máxima en la cual el usuario puede ir a recoger el objeto.
- **Fecha Entrega**: Fecha en la cual el usuario ha ido a devolver el material.
- **Fecha Entrega Ant**: Fecha en la cual el anterior usuario que reservó el material fue a entregarlo.
- **Descripción**: Si existe algún dato que puntualizar en la reserva se abre la posibilidad de anotar lo sucedido en una descripción de la reserva.
- **Max Días**: El máximo periodo en días que el usuario puede permanecer con el material.
- **Max Cola**: El máximo periodo en días que el usuario puede tardar en recoger el material.
- **Sanción**: El periodo de días de sanción que equivale un día de retraso. Si almacenamos la sanción por cada reserva podemos asignar distintos tipos de sanciones dependiendo del criterio que se desee.

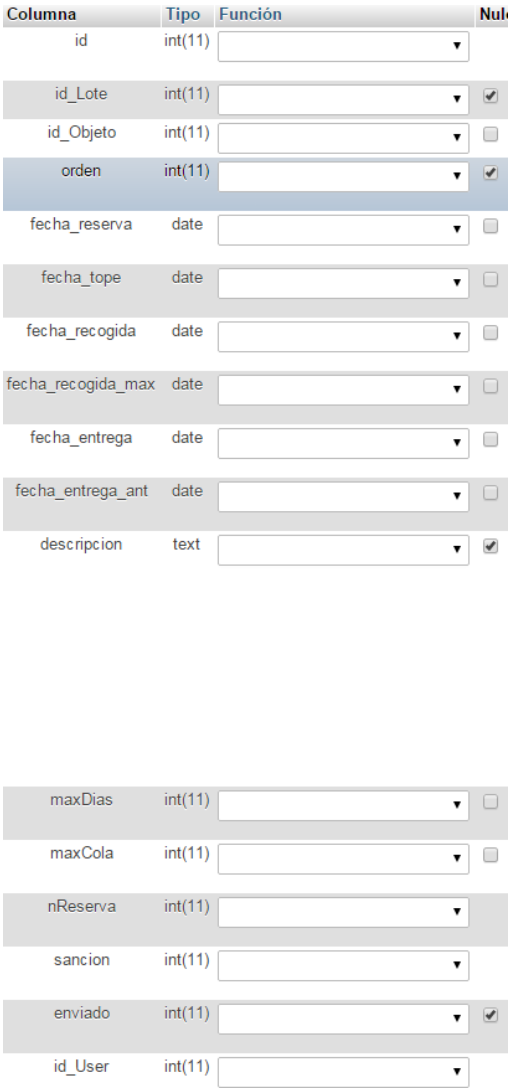

Ilustración 33: Tabla Reserva

6. Resultados y Ampliación

En el siguiente apartado comprobaremos que todos los objetivos que fueron asignados han sido cumplidos y se planteará opciones de mejora de la aplicación para un futuro.

6.1 Resultados

El primer objetivo consiste en ofrecer la posibilidad de crear usuarios con distintos roles y poder administrarlos. Cuando se crea un usuario, la aplicación permite asignar al usuario su correspondiente rol y una vez creado se puede actualizar mediante la tabla de administración.

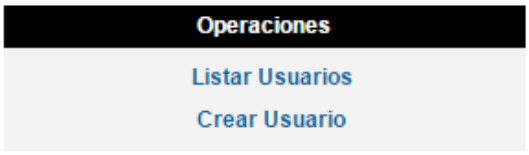

Ilustración 34: Opción crear usuario

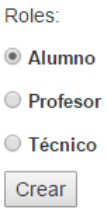

Ilustración 35: Selección de rol

Ilustración 36: Administración usuarios

El siguiente objetivo debe permitir registrar objetos en la aplicación y poder agrupar los objetos en lotes, es por ello que se muestran las opciones de crear objetos, crear lotes y la posibilidad de añadir objetos.

Ilustración 37: Crear objetos

Ilustración 38: Crear lotes

Ilustración 39: Añadir objetos

Ilustración 40: Selección añadir objeto

Para poder llevar un orden en la aplicación se definió el objetivo de ofrecer la opción de administrar dinámicamente los objetos registrados, es por ello que en la ilustración 41 se muestra la tabla donde se pueden listar, actualizar y borrar objetos del sistema.

Búsqueda

Nombre Objeto	Nombre Campo	Id Categoria	Id Lote	Disponible	Url
Canon EOS 1200D	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19	14	1	
Nikon D3200	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Canon EOS 700D	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Canon EOS 1100D	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Canon SX510	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Nikon D3300	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19	19	1	
Samsung NX300	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19	19	1	
Sony DSC-WX220	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19	19	1	
Samsung WB350F	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Canon SX600	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Samsung NX3000	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Canon SX270	MegaPixeles;Velocidad de Obturación;Monitor;Flash;Descripción;	19		1	
Genesis GX77	Precisión;USB;Wireless;Descripción;	20		1	
Genesis G55	Precisión;USB;Wireless;Descripción;	20		1	
Razer DeathAdder	Precisión;USB;Wireless;Descripción;	20	16	1	
B-Move Droid	Precisión;USB;Wireless;Descripción;	20		1	

Ilustración 41: Administración de objetos

Una vez puesta en funcionamiento nuestra aplicación pueden surgir preguntas o problemas técnicos por parte de los usuarios. Como consecuencia se estableció como objetivo principal desarrollar un formulario de contacto para poder establecer una conexión con el administrador, tal y como se muestra en la ilustración 42.

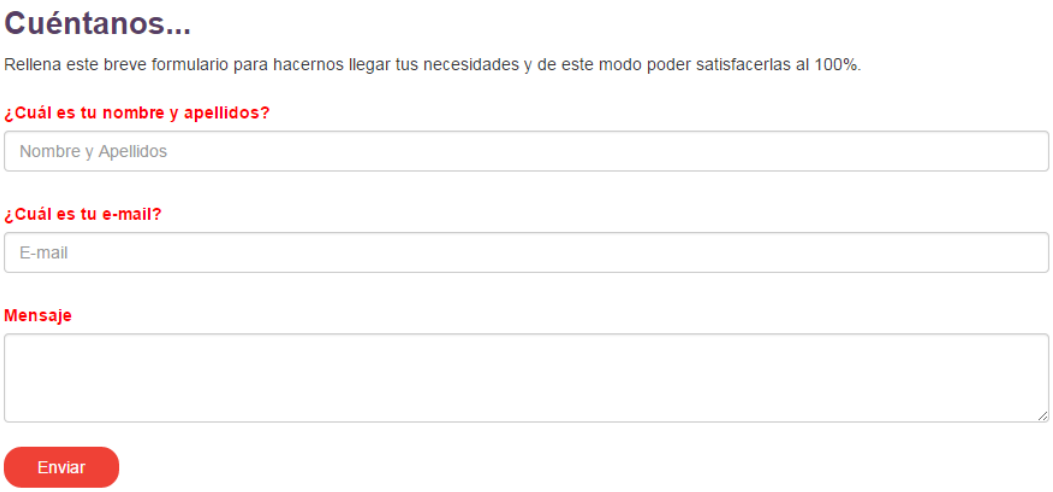

Ilustración 42: Formulario de contacto

El último objetivo principal que fue definido por el tutor fue establecer un método de sanción según la infracción cometida en el periodo de reserva. Para ello el sistema por defecto tiene una sanción predefinida de dos días de sanción por uno de retraso, aunque se da la opción de poder modificarlo. Para ello se comprueba la fecha actual del sistema y se resta la fecha tope en que pudo devolver la reserva. Una vez comprobado estos parámetros sabemos que usuarios del sistema deben sancionarse y porque reserva han recibido la sanción, por lo que se insertará mediante el método sancionar la nueva fecha de sanción en el usuario. De esta forma cada vez que el usuario inicie sesión le indicará que está sancionado y hasta que fecha. Debido ante la imposibilidad por motivos técnicos de utilizar el cron para actualizar todos los usuarios a una determinada hora del día, se estableció el método actualizar sancionados donde se actualizará la lista de sancionados cuando el usuario o el administrador inicien sesión.

```php
public function actualizarSancionados($id = NULL) {
    $usuarios = $this->getUsuariosPendientesSancion($id);

    foreach ($usuarios as $usuario) {
        $this->sancionar($usuario);
    }
}

public function sancionar($usuario) {

    $reservasPendientes = Reserva::model()->getReservasPendientesSancion($usuario->id);

    foreach ($reservasPendientes as $reserva) {

        $reserva->sancionar();
    }
}
```

Ilustración 43: Métodos sanción

Como podemos observar, estos métodos tienen la particularidad que podemos definir la cantidad de usuarios que necesitamos actualizar, es por ello que si inicia sesión un alumno se pasará por parámetro el identificador del alumno y solo se sancionará a él. En cambio sí inicia sesión un técnico o un profesor el parámetro que se pasará a la función será null, actualizando todos los usuarios sancionados. De esta forma se consigue solventar los problemas de actualización de la aplicación en cuanto a sanciones, repartiendo la carga del sistema entre los diversos usuarios y así garantizar un sistema eficiente.

Una vez se ha visto los objetivos principales marcados por el tutor, se han de comprobar los objetivos añadidos para una mejora del sistema. El primer objetivo de mejora que se propuso fue poder crear formularios de manera dinámica, para registrar cualquier tipo de objeto. Como se puede observar en la ilustración 44 se pueden ir añadiendo los campos del formulario previsto junto a sus contenedores y en el caso de que sea un RadioButton se podrá añadir valores predeterminados.

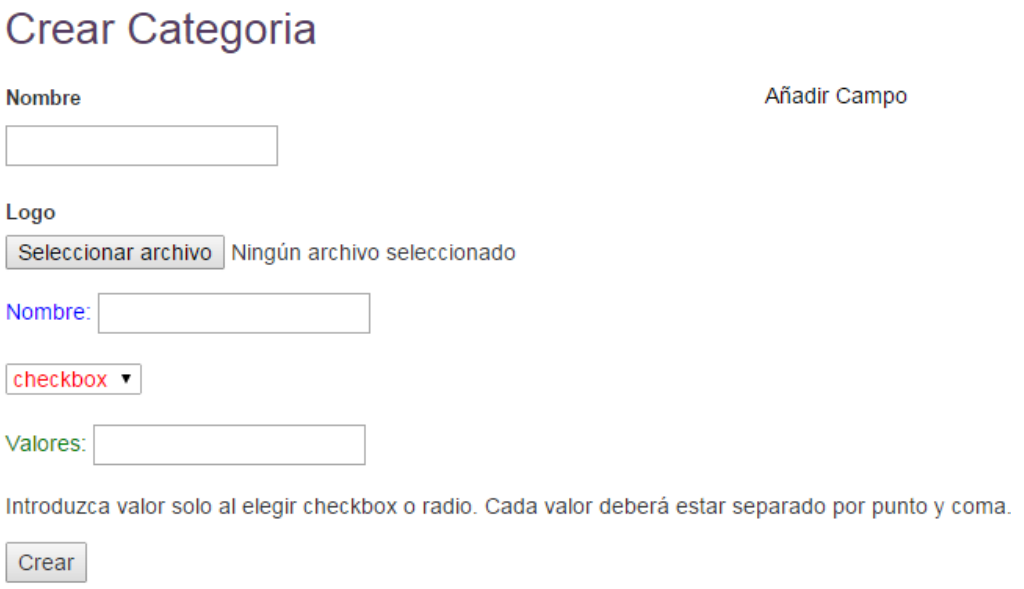

Ilustración 44: Creación formulario

Si se quiere añadir otro campo al formulario se deberá hacer click en el enlace "Añadir Campo" y automáticamente se añadirá como podemos ver en la ilustración 45.

Ilustración 45: Añadir campo

El siguiente objetivo de mejora consiste en desarrollar un sistema de búsqueda para cada elemento de la tabla de administración. Para ello encima de cada tabla de administración se encuentra el enlace "Búsqueda" que al ser seleccionado se despliega el sistema de búsqueda.

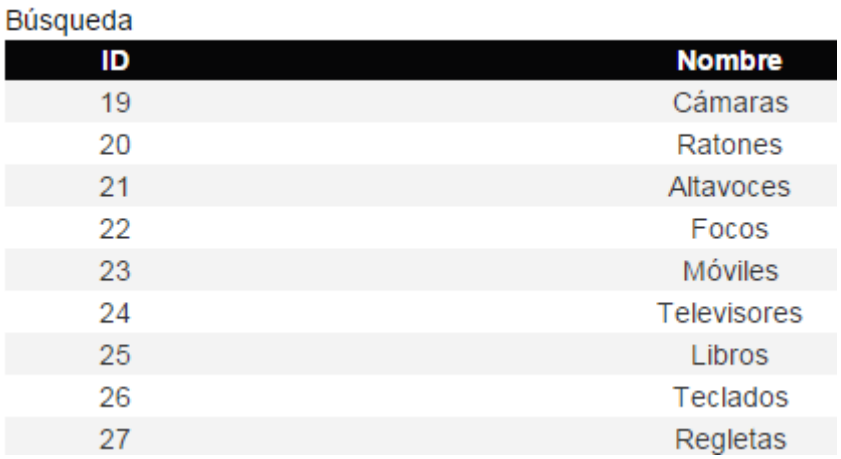

Ilustración 46: Búsqueda

Ilustración 47: Búsqueda y filtro

De esta forma podemos realizar filtros en la propia tabla, seleccionando el campo en concreto que queremos. También se ha establecido un sistema de orden, es decir que si queremos ordenar las "ID's" por orden numérico o la columna Nombre por orden alfabético podemos realizarlo solo haciendo click en el nombre de la columna.

Una vez se hayan creado las categorías oportunas se planteó como objetivo poder asignar los objetos mediante sus filtros, como se muestra en la ilustración 48.

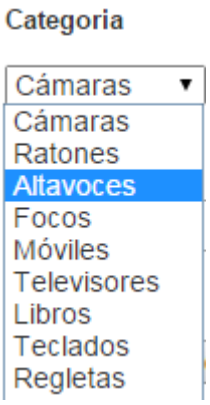

Ilustración 48: Asignación categoría

Por último se decidió hacer hincapié en la seguridad del sistema, por eso se marcó como objetivo utilizar algoritmos de encriptación. En nuestra aplicación se ha utilizado el algoritmo SHA-1, almacenando las contraseñas de forma segura en nuestra base de datos tal como muestra la ilustración 49.

nick	password
alumno	3b7f597562f4f92989516a1094c428a441604c52
bla	1ee99a7c4bbdc0c515aacbfb24552773069a331f
david	5e0c234f74aeb660a829b802b9da2e9d91ff53af
guille	ad6fbb1b57d0f84bd91d8d2eb87b8adcf5d27767
jose	135e00a18fd564749a52418eee258219ba3b143b
tecnico	103a377f64ce77bdea4ce120709e9a337097b3bb
vicente	bc8f2b3dbae6a4ae7dd4be8a9d81fa754e6ec34f

Ilustración 49: Base de datos SHA

Mediante la función hashPwd generamos el token para cifrar la contraseña introducida por el usuario a SHA y viceversa.

```
public static function hashPwd($usr, $pwd) {
    return sha1($usr . Yii::app()->params['tokenPWD'] . $pwd);
}
```

Ilustración 50: Función SHA

Esta función se debe utilizar cuando el usuario se autentifique, como se muestra en la ilustración 51.

```
public function authenticate()
{

        $usuario = User::model()->findByPk($this->username);

        User::model()->actualizarSancionados();
        User::model()->actualizarNoSancionar();
        User::model()->actualizarReservasCaducadas();

    if(!$usuario) {
        $this->errorCode=self::ERROR_USERNAME_INVALID;
    } else if(User::hashPwd($this->username, $this->password)!==$usuario->password) {


        Yii::trace("En la BD ? : " . $usuario->password);
        $this->errorCode=self::ERROR_PASSWORD_INVALID;
        Yii::trace(" HASHEO: " . User::hashPwd($this->username, $this->password));

    } else {
        $this->errorCode=self::ERROR_NONE;
    }
    return !$this->errorCode;

}
```

Ilustración 51: Función autentificación

6.2 Mejoras

Una vez realizado el proyecto y habiendo realizado todas las pruebas oportunas se estudian cuatro mejoras que en un futuro se podrían realizar.

La primera mejora consiste en el desarrollo de un sistema de avisos mediante telefonía y email. Esta mejora permitiría una vez realizada la reserva o esperado el periodo de cola, notificar con un mensaje vía email o vía teléfono al usuario.

Otra mejora más revolucionaria sería utilizar la API de Facebook para poder hacer reservas o para notificar que se ha registrado un nuevo producto en la aplicación. La mejora consistiría en recoger los datos de la cuenta de Facebook que se debería generar para la asignatura y transformar estos datos de JSON a texto plano para realizar las operaciones necesarias en nuestra base de datos.

Poliformat permite exportar los alumnos de una asignatura y generar un documento PDF. Una mejora que se podría introducir en nuestra aplicación sería poder importar el PDF generado en Poliformat, creando los usuarios automáticamente.

Por último la mejora que se plantea es la utilización del cron de Yii en el servidor. Básicamente lo que ofrecería esta mejora es llevar un control detallado de las actualizaciones que se realizan a una determinada hora del día y además realizar un backup de la aplicación.

7. Conclusión

La elaboración del proyecto ha supuesto un amplio estudio de las herramientas que se iban a utilizar junto a su diseño y sus algoritmos. En primer lugar se encontró la dificultad de utilizar un framework muy potente, que supone una adaptación previa a su estructura y uso. Otra dificultad añadida al proyecto fue cuando se decidió realizar la aplicación totalmente dinámica incorporando la creación de los propios formularios. Esta dificultad sin duda alguna planteó seriamente un gran reto personal debido a su complejidad a la que nunca se había enfrentado.

La duración del proyecto ha sido de cinco meses, aunque los primeros tres meses no se pudo desarrollar al 100% por la realización de prácticas de empresa, donde se profundizo en conocimientos de PHP. Una vez concluido el proyecto se valora positivamente el alto riesgo que supuso utilizar un framework desconocido hasta el momento, al igual que la gran variedad de lenguajes utilizados.

Con la elaboración de este proyecto también se ha dado la oportunidad de poner de manifiesto los conocimientos adquiridos en la carrera, como por ejemplo: un previo estudio a alumnos, un planteamiento mediante diagramas UML, la utilización de SQL, HTML y CSS, la utilización de cifrados como SHA y la optimización de algoritmos.

Para concluir, como opinión personal me satisface saber que todo lo aprendido a lo largo de la elaboración del proyecto no va a quedar en saco roto, ya que son tecnologías que se utilizan y se seguirán utilizando en un futuro. Por último, me es de gran orgullo poder colaborar en una aplicación que va a ser utilizada por la universidad, aportando un granito de arena a aquella que me ha dado los conocimientos necesarios para poder desarrollar este proyecto.

8. Bibliografía

[1] Página web dinámica. Disponible en: < http://janetes.jimdo.com/tipo-y-caracteristicas-pagina-web-dinamica/>. Fecha de consulta octubre de 2014.

[2] SGBD. Disponible en: < http://www.maestrosdelweb.com/que-son-las-bases-de-datos/>. Fecha de consulta octubre de 2014.

[3] HTML. Wikipedia. Disponible en: < http://es.wikipedia.org/wiki/HTML>. Fecha de consulta octubre de 2014.

[4] CSS. W3C. Disponible en: <http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo>. Fecha de consulta octubre de 2014.

[5] Bootstrap. Disponible en: < http://openwebcms.es/2013/que-es-bootstrap/>. Fecha de consulta octubre de 2014.

[6] MediaQuery. Wikipedia. Disponible en: <http://es.wikipedia.org/wiki/Media_query>. Fecha de consulta octubre de 2014.

[7] Javascript. Disponible en: < http://www.lcc.uma.es/~eat/services/html-js/manual14.html>. Fecha de consulta octubre de 2014.

[8] JQuery. Wikipedia. Disponible en: < http://es.wikipedia.org/wiki/JQuery>. Fecha de consulta noviembre de 2014.

[9] MySQL. Disponible en: <http://www.sinemed.com/recursos/docs/MySQL.pdf>. Fecha de consulta noviembre de 2014.

[10] PHP. Disponible en:< http://es.wikipedia.org/wiki/PHP>. Fecha de consulta noviembre de 2014.

[11] ¿Qué es Yii?. Disponible en: <http://es.scribd.com/doc/189203494/Comparativa-Framework>. Fecha de consulta noviembre de 2014.

[12] Funcionamiento Yii. Disponible en: < http://quimv-yii.blogspot.com.es/2012/05/fundamentales-yii.html>. Fecha de consulta noviembre de 2014.

[13] XAMPP. Disponible en: < http://es.slideshare.net/Kamisutra/presentacion-xampp>. Fecha de consulta noviembre de 2014.

[14] NetBeans. Disponible en: <https://netbeans.org/community/releases/61/index_es.html>. Fecha de consulta noviembre de 2014.

[15] GitHub. Wikipedia. Disponible en: < http://es.wikipedia.org/wiki/GitHub>. Fecha de consulta noviembre de 2014.

[16] Casos de uso. Disponible en: http://www2.uah.es/jcaceres/capsulas/DiagramaCasosDeUso.pdf>. Fecha de consulta noviembre de 2014.

[17] Asociación UML. Disponible en: <http://cespinozaj.files.wordpress.com/2011/12/casos-de-uso.pdf>. Fecha de consulta noviembre de 2014.

ANEXO A: Manual de instalación

En el siguiente manual se explicará la instalación y configuración de la aplicación en un servidor apache y el SGBD llamado PhpMyAdmin, utilizando la herramienta XAMPP mediante los siguientes pasos:

1. Acceder a la URL https://www.apachefriends.org/es/index.html
2. Seleccionar la versión que se desea descargar dependiendo nuestro sistema operativo.
3. Una vez descargado el archivo hay que ejecutarlo para abrir el proceso de instalación
4. En el proceso de instalación hay que observar que nuestro ordenador es totalmente compatible con las características que nos indica. Una vez realizado todos los pasos de la instalación procedemos a ejecutar el programa.
5. Antes de ejecutar el servidor apache es conveniente desactivar todas las aplicaciones que utilicen el puerto 443 y 80, por ejemplo Skype.
6. Una vez cerradas todas las aplicaciones seleccionamos el botón “Start” tanto en “Apache” como en “MySQL”
7. Una vez iniciadas las aplicaciones probaremos que funciona correctamente abriendo el navegador y accediendo a la URL *http://localhost/xampp/*
8. Para configurar la base de datos deberemos acceder al phpMyAdmin mediante la URL *http://localhost/phpmyuadmin/* donde crearemos un usuario para nuestra base de datos e importaremos el archivo SQL que se entrega en el proyecto.
9. Una vez realizado todos los pasos procederemos a situar nuestra aplicación en la ruta “C:\xampp\htdocs”.
10. Para relacionar nuestra aplicación con la base de datos debemos acceder al archivo “main.php” que se encuentra en la ruta “C:\xampp\htdocs\yii\tfg\protected\config”.
11. Indicamos el usuario y la contraseñas previamente creadas en:

```
'db' => array(
    'connectionString' => 'mysql:host=localhost;dbname=joboirui',
    'emulatePrepare' => true,
    'username' => 'joboirui',
    'password' => '**********',
    'charset' => 'utf8',
),
```

12. Por último solo falta comprobar que la aplicación funciona correctamente accediendo a la URL http://localhost/yii/tfg/