DPTO. INFORMATICA - I.E.S. SAN SEBASTIÁN
MÓDULO PROYECTO
C.F.G.S. DESARROLLO DE APLICACIONES WEB
TÍTULO DEL PROYECTO

AUROS ONLINE

Autor: Luis Javier Velázquez Murillo
Fecha: 10/6/2018
Tutor: Antonio José Vazquez Romero

ÍNDICE

1.Introducción

Los juegos de rol de hoy en día se basan en gráficos y una gran jugabilidad para, así, captar la atención completa de los jugadores. Aún así, a día de hoy, los juegos de rol de texto siguen teniendo popularidad entre aquellos jugadores que prefieren escribir ellos mismos sus aventuras.

AUROS es una migración con rediseño de un juego de rol web de texto llamado **Lifo** creado en 2006 por Luis Quesada Torres. Lifo fue creado como un proyecto personal que pretendía crear un sitio web que compartiese lo básico de un juego de rol y la funcionalidad de un foro online, dando así como resultado un juego simple pero adictivo, que permitía crear unir jugadores bajo una misma comunidad.

En 2014 el código fuente fue liberado, permitiendo así su libre uso y distribución. El fin del proyecto es reescribirlo y actualizarlo para que así permita más funcionalidades y esté a la altura de los juegos de hoy en día.

Para conseguir estos objetivos se ha decidido optar por el uso de tecnologías de software libre. Se ha usado el Framework Boostrapt 4(HTML, CSS, Bootstrap, y Javascript) para la interfaz, el framework Symfony 4.1 (PHP) para el programa y MySQL para la base de datos.

2.Estudio de viabilidad

2.1 Introducción

2.1.1 Descripción

El proyecto a realizar es una migración con rediseño, por lo que se necesitará realizar un análisis de código en una primera instancia para identificar el funcionamiento interno de la función, identificar los distintos bloques que componen a la misma y ser capaces de modular dichos componentes.

2.1.2 Objetivos del proyecto

El objetivo principal es realizar una aplicación funcional base, la cual sea capaz de evolucionar en base a los requisitos especificados por el cliente. En nuestro caso, un foro de rol con personalización de personaje, un gestor de usuarios, etc.

2.2 Estudio de la situación actual

2.2.1 Contexto

En la situación actual, tenemos un código escrito en 2006 para PHP 5.6 sin MVC, lo cual dificulta mucho su análisis. No es código autodocumentado y la funcionalidad es limitada, por no decir que además que el código tiene partes incompletas, mal escritas o genera errores no controlados

2.2.2 Lógica del sistema

El sistema se compone básicamente de 4 módulos diferenciados relacionados con el sistema(Foro, Sesiones, Mensajería, Gestión de usuarios) y de 2 módulos adicionales, en este caso, relacionados con el sistema de juego(Sistema de juego y Sistema de clanes).

Además, existe un módulo externo de control de todo el sistema, englobando este a los paneles de administración de los administradores y moderadores de la aplicación.

Dentro de estos módulos a su vez, todos los procesos seguiran un mismo patrón lógico a la hora de la realización de sus funciones.

(Usuario => Función => Cambio en el programa => Respuesta)

2.2.3 Diagnóstico del sistema actual

Dada la antigüedad del código y de la dificultad que conlleva su interpretación, se realizará una anáilisis de las funciones en base a las vistas finales(EndPoints) de la própia aplicación.

Se adjuntará una carpeta de anexo con todas las capturas realizadas sobre los distintos puntos finales.

2.3 Requisitos del sistema

2.3.1 Requisitos

Al realizar la aplicación en Symfony, se necesitará un gestor de dependencias como composer. La instalación de la aplicación debe de ser por medio de un terminal, ya que se debe ejecutar dicho comando cada vez que se quiera desplegar la app o hacer una migración a la base de datos.

Además se necesitará una base de datos MySQL para poder almacenar la información.

2.3.2 Restricciones del sistema

Siguiendo el punto anterior, hay pocas opciones a la hora de desplegar la aplicación, puesto que el framework que se va a usar limita los hosting que vayan a elegir en un futuro.

No se podrá usar un hosting normal para desplegar la aplicación, se precisará de un Cloud, que cuente con SSH para poder ejecutar comandos y así poder desplegarla.

Además, de esta forma, podemos instalar las dependencias que necesitemos o nos será más facil crear subdominios si lo precisásemos.

2.3.3 Catalogación y priorización de los requisitos

Siguiendo la categorización anterior para la división de los módulos del programa, la prioridad durante el desarrollo será:

1. Creación del módulo gestor de usuarios
2. Creación de las sesiones
3. Sistema de mensajería de la aplicación
4. Foro completo para usuarios
5. Paneles de control para administradores y moderadores
6. Sistema de agrupación por clanes entre jugadores
7. Sistema de juego para los personajes de los usuarios

Como se puede apreciar por el orden de prioridad, los dos últimos puntos que se realizarán son totalmente modificables. Aunque esta aplicación está destinada a ser un juego de rol por texto, si modificamos los dos últimos bloques, podemos extrapolar la aplicación a otra completamente distinta.

2.4 Recursos del proyecto

Respecto a los recursos con los que cuenta el proyecto, tenemos el código fuente del proyecto liberado en 2014, el cual es la base de toda la aplicación y el própio proyecto. Se puede descargar desde el siguiente enlace: LifoSource

2.5 Evaluación de riesgos

2.5.1 Lista de riesgos

Los riesgos que puede tener la aplicación tienen que ver con los propios que pueden tener un foro, siendo estos por ejemplo el spam masivo por parte o hacia los usuarios. Se debe tener en cuenta que los usuarios pueden explotar las funcionalidades de la aplicación, como el envio de mensajes a otros jugadores o los post en el foro.

2.5.2 Plan de contingencia

Siguiendo el punto anterior, ¿se puede controlar para evitar esto? Hay varias soluciones:

1. Se puede aplicar un nivel mínimo a los personajes de los usuarios para realizar dichas acciones.
2. Se puede limitar el envio de mensajes a otros jugadores o en el foro mediante un número dentro de un periodo establecido, por ejemplo, 10 mensajes en total cada media hora.
3. Igualmente, los moderadores tienen la función de silenciar a los jugadores que tengan conductas contrarias al foro. Así que, también cuentan como una medida de seguridad del juego.

2.6 Costes

A continuación analizaremos los distintos costes que conllevará el desarrollo de la aplicación o su mantenimiento.

2.6.1 Coste material

Dado que el uso del software es libre y de que suponemos que se dispone de un equipo básico para el desarrollo del programa, como costes materiales, tendremos en cuenta sólo el alquiler del cloud y la compra annual del dominio.

En godaddy.com podemos comprar un cloud por 3.95€ el primer año, incluyendo este precio además el dominio. Haremos el cálculo para 1 año.

2.6.2 Coste personal

Para poder rellenar este apartado, primero se ha tenido que esperar a la realización completa del programa y así conocer la duración total que ha llevado el desarrollo del mismo.

Se ha trabajado una media de 2 horas y media cada día contando los fines de semana durante tres semanas. Suponiendo que la media de lo que cobra por hora completa de trabajo un desarrollador web es de 9.5€, podemos calcular cuanto ha costado la mano de obra.

	Días	Horas	Coste
Programación	21	52.5	498.75 €
Cloud	365	-	3.95 €
TOTAL:	-	-	502.7 €

2.6.3 Resumen y análisis coste beneficio

Un coste de 502.7 € no es un alto precio para una aplicación de tal densidad. Además debemos tener en cuenta que el modelo de negocio es Freemium, y haciendo un poco de publicidad a la misma para darle “tirón”, se multiplicarían las ganancias.

2.7 Conclusiones

En conclusión, ¿es rentable la apicación? En los últimos años, el modelo de negocio freemium ha llegado a su punto más alto de popularidad, teniendo este un éxito inmenso en aquellos juegos que lo siguen.

Si se hace que sea atractiva y se lleva un mantenimiento continuo sobre la misma, se puede confirmar que en poco tiempo reportará beneficios que supliran los costes de creación por bastante diferencia.

Hay un inconveniente principal, y ese es la campaña de marketing que hay que realizar en una primera instancia para dar a conocer el juego, pero eso no lo tendremos en cuenta a la hora de realizar cálculos, ya que esto es un proyecto sobre un programa y su realización, no sobre su propaganda.

3.Análisis

3.1 Introducción

A continuación vamos a exponer las funcionalidades requeridas por el sistema en base a los descrito anteriormente y al código fuente facilitado.

3.2 Requisitos funcionales

Funcionalidades disponibles en el sitio web:

1. **Gestión de usuarios**: Permite gestionar los usuarios registrados en Auros.
    - Alta usuarios
    - Edición usuarios
    - Silenciar usuarios
    - Quitar silencio usuarios
    - Banear usuarios
    - Quitar baneo usuarios
    - Dar moderación
    - Quitar moderación
2. **Gestión de personajes**: Permite gestionar los personajes asociados a los usuarios.
    - Crear personaje
    - Listar personajes
3. **Gestión de mensajería**: Permite gestionar la mensajería de un personaje
    - Crear mensaje
    - Listar mensajes
    - Leer mensaje
    - Borrar mensajes
    - Reportar mensaje
4. **Gestión de foro**: Permite gestionar el foro de personajes
    - Crear tema
    - Borrar tema
    - Crear Post
    - Ver Tema

5. **Gestión del acciones de juego**: Permite gestionar las acciones de juego de los personajes dentro de la aplicación.
    - Trabajar
    - Comer
    - Vender
    - Comerciar
    - Combatir

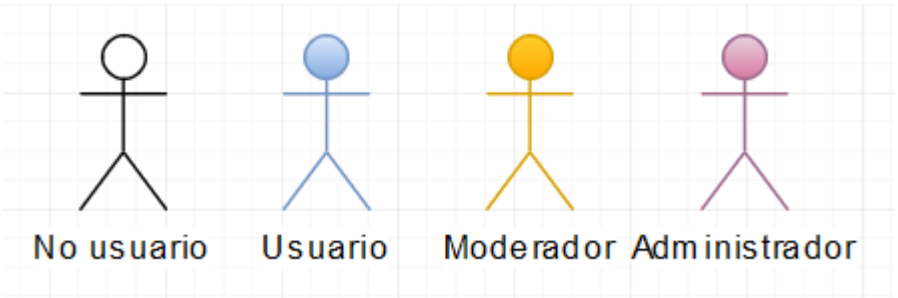
3.3 Requisitos no funcionales

La aplicación debe de ser completamente compatible con programas de lectura de pantalla. Esto es debido a que los juegos de rol de texto tienen un gran número de jugadores invidentes. La mayoría buscan una experiencia que no pueden conseguir en los juegos con gráficos normales, puesto que no están diseñados ni programados para ello.

3.4 Diagramas de casos de uso / Casos de uso

En un diagrama de casos de uso, existen los actores, que son entidades ajenas al sistema que tienen la capacidad de interactuar o comunicarse con el mismo.

Hay 4 tipos de actores en Auros. Tenemos a los usuarios no registrados, que tienen acceso a la información pública de la aplicación, como el registro o la wiki. Luego tenemos a los usuarios, que son aquellos que tienen acceso a la mayor parte del contenido de la misma. Los moderadores son usuarios con ciertos privilegios y con acceso al panel de moderación, desde donde pueden gestionar los silencios y reportes de los usuarios. Por último, estan los administradores, que son aquellos usuarios que tienen acceso completo a toda la aplicación, pudiendo incluso banear el acceso de los jugadores a la aplicación.



Actores Diagrama Casos de Usos

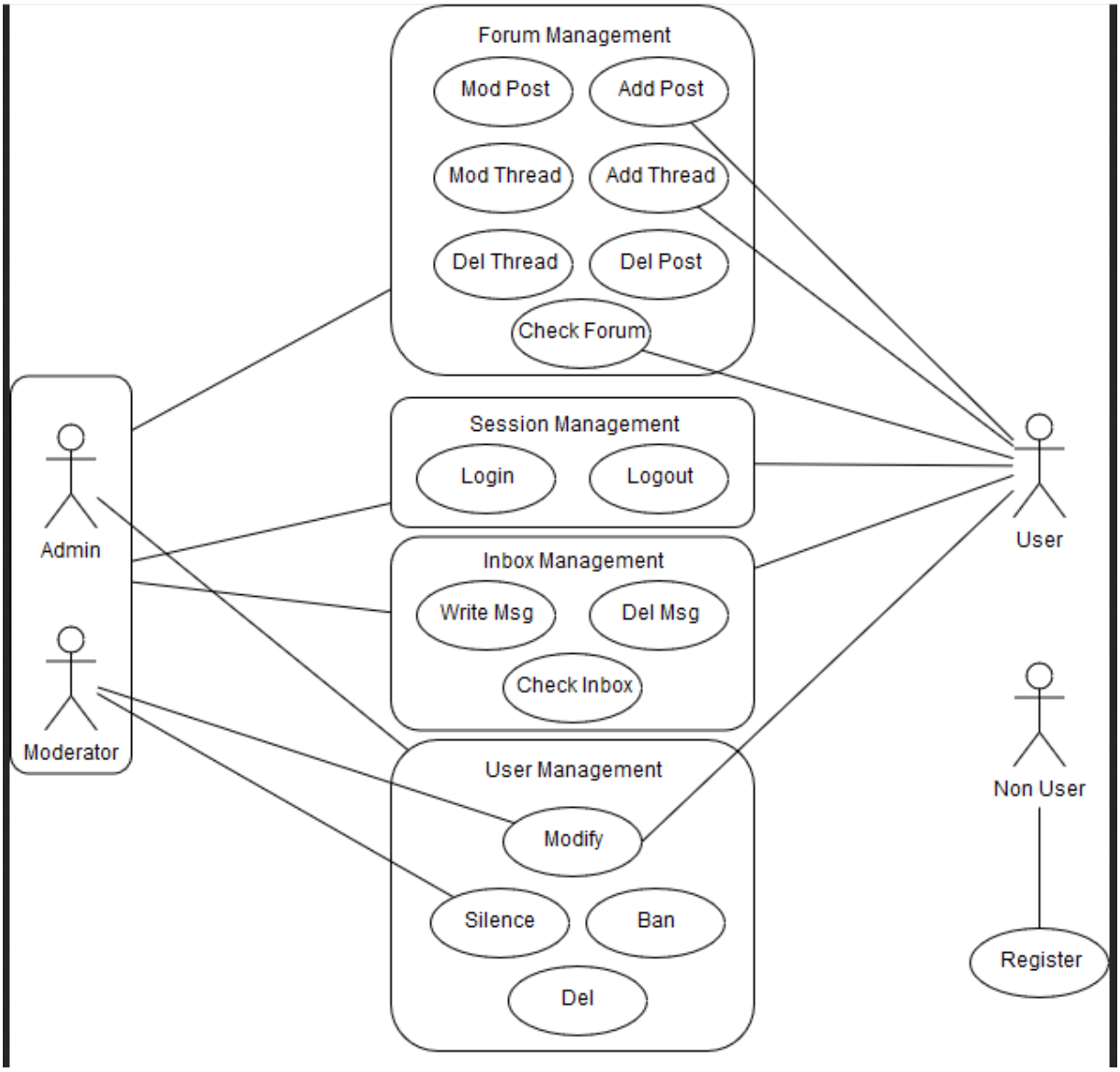


Diagrama Casos de Usos. Parte 1

En la primera parte del diagrama podemos apreciar los distintos bloques que componen la aplicación y los distintos accesos que tienen cada actor a la aplicación. Los no usuarios por ejemplo, solo pueden registrarse.

Aquí se visualizan los bloques de gestión de usuarios, de mensajería, de sesión y de foro.

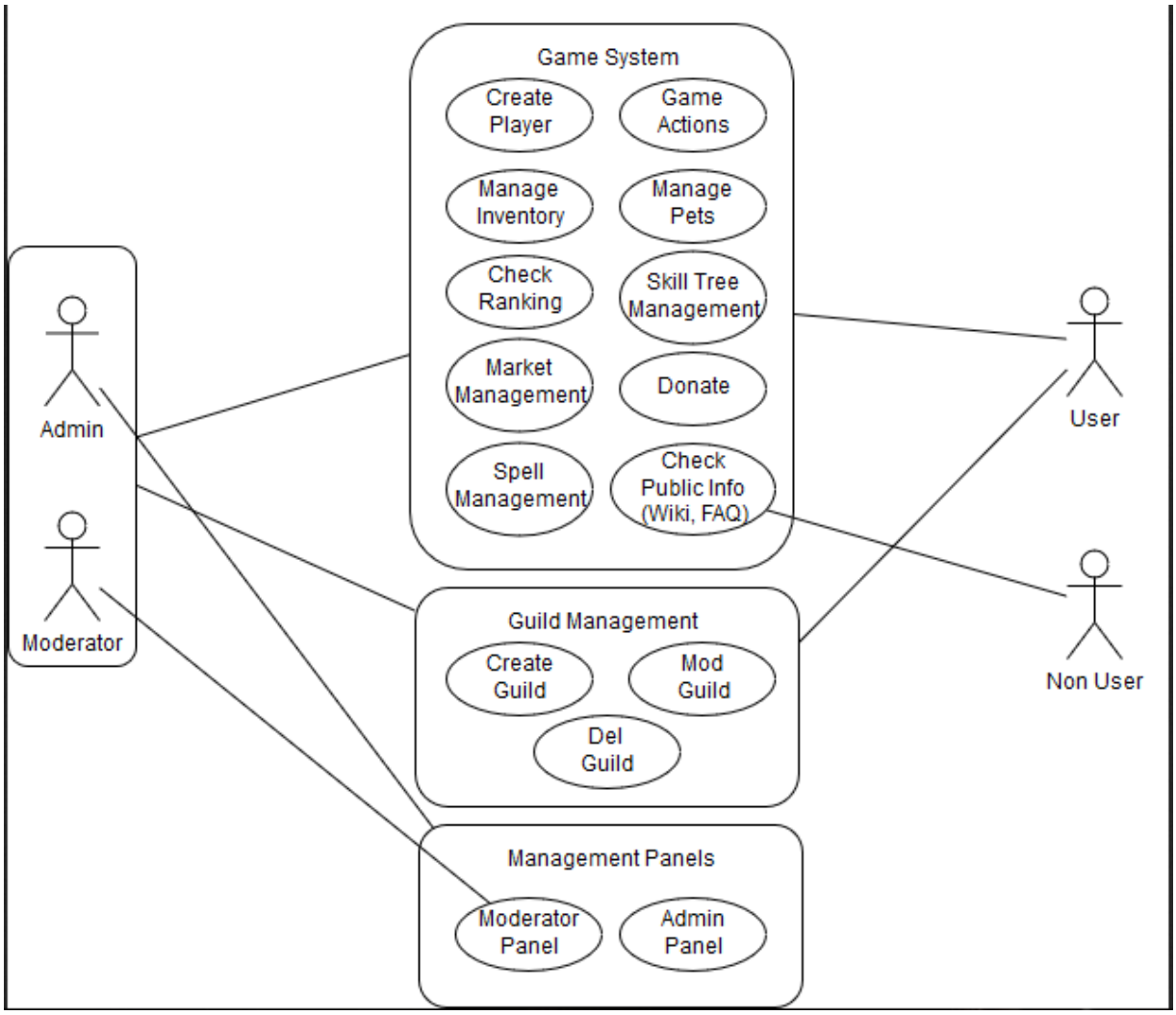


Diagrama Casos de Usos. Parte 2

En la segunda parte del diagrama, apreciamos los 3 módulos restantes. La gestión del sistema de juego, la gestión de los clanes y la gestión de paneles.

Los paneles de control, como anteriormente se explicó, son solo para los moderadores y para los administradores, ya que cuentan con herramientas de control de usuarios, del foro o del juego.

3.5Diagrama de entidad-relación

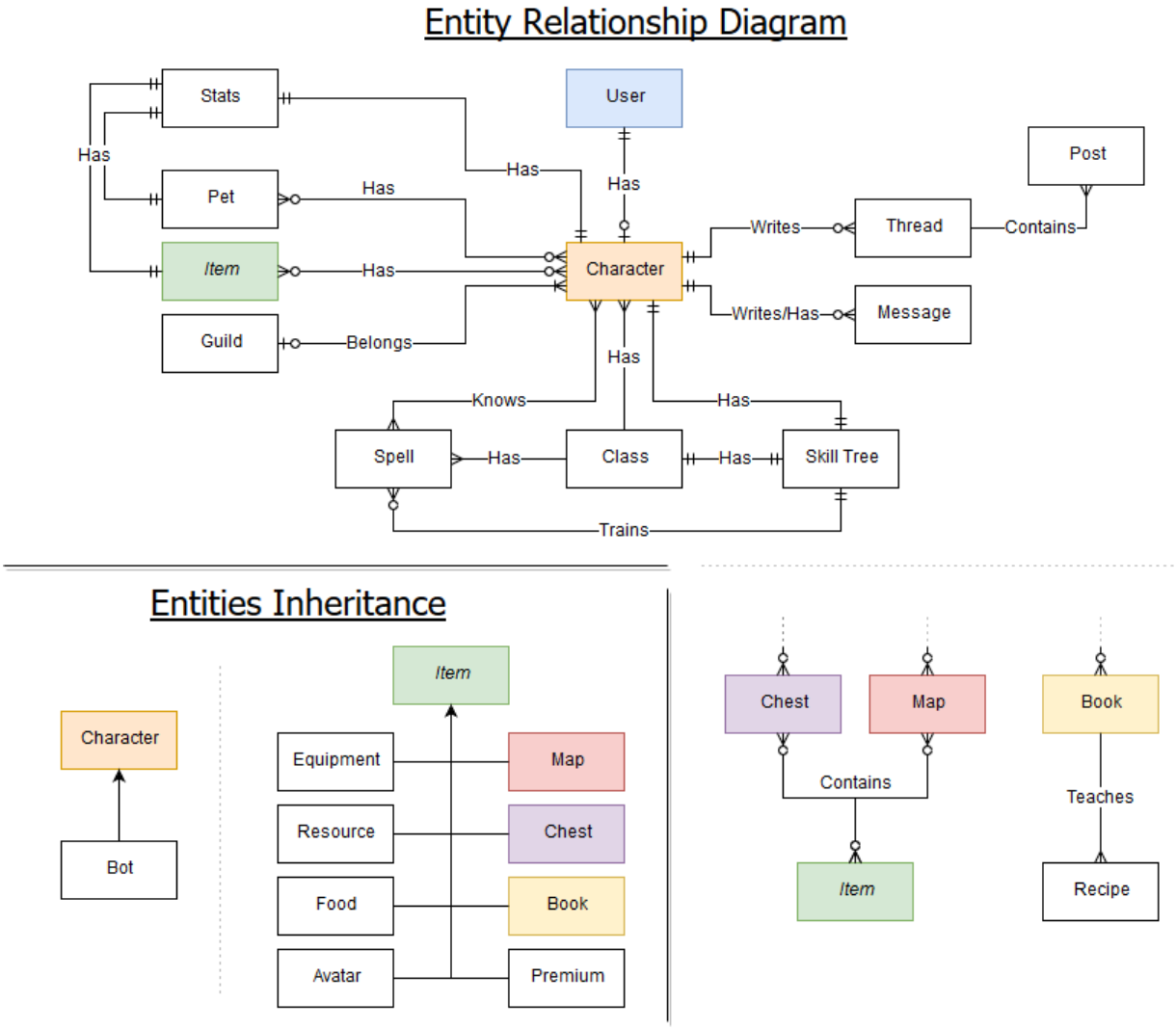


Diagrama entidades.

Tras un análisis del código y de los existente anteriormente, sacamos este diagrama de relaciones entre entidades siendo, además, un diagrama representativo de las própias clases.

La entidad principal y a la cual todo gira en torno es Personaje, pero para que exista un personaje, primero debe existir un usuario.

A continuación analizaremos los detalles más destacables del diagrama:

- Un usuario puede tener o no un personaje, pero siempre un personaje pertenece a un usuario.

- Los personajes escriben o reciben mensajes, pero siempre son emitidos por un personaje.
- Todos los personajes tienen un tipo clase, un tipo de arbol de habilidades y muchos tipos hechizos.
- Todos los tipos de clases tienen asociadas siempre un tipo de arbol de habilidades, y todas las clases tienen muchos tipos de hechizos.
- Todos los hechizos son exclusivos de un tipo de clase, aunque, un arbol de habilidades puede enseñar un hechizo.
- Los personajes, los distintos tipos de mascotas y los tipos de objetos siempre tienen unos stats y un stat siempre pertenece a uno de ellos.
- Los personajes pueden o no pertenecer a un clan, y a un clan pertenecen siempre de uno a muchos personajes.
- Un personaje tiene muchos tipos de objetos y un tipo de objeto puede tenerlo muchos personajes.
- Los personajes, además, pueden ser Bots, que significa que son usuarios de prueba para que interactuen con otros usuarios.
- Los items pueden tener distintas clases, como comida, equipamiento, etc.
- Los cofres y los mapas contienen a otros items.
- Los libros enseñan recetas, y las recetas siempre son enseñadas por un tipo de libro.

3.6Esquema de base de datos

Al ser una imagen demasiado grande y compleja de dividir para insertar en este documento, se ha optado por dejarla a parte como anexo en la carpeta de documentación.

En la imagen del esquema de base de datos, podemos apreciar de forma desglosada las relaciones y las entidades mencionadas anteriormente con todas sus propiedades.

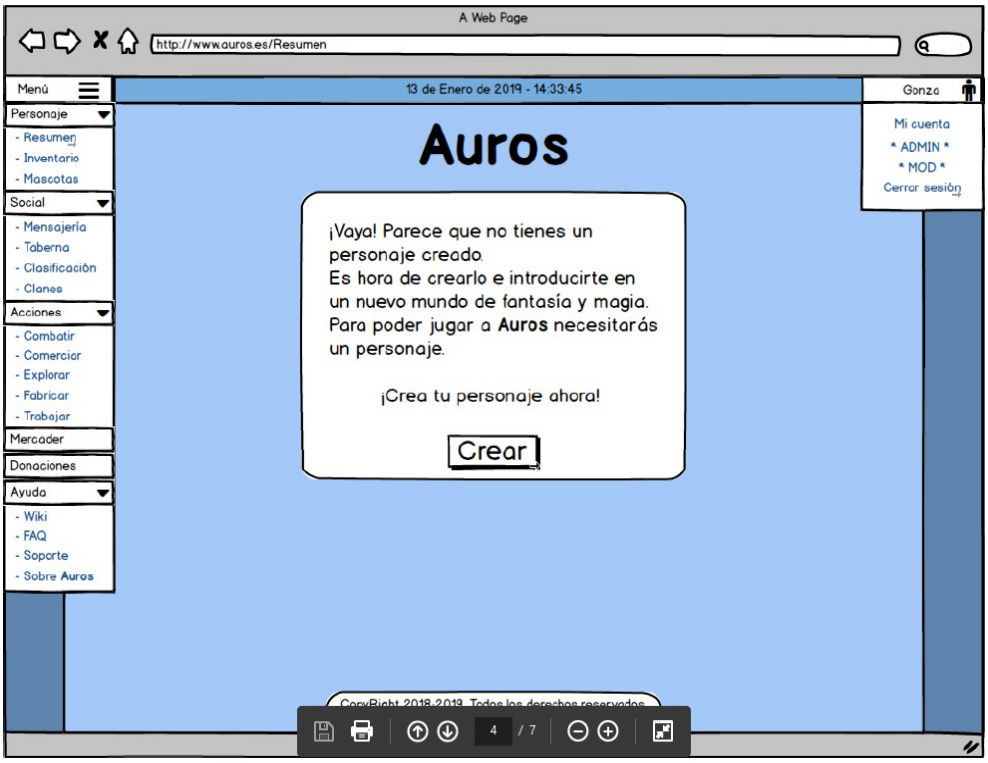
Las tablas intermedias como *haspets* y *haveitems* represetan la tabla intermedia de colecciones de dichas entidades para los jugadores.

3.7Prototipos

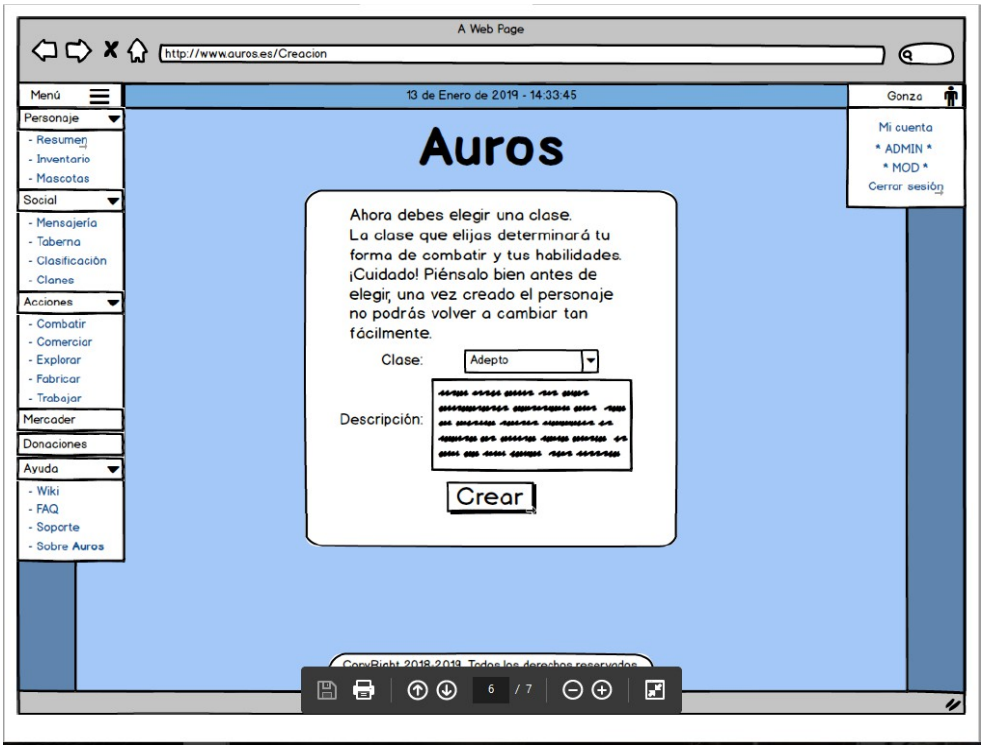
Se ha realizado un prototipo básico para representar el concepto básico del diseño que se busca en la app. Por ello, se ha usado la aplicación Mockups para diseñar una serie de plantillas.

Ahora veremos una series de capturas:

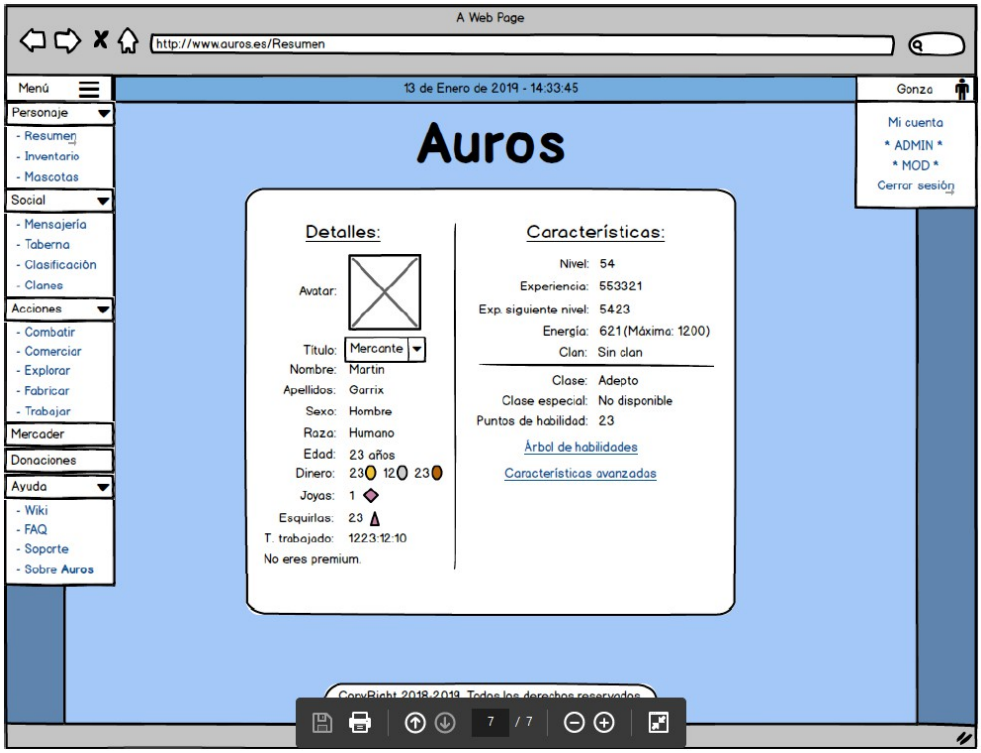
Concepto 1: Pantalla de inicio.



Concepto 2: Pantalla tras logeo.



Concepto 3: Pantalla de creación de personaje.



Concepto 4: Pantalla Resumen Personaje.

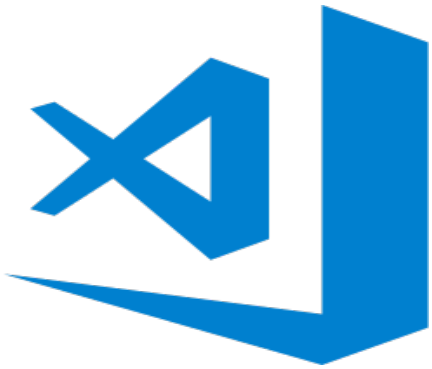
4.Diseño

4.1 Introducción

En este apartado, vamos a plantear las distintas cuestiones con respecto a como se va realizar el diseño de la própia aplicación, es decir, su desarrollo.

4.1.1 Selección del entorno de desarrollo

Para el desarrollo de la aplicación, se ha optado por usar Visual Studio Code, un IDE gratuito que permite trabajar de una forma facil, rápida y sencilla con las carpetas de documentos. Dispone de gran cantidades de extensiones, soporte para git y con un terminal incorporado.



En nuestro caso, hemos usado el terminal para ejecutar los comandos de composer y los de symfony.

4.1.2 Selección de base de datos

Para la base de datos optaremos por MySQL. La elección se debe a que ya estamos familiarizados con ella, conocemos su potencia y su sintaxis, y eso reduce tiempo de desarrollo. Además, el hosting de godaddy ofrece MySQL, por lo que es una de las mejores opciones para el desarrollo.

4.2 Configuración de la plataforma

Para la configuración de la plataforma, debemos ir a nuestro cloud, donde tengamos alojada la aplicación y establecer los parámetros necesarios para la misma. Estos pueden ser como la versión de PHP, las bases de datos, los servicios externos, plugins, extensiones, etc.

4.3 Capas de la aplicación

Para Auros, se contemplan 3 capas bien diferenciadas entre sí:

1. El interfaz, que es con lo que interactúa el usuario.

2. La lógica del programa, que es la que maneja todo el back-end.

3. La capa de datos, que es la que almacena toda la información.

4.4 Arquitectura de la aplicación

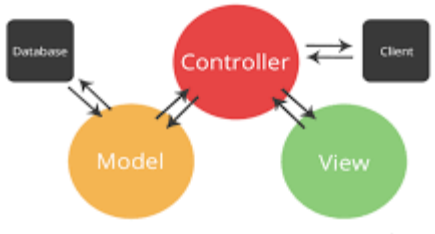
Symfony 4.1, el framework de PHP, usa el modelo MVC (Modelo-Vista-Controlador).

Este patrón de arquitectura permite separar en distintas capas el programa, y así dividir la lógica de los datos, así como también lo separa del interfaz de los usuarios.

El modelo interactúa con todo lo relacionado con los datos, el acceso a la base de datos, su conexión, las consultas, las inserciones, etc.

Las vistas son los puntos finales que visualiza y con los que interactúa el usuario final.

Por último, el controlador se encarga de unir las dos capas anteriores, permitiendo así una capa de abstracción y seguridad a la aplicación.



Concepto 4: Patrón MVC.

5.Implementación

5.1 Introducción

Este apartado tratará las distintas tecnología empleadas en las distintas capas de la aplicación. Todas ellas son Software Libre.

5.2 Codificación

5.2.1 Parte Cliente

HTML:

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

CSS:

"Hoja de estilo en cascada o CSS (cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML). El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores." Wikipedia

JAVASCRIPT:

"Este JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMA Script. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

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

5.2.2 Parte Servidor

APACHE:

"El servidor HTTP Apache es un servidor web HTTP modular, multiplataforma, de código libre, extensible y resulta muy fácil conseguir ayuda y soporte. Apache se muestra como un servidor muy competitivo, rápido, estable y con más ventajas que otros servidores, incluso en entornos con millones de visitas al día. " Wikipedia

PHP:

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

SYMFONY:

"Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. " Wikipedia

MySQL:

"MySQL es una base de datos relacional gratuita que se distribuye bajo un sistema de doble licencia. Se puede obtener gratuitamente bajo licencia GPL y en el caso en el que se quiera distribuir una aplicación que no cumpla los términos GPL pero que incluya MySQL existe también una licencia comercial. " Wikipedia

6.Conclusiones

6.1 Conclusiones finales

Ya terminado el proyecto, tenemos una web funcional que resuelve los problemas planteados al inicio del proyecto. Los usuarios disfrutan de un juego sencillo, tanto usuarios normales como invidentes. Los administradores y moderadores manejan de forma sencilla y eficaz el juego.

Gracias a la modularidad del proyecto, éste es extrapolable a otros tipos de juegos, o no tienen por qué ser juegos, pueden ser otro tipo de aplicaciones.

6.2 Desviaciones temporales

Debido al corto periodo de tiempo de desarrollo con el que ha contado el proyecto, se ha omitido casi en su completa totalidad el módulo de juego, quitando así una parte bastante importante del juego. Aún así el juego permite su uso completo del foro y otras funcionalidades.

6.3 Posibles ampliaciones y modificaciones

Como posibles ampliaciones, cabría decir que se puede completar el módulo de juego que falta, así como añadir otras funcionalidades ni siquiera planteadas anteriormente. Por ejemplo, configurar un servidor de correo para que el administrador pueda enviar correos a sus usuarios, ya sean con avisos o con información.

6.4 Valoración personal

En mi opinión, es un proyecto ambicioso y con mucho potencial, con algo de tiempo puede convertirse en una gran aplicación y retribuir buenos beneficios a sus desarrolladores.

Igualmente, se vé cláramente que necesita tiempo, es posible que siga, en lo personal, trabajando en el mismo.

7.Bibliografía

1. Google. (2018) https://www.google.es/
2. Wikipedia https://es.wikipedia.org/wiki/Wikipedia:Portada
3. Symfony 4.1 https://symfony.com/
4. PHP http://php.net/
5. MySQL https://www.mysql.com/
6. W3Schools https://www.w3schools.com/
7. Bootstrap 4 https://getbootstrap.com/docs/4.0/getting-started/introduction/