

Telefónica

Instituto Tecnológico Telefónica



MECATOL HEX

PROYECTO DE DESARROLLO DE
APLICACIONES WEB

Técnico superior en desarrollo de aplicaciones Web.

Autor:

Mikolaj Bernecki

Tutor:

Félix de Pablo Lobo

Introducción

En este documento se describe obras realizadas durante el proyecto final del curso de formación profesional en desarrollo aplicaciones web. La idea de proyecto es crear una solución en forma de aplicación Web para gestionar la colección de juegos de mesa a una asociación de aficionados de juegos - Mecatol Rex.

El objetivo de dicha aplicación es gestionar los datos de los juegos que alberga la asociación en su local y proporcionar una interfaz gráfica en forma de una página Web para que socios y la junta de la asociación puedan acceder a esos datos. La web es accesible desde cualquier navegador, se enfoca el desarrollo para dispositivos móviles, sobre todo tablets, para poder realizar las consultas en el local de la asociación.

La aplicación tiene dos tipos de usuarios – un administrador único que representa la comisión de ludoteca del club y los usuarios registrados-socios. El acceso a la aplicación está restringido, no se puede acceder sin registrarse.

Contexto

La asociación Mecatol Rex es un grupo de aficionados a los juegos de mesa que tiene su sede en Madrid. La asociación tiene más de 150 socios, que frecuentan la sede, donde puede haber hasta 50-60 personas en mismo tiempo. En su local albergan la colección de juegos que se pueden jugar en el sitio y no se pueden prestar para llevar, pero a veces la asociación los lleva para eventos fuera del club – convivencias en el campo.

Ahora mismo dicha colección consiste en más de 25 estanterías donde cada estantería puede almacenar cerca de 50 títulos. Los juegos tienen varios propietarios, pueden ser individuales o lo puede ser la misma asociación.

Índice

1. [Introducción](#)
2. [Índice](#)
3. [Módulos formativos aplicados en el trabajo](#)
4. [Objetivos del proyecto](#)
5. [Herramientas/Lenguajes utilizados](#)
6. [Fases del proyecto](#)
7. [Conclusiones/Resultado final del proyecto](#)
8. [Bibliografía](#)

Módulos formativos aplicados en el trabajo

Programación

La parte prominente del proyecto fue realizada en lenguaje Java. Conocimientos de este módulo sirvieron para comprender la organización de un proyecto Java y sus formas de estructurar datos: paquetes, clases, tipos de variables.

Entornos de Desarrollo

En este proyecto se utilizó dos entornos: IDE de Eclipse y editor de VS Code. Se aplicaron los conocimientos de características (generación de estructuras de clase, recuperación de ficheros, depuración) de esos programas para agilizar el desarrollo.

Bases de datos

El núcleo del programa fue creado en base de lo que he aprendido de la creación de tablas y relaciones entre ellas en este curso. Aunque las consultas fueron realizadas en JPQL, la forma de declararlas se basa en SQL enseñado durante este curso.

Lenguajes de marcas y sistemas de gestión de información

Proyectos de Web en Java tienen partes donde se aplica de forma extensa XML – declaraciones de dependencias de Maven o definiciones de servlets (se consideraba esta opción, pero fue descartada porque no es una herramienta muy ágil – el motor tarda bastante en interpretar cualquier cambio). Una parte a que no se ha llegado a realizar, pero en cual sería muy crucial el parseo de XML fue la interpretación de respuestas de servidor de API.

Diseño de interfaces Web

Gracias a este módulo conocí a varios programas de modelación (wireframing) de diseño y la metodología de realizar esta tarea. Aunque se realizaba con otra herramienta, la forma de proceder fue muy parecida.

Despliegue de aplicaciones Web

El proyecto aun no está desplegado, pero se aplicó los conocimientos de servidores localhost para realizar el desarrollo.

Desarrollo Web en entorno Cliente

Conocimientos de JavaScript se utilizaron para crear la parte de gestión de dirección del usuario en la API de Google Maps. También se utilizó herramientas de desarrollador de navegadores para gestionar cambios de estilo.

Desarrollo Web en entorno Servidor

El programa está realizado en parte de servidor, por lo tanto, es el eje de todas transacciones realizadas con la base en forma de patrón DAO (Data Access Object) con el uso de nuevas APIs que conocí durante el curso – Java Persistence en forma de EclipseLink para hacer mapeo objeto-relacional (ORM), Criteria Query para realizar consultas dinámicas. En el futuro el proyecto se va a transformar en un servicio de Java con endpoints JSON para poder crear una Single Page Application en alguna de librerías Front para agilizar la renderización y quitar al servidor la carga de formar respuestas HTML.

Objetivos del proyecto

Por parte de capa de datos

Gestionar flujo de copias de juegos en el club para facilitar selección de un juego para usuarios y control de copias en el club por parte de la junta de club.

Usuarios pueden:

- ⦿ buscar juegos en el club para obtener datos de la estantería donde se ubican y datos principales que un grupo de jugadores debe afrontar: tiempo de una partida, máximo de personas, complejidad, dependencia lingüística de un juego.
- ⦿ prestar copias para jugar y así marcar un juego como no disponible para los demás en este momento
- ⦿ ver copias de sus juegos prestados al club para el uso general
- ⦿ gestionar sus datos de contacto.

Administrador puede:

- ⦿ Registrar nuevas copias de juegos aportadas o compradas por el club.
- ⦿ Cambiar datos de un juego.

Por parte de capa visual

Proporcionar una interfaz limpia y útil siguiendo las reglas expuestas en el libro de “No me hagas pensar” de Steve Krug¹:

- ⦿ Jerarquía visual clara – marcar la importancia del contenido con color y tamaños
- ⦿ Convenciones – La interfaz corresponde a lo que usuario reconoce de su ámbito
- ⦿ Dejar bien claro cuáles son elementos con los que se puede interactuar
- ⦿ Evitar el ruido visual
- ⦿ Omitir palabras innecesarias

El motivo principal de seguir estas reglas es poder proporcionar una interfaz que no pesa mucho en el móvil o tablet, donde se va a renderizar.

Herramientas/Lenguajes utilizados

Lenguajes:

- ☉ **Java**: JSTL para crear vistas de respuestas, JPA para persistencia, JPQL para realizar consultas
- ☉ **MySQL** – para crear base de datos y realizar consultas de prueba
- ☉ **XML** – para configuraciones de dependencias
- ☉ **JavaScript** – para realizar consultas con API de Google Maps
- ☉ **SCSS/Sass** – para crear hojas de estilo
- ☉ **Git** – para guardar estado final y realizar control de versiones en el futuro

Herramientas:

Entorno servidor:

- ☉ **Eclipse for Enterprise Java Developers** – el IDE de Java que permite crear proyectos Web J2EE
- ☉ **phpMyAdmin** – capa gestora del servidor Apache para manejar bases de datos MySQL, tiene bien resuelto el formulario de inserción de datos, cuando queremos probar registros con datos abstractos, compuestos principalmente de llaves foráneas.
- ☉ **MySQL Workbench** – herramienta oficial de Oracle para manejar bases MySQL, tiene bien resuelta parte de consultas y inserciones masivas de datos (se puede copiar directamente desde una hoja Excel, por ejemplo) o – por otra parte, representaciones visuales de modelo.
- ☉ **Wampserver64** – la herramienta de crear servidores Apache con bases MySQL en el entorno local. Tiene como ventaja el soporte para versión 8 de MySQL, que es mucho más eficiente.
- ☉ **MvnRepository** – el repositorio de dependencias de Java que utilice para descargar drivers de conexión a la base de datos, clases de utilidades tipo JavaMail, Apache commons.

Capa visual

- © **Vista desarrollador de Firefox y Chrome** – vienen muy útiles a la hora de crear estilos, tienen herramientas integradas de generar polígonos (en este caso hexágonos), sombras, colores en tiempo real o controlar display flex y grid.
- © **Consola y Sass** – compilación de hojas de estilo con anidación y imports más eficientes (el import por defecto de CSS genera varias peticiones, el import de Sass incluye la hoja adicional dentro de la hoja generada), bucles y simplificación de sintaxis de calculaciones.
- © **Visual Studio Code** – su ventaja frente a Eclipse son extensiones de validación y Intellisense de SCSS y CSS.
- © **inVision Studio** – utilizada para hacer el wireframe de la página.
- © **Inkscape** – utilizada para crear iconos de svg.

Fases del proyecto

Wireframing y diseño de vistas de la aplicación

El diseño de vistas se realizó en inVision – una aplicación en línea que permite visualizar transiciones entre vistas particulares al interactuar con la interfaz.

La estructura de la página es siguiente (sangrías marcan subpáginas):

Entrada (Login)

Recuperación de contraseña

Inscripción

Página principal

Opciones de perfil

Edición de datos de perfil

Busqueda de colección del usuario

Prestamos del usuario

Búsqueda de juegos

Página de opciones de admin

Gestión de colección

Edición de un juego

Hay dos zonas principales que corresponden a acciones:

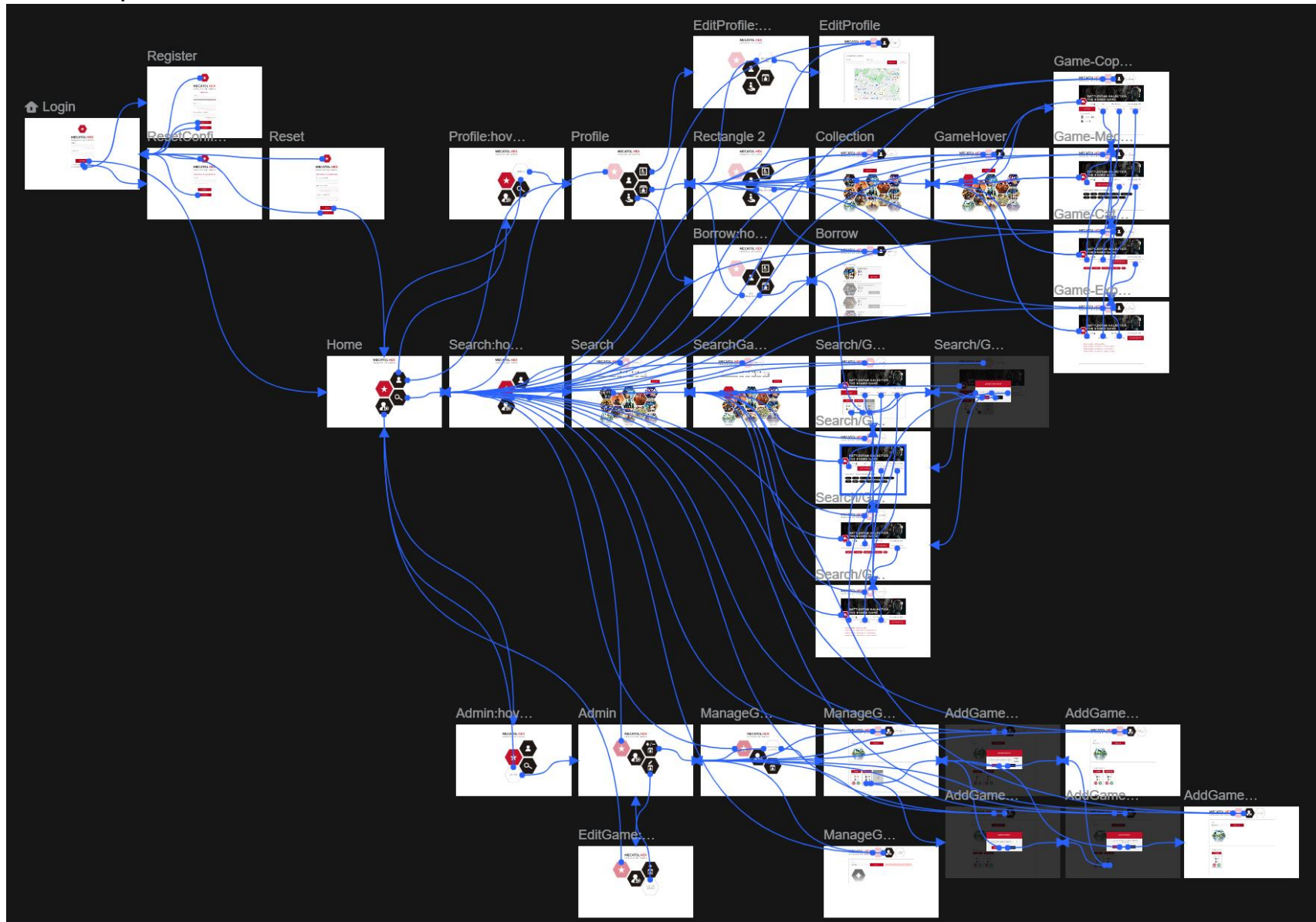
Login

Acciones de usuarios (pág.principal)

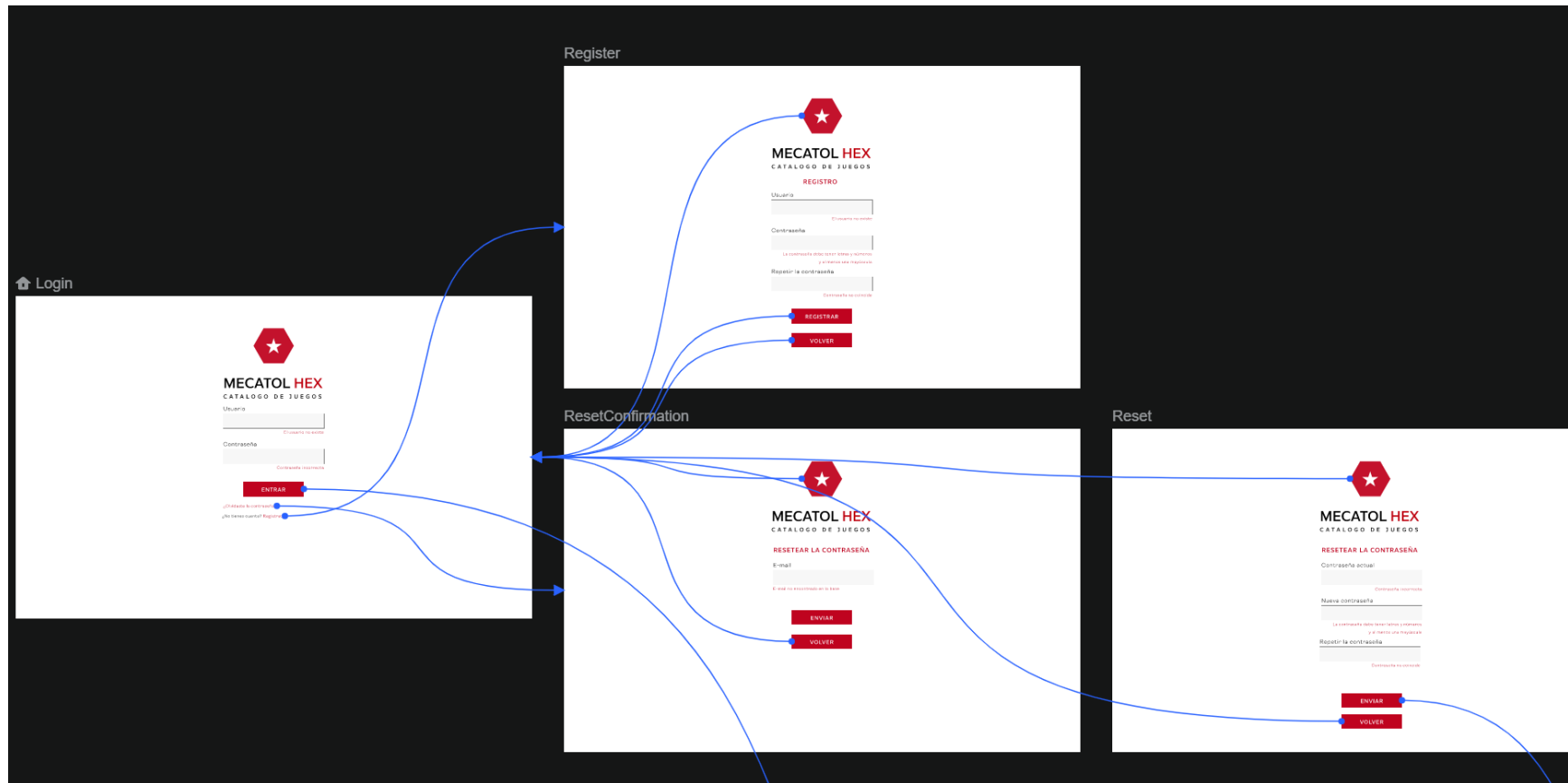
Las **acciones** se realizan en zonas marcadas con color **negro**.

El esquema junto con acciones en la página siguiente ...

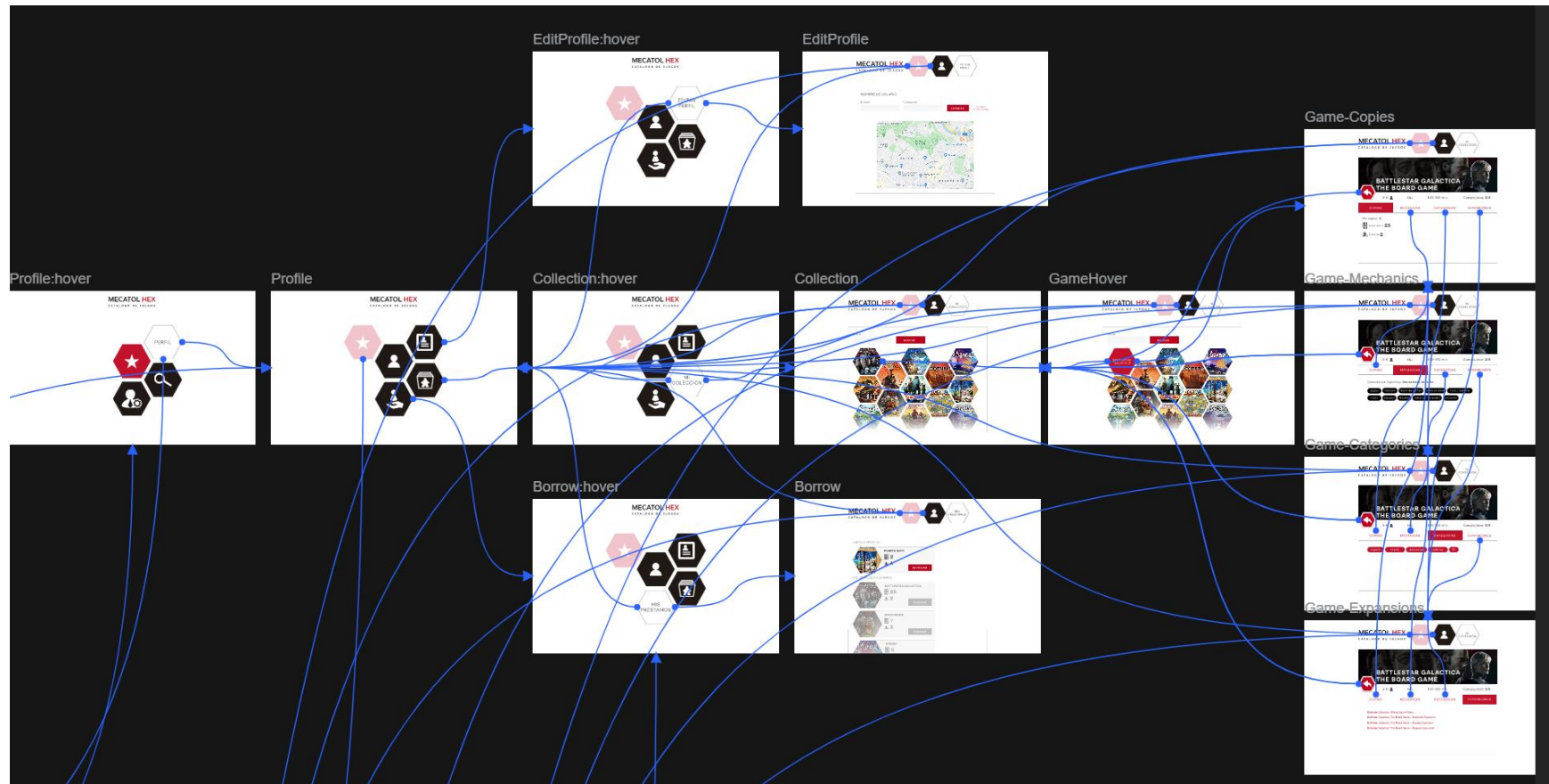
Vista completa:



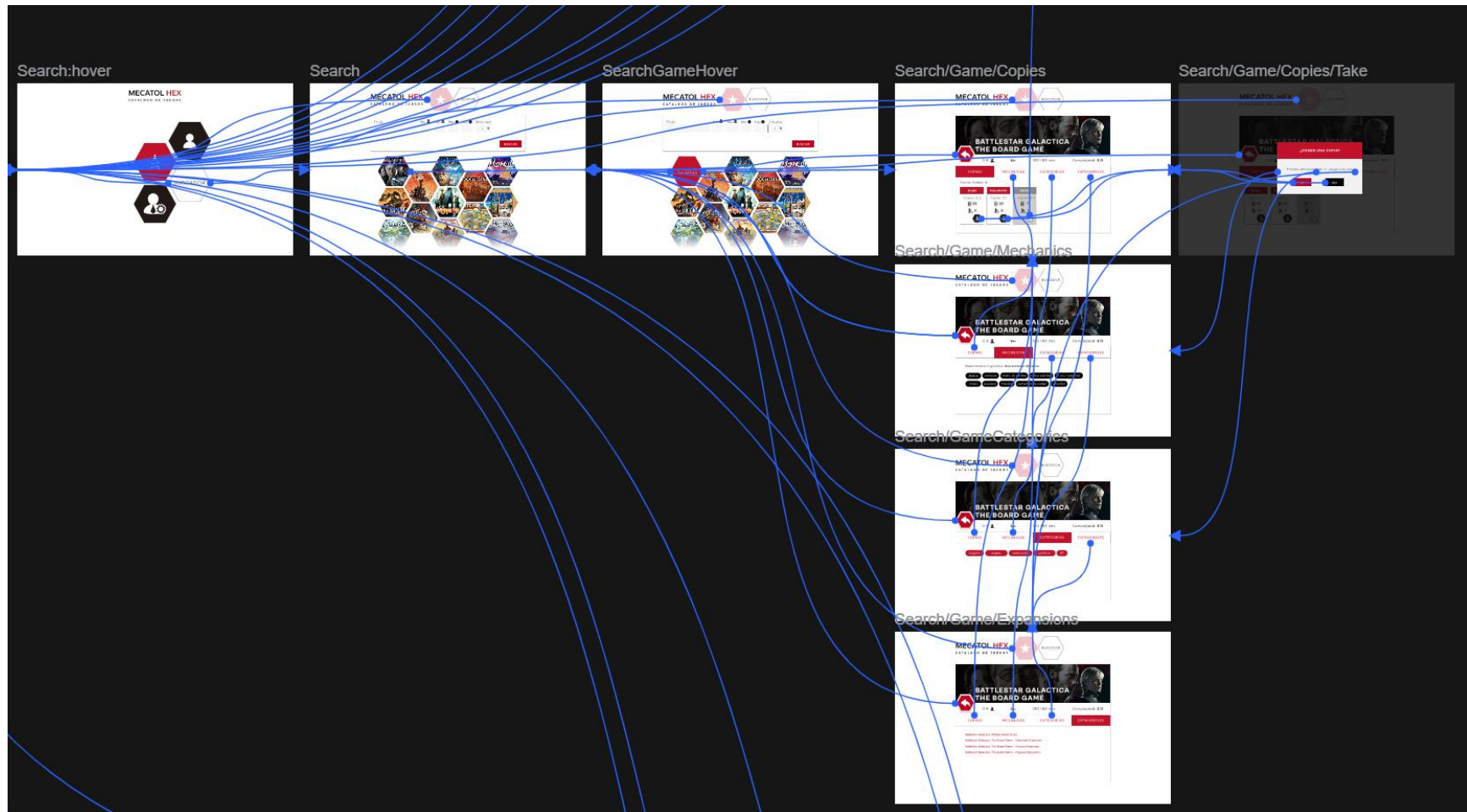
Zona Login/Registro



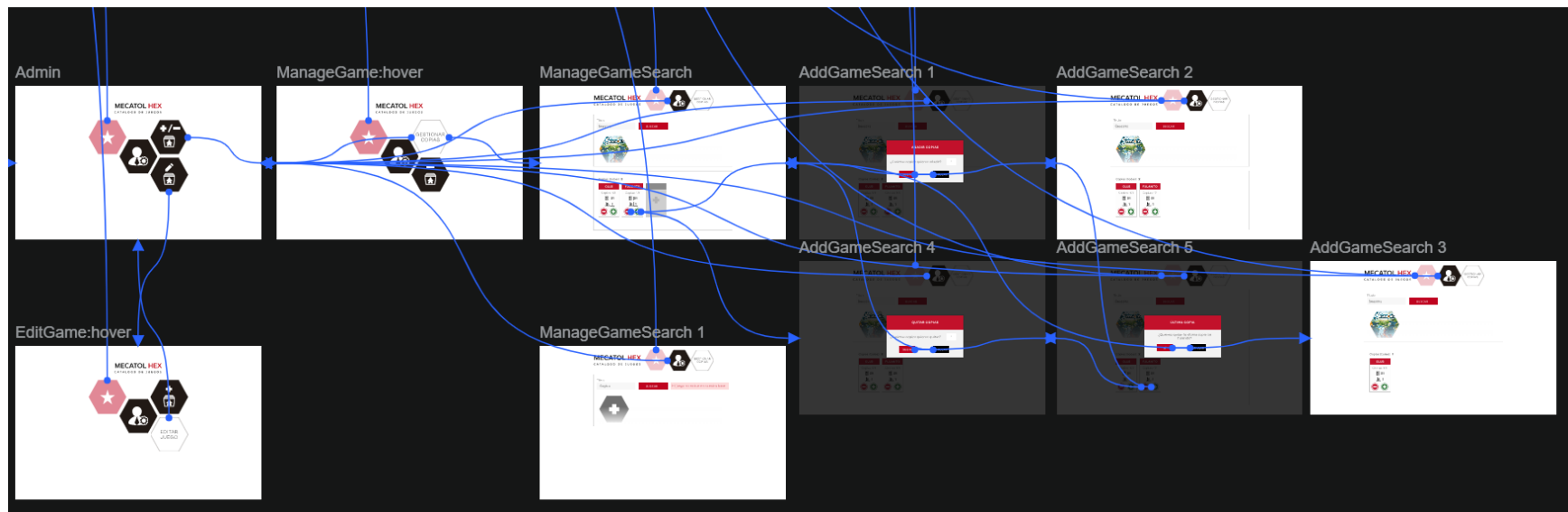
Zona Perfil



Zona búsqueda



Zona admin

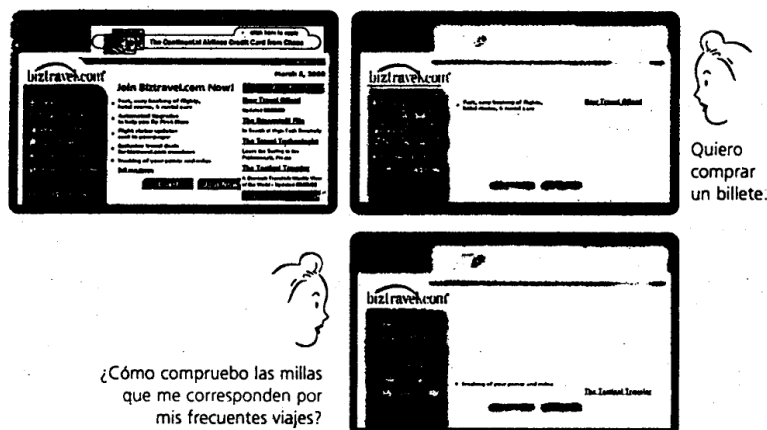


En este punto voy a profundizar el tema de las pautas de diseño de Steve Krug mencionadas anteriormente. El libro cumple ya 20 años y se nota que fue escrito en otra época por los comentarios que buenas páginas web deberían reproducir la experiencia típica de leer periódicos de gran formato. Aun así, muchas de observaciones allí siguen en vigor, lo cual se puede comprobar con [cualquier tutorial](#) ² de diseño UI de Web en Youtube.

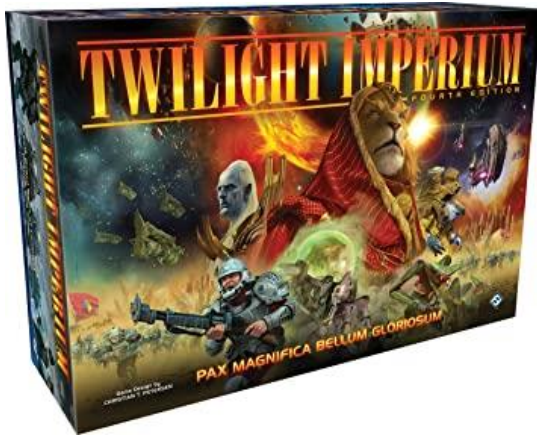
☉ Jerarquía visual clara

Krug menciona que *Uno de los pocos hechos bien documentados sobre el uso de la Web es que las personas tienden a invertir poco tiempo en leer la mayoría de páginas web* ³. Lo cual le lleva a la conclusión que la concentración del usuario es un recurso que hay que gestionar cuidadosamente. Allí también se marcan ciertas peculiaridades del mundo web - resulta que *tres clics mecánicos e inequívocos equivalen a un clic que requiere cierto grado de reflexión* ⁴ - es decir, los usuarios prefieren estructura de páginas más anidada en vez de menús enormes, porque realizar más acciones no cuesta casi nada de energía en el mundo físico. Eso se debe también a un fenómeno basado en funcionamiento del cerebro - solo somos capaces de procesar 4-5 cosas en mismo tiempo. ⁵ Por eso he decidido que mis páginas deben repartirse según acciones que quieren realizar usuarios y tener la estructura de filtro que lleva de lo más genérico a lo más concreto. Afortunadamente en este caso no tenía el factor de requisitos del cliente, lo cual bien describe [este video](#) gracioso.

LO QUE LOS DISEÑADORES CREAN... LO QUE LOS USUARIOS VEN...



- © Convenciones – La interfaz corresponde a lo que usuario reconoce de su ámbito
En este caso no podemos apostar que nuestros usuarios son ávidos lectores de periódicos, pero sabemos que suelen jugar a juegos de mesa. Entonces que mejor manera que utilizar esta referencia en nuestro menú.
Muchos juegos contienen losetas de hexágonos, incluso la que dio nombre al mismo club:



Por eso el diseño final por que he apostado es el siguiente:

MECATOL HEX
CATALOGO DE JUEGOS



- © Dejar bien claro cuáles son elementos con cuales se puede interaccionar
Esta característica mantenemos con hexágonos que responden al pasar el ratón por encima.

Las características anteriores explican también dos últimas:

- ☉ Evitar el ruido visual
- ☉ Omitir palabras innecesarias

Si el objetivo de la aplicación es servir como una herramienta de búsquedas rápidas no debería contener más que contenido que sirve para realizar esta tarea. Para acceder a los contenidos más elaborados, los usuarios tienen la página del club y el foro.

Análisis de páginas y usos:

Ingresar (index.jsp)

El usuario puede acceder utilizando su nombre de usuario o e-mail.

El usuario puede recuperar su contraseña

El usuario puede registrarse. Esta parte se va a modificar - el administrador debe controlar inscripciones.

The diagram shows the layout of the login page with red arrows indicating the user flow:

- An arrow from "El usuario puede acceder utilizando su nombre de usuario o e-mail." points to the "Usuario" label and the input field.
- An arrow from "El usuario puede recuperar su contraseña" points to the "¿Olvidaste la contraseña?" link.
- An arrow from "El usuario puede registrarse. Esta parte se va a modificar - el administrador debe controlar inscripciones." points to the "¿No tienes cuenta? Regístrate" link.

MECATOL HEX
CATALOGO DE JUEGOS

Ingresar

Usuario

Contraseña

Entrar

[¿Olvidaste la contraseña?](#)

[¿No tienes cuenta? Regístrate](#)


Cambio de contraseña (reset.jsp)

El usuario después de introducir su email recibirá a esa dirección una contraseña temporal. Se le avisará de necesidad de cambiar después de ingresar.

Se hace la comprobación si el email pertenece a un usuario registrado.



The diagram consists of a red line that starts from the text 'Se hace la comprobación si el email pertenece a un usuario registrado.' and ends with an arrow pointing to the 'Enviar' button.


MECATOL HEX
CATALOGO DE JUEGOS
Cambiar la contraseña

E-mail

Enviar

Volver

MECATOL HEX

CATALOGO DE JUEGOS

Inicio



Pantalla principal (home.jsp)

El usuario puede pasar a otras secciones. Nombre de la sección aparece al pasar el ratón encima de opción. Con este cambio se marca que es un elemento interactivo.

Se controla si el usuario tiene derechos de administrador. En caso de no tenerlos, no aparecerá la sección de admin.

MECATOL HEX

CATALOGO DE JUEGOS

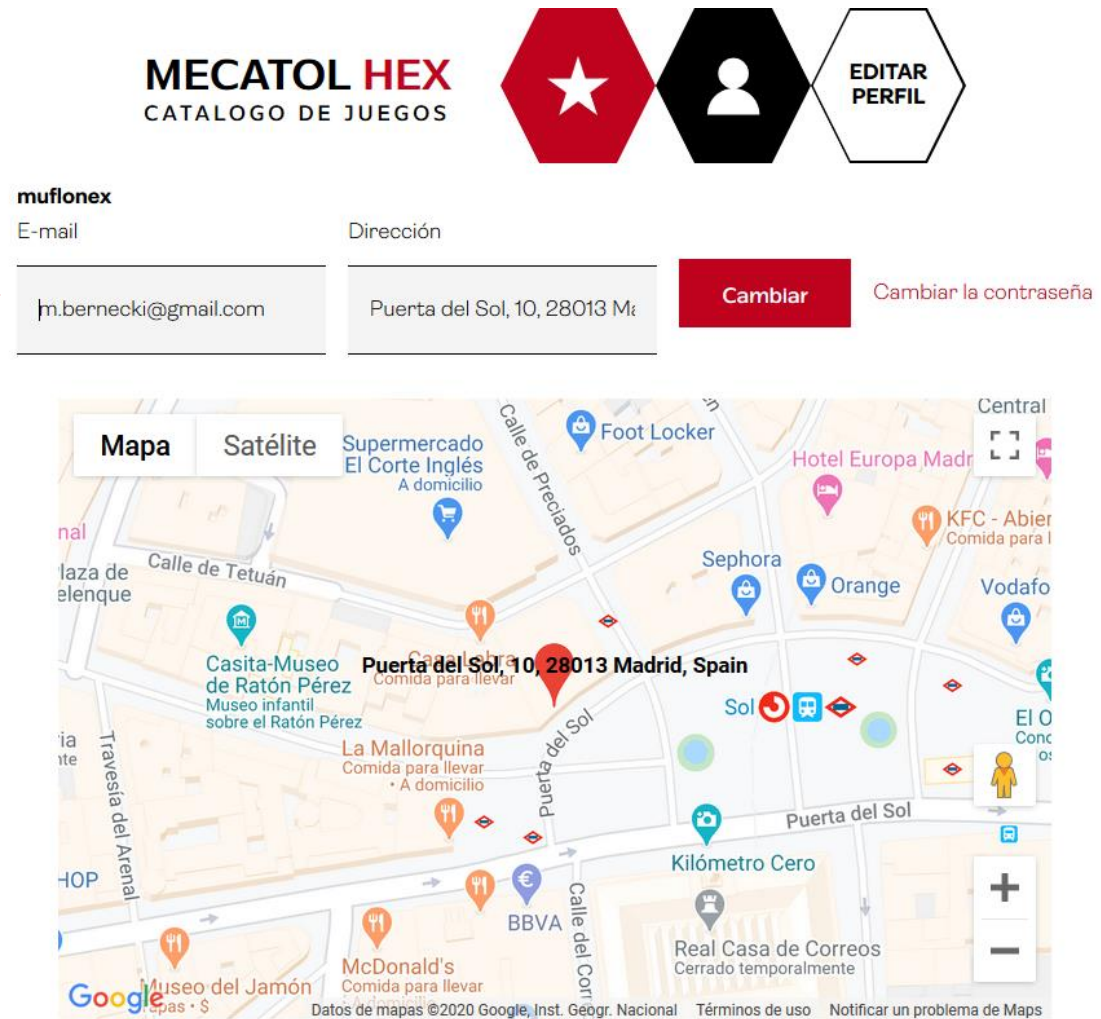
Inicio



El usuario puede cambiar su dirección física y de e-mail. Además, puede cambiar su contraseña.

Consideraciones para el futuro:

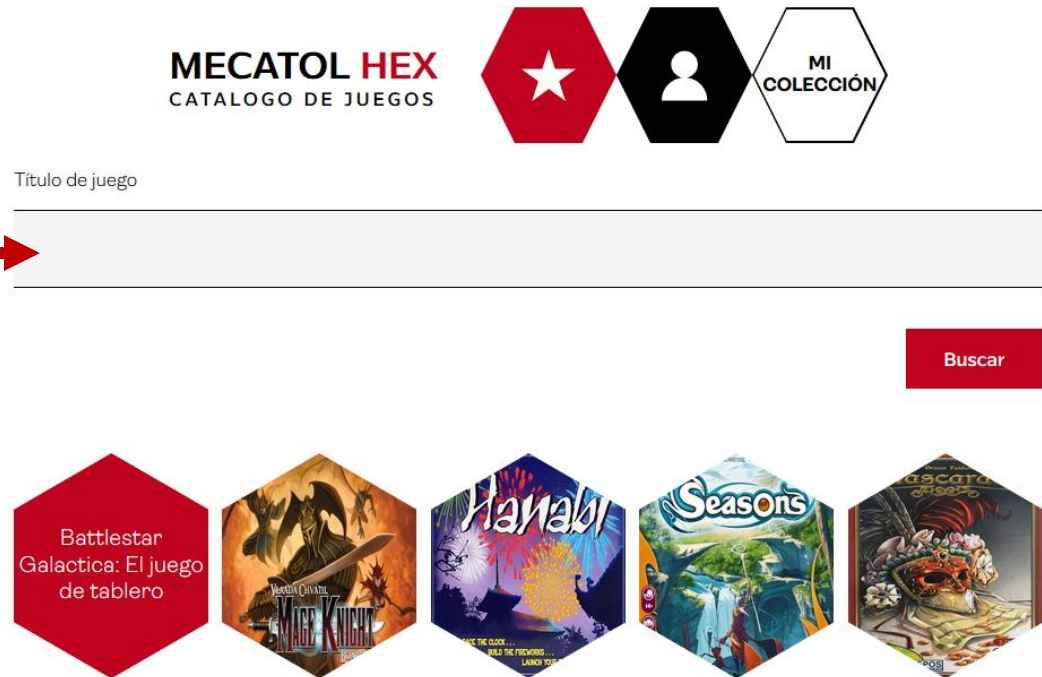
Usar las cuentas del foro en esta aplicación. Esto permitirá evitar problema de politica de protección de datos en versión en producción.



Colección del usuario (profile-collection.jsp)

Aquí el usuario puede filtrar los juegos que tiene prestados al club y pulsando en la loseta acceder a la información sobre su ubicación.

La consulta se realiza en la base utilizando opción `LIKE UPPER(:gameTitle)"` lo cual permite encontrarlo según partes de nombre.



Pantalla de préstamos del usuario (profile-loans.jsp)

Aquí el usuario puede controlar sus préstamos actuales y anteriores.





En el caso de tener un préstamo, este saldrá en primera posición sin fecha de fin, con opción de devolverlo.


Las demás copias prestadas anteriormente quedarán sombreadas en gris con fechas de inicio y fin.

En el caso de no tener una copia prestada en actualidad, los prestamos anteriores tendrán su botón de prestar activo.


En el caso que una copia está actualmente prestada por otro usuario, el botón de prestar de nuevo saldrá sombreado.

Prestando de nuevo se cambian fechas de préstamo anterior y “levanta”. Es una consideración de diseño de datos. Se valora más tener préstamos “favoritos” que mantener el historial de todos préstamos en la base.







Hanabi
Inicio: 15/06/2020 18:57
2
2
Devolver



Battlestar Galactica: Reglas opcionales
Inicio: 14/06/2020 23:21
1
Fin: 14/06/2020 23:41
1
Prestar de nuevo



Gloomhaven
Inicio: 15/06/2020 11:55
4
Fin: 15/06/2020 18:57
1
Prestar



Mage Knight
Inicio: 27/05/2020 20:11
3
Fin: 27/05/2020 20:45
4
Prestar de nuevo

Pantalla de búsqueda (search.jsp)

El usuario puede realizar la consulta conjunta, marcando los campos con cuales quiere filtrar. Pulsando en lo seta accederá a información sobre un juego.

En caso de elegir número de jugadores se van a seleccionar juegos que tienen por lo menos tantos jugadores.

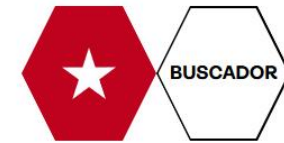
En caso de elegir el número máximo – los que tienen hasta tantos jugadores.

En caso de elegir el tiempo máximo – los juegos cuyas partidas suelen durar hasta tanto tiempo.

Consideraciones para el futuro:

Meter texto en opciones descriptivas de dificultad y nivel de dependencia lingüística en vez de valores numéricos.

MECATOL HEX
CATALOGO DE JUEGOS



Título	Min	Max	Buscar
<input type="text" value="Battlestar"/>	<input type="text" value=""/>	<input type="text" value=""/>	
Dependencia lingüística	Max	Complejidad	
<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	



Pantalla de detalles de juego (game.jsp)

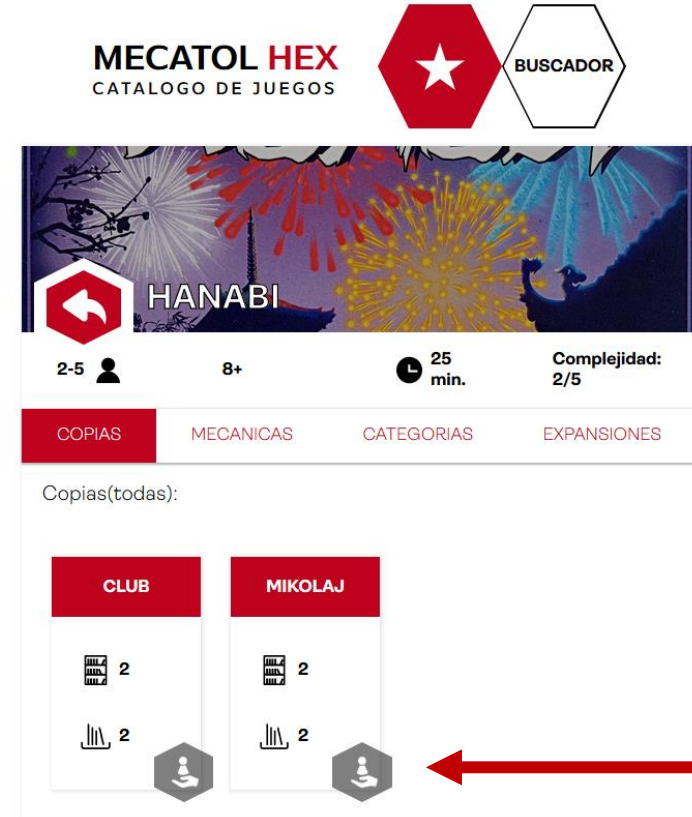
Aquí el usuario puede comprobar datos del juego. De momento solo la parte de información de copias está disponible.

En caso de tener un préstamo en curso las copias estarán marcadas como no disponibles con color gris.

El usuario puede prestar una copia de juego pulsando en el botón en su carta. Al pasar el ratón encima del botón se ampliará el botón y el color pasará a negro profundo.

Pulsando en botón de volver el usuario podrá volver a la página de búsqueda o a la página de su colección.

Consideraciones para el futuro: Completar la pantalla con otros datos (mecánicas, categorías y número total de copias).



Pantalla de editar colección de juegos [ADMIN] (admin-edit-game.jsp)

Esta página no ha sido creada durante el proyecto por motivo de problemas con la API. La idea para esta funcionalidad fue proporcionar un buscador de título de juego que realizaría una consulta contra la API de boardgamegeek (son dos APIs, una que permite buscar el ID de juego según su nombre y otra que permite buscar datos del juego según su ID).⁶

Desafortunadamente la api está hecha con XML sin WSDL (Web Services Description Language que permite utilizar opciones de creaciones de clases responsables de conexión a través de un wizard de Eclipse), con lo cual su introducción significaría mucho más esfuerzo relacionado con crear conexiones de manera manual y parsear datos.

El propósito de la página fue permitir editar datos del juego en si.

El administrador podrá buscar un título de juego.

En caso de haber juegos con nombres parecidos en la base, saldrá una lista de losetas-enlaces a formulario de editar.

MECATOL HEX
CATALOGO DE JUEGOS

Título
Caylus

BUSCAR

El juego no existe en nuestra base

EDITAR JUEGO

En caso de no haber ninguno que corresponde a esta búsqueda saldrá un aviso.

El administrador podrá también pulsar la loseta gris para añadir nuevo registro, en este caso, se realizará la búsqueda en la API de Boardgamegeek, rellenando campos del formulario

Pantalla de editar colección de juegos [ADMIN] (admin-manage-copies.jsp)




Esta pantalla no se realizó durante el proyecto por falta de tiempo y por no ajustarse al modelo de datos en la aplicación. En esta pantalla el administrador puede añadir o quitar copias del juego.

Se realiza la búsqueda del título de juego y según el resultado salen las copias relacionadas con él.

El administrador podrá cambiar el sitio de la copia utilizando los campos con estos datos en tarjetas de copias.

El administrador podrá crear nueva copia pulsando en el rectángulo gris.


MECATOL **HEX**
CATALOGO DE JUEGOS



Título

Seasons


BUSCAR






Copias (todas): 2

CLUB

Copias: 1/1


 21


 1






FULANITO

Copias: 1/1

 21

 1





Construcción de lógica de backend: Base de datos

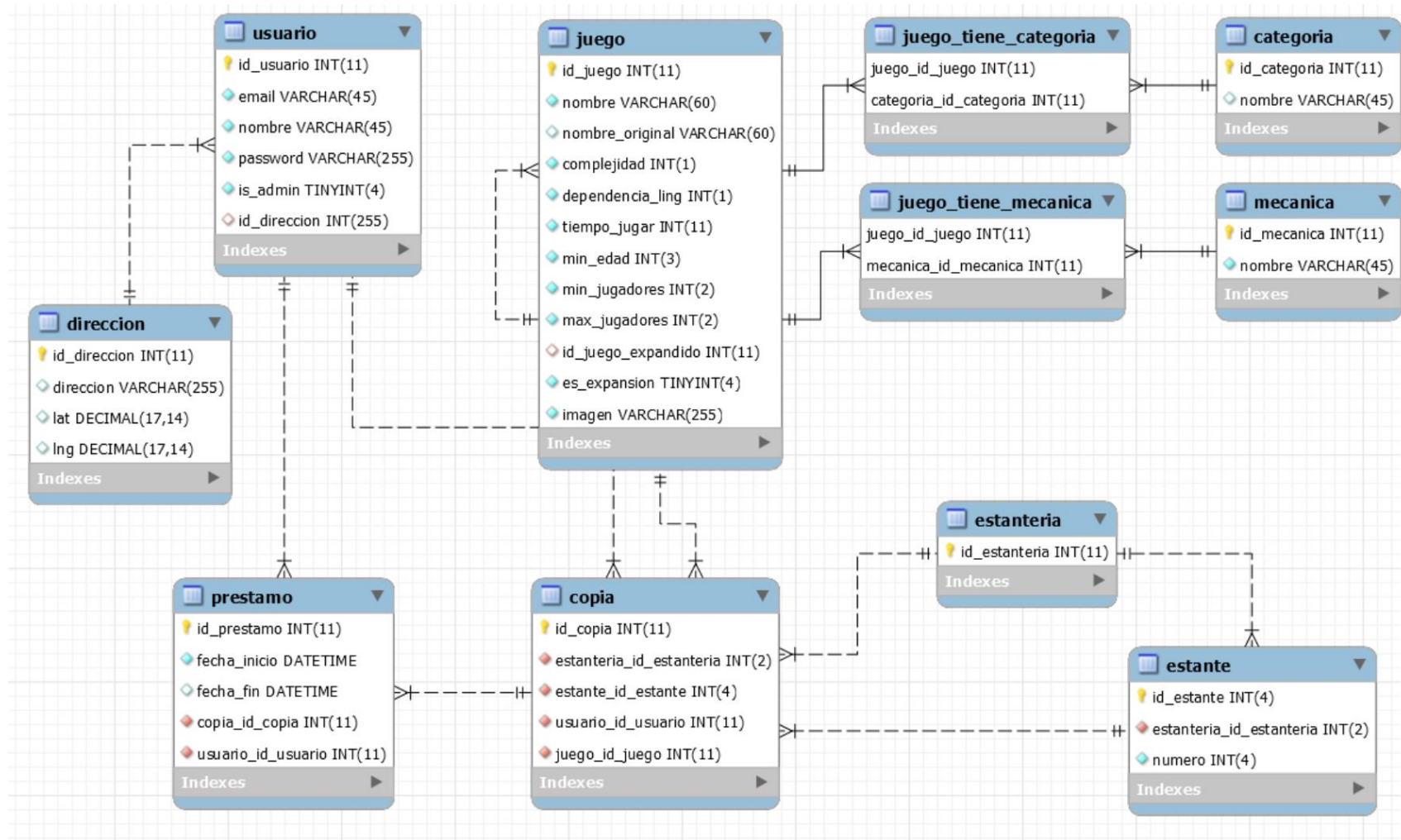
Relaciones que tenemos en nuestro proyecto

Relación	Cardinalidad
Usuario tiene Copias (de juegos)	1 - N
Usuario tiene Prestamos	1 - N
Usuario tiene Dirección	1 - 1
Juego tiene Copias	1 - N
Copia está presente en Prestamos	1 - N
Juegos tienen Categorías	N - N
Juegos tienen Mecánicas	N - N
Juego tiene Expansiones (Juegos)	1 - N
Estantería tiene Estantes	1 - N
Estantería almacena Copias	1 - N
Estante almacena Copias	1 - N

Con este diseño queremos cumplir con las formas normales que quieren evitar las anomalías de registros que podrían llevar dependencias internas entre las propiedades. Por eso extraemos:

- ☉ tablas renacidas de Mecánicas y Categorías.
- ☉ tabla de Dirección para poder marcarla con lng y lat geográfica en mapas Google
- ☉ Tabla de Estante de Estantería porque una Estantería tiene muchos estantes, y por tanto, Estantería tampoco puede ser una propiedad directa de Juegos.

Diagrama de base de datos



Realización de la capa de interfaz y servicios

Se ha apostado por tecnologías de Java.

Nuestro código sigue el patrón Modelo-Vista-Controlador. Dicho patrón corresponde al paradigma de separación de intereses por cual apuestan desarrolladores de movimiento de Código Limpio que – entre varias pautas – consiste en separar partes de código en función de su intención para facilitar su lectura.

Modelo de datos reside en Beans de Java, que son clases que describen entidades. Esas clases llevan anotaciones de ORM (modelación objeto-relacional) que permiten traducir datos de base al mundo de programación orientada a objetos.

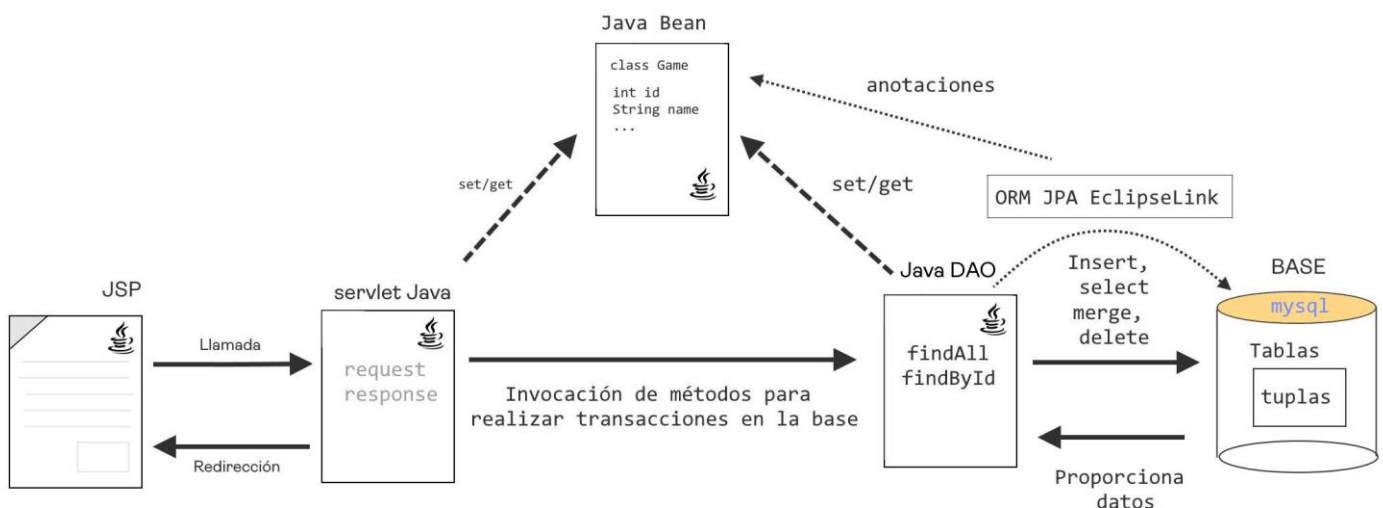
Las **Vistas** vienen de parte de hojas de JSP con JSTL que corresponden a partes marcadas en la fase de diseño.

El **Controlador** está representado por servlets que se encargan de gestionar llamadas y respuestas al cliente. Estas clases se reparten en función un servlet por un caso de uso, la idea de eso es no mantener el estado de datos dentro del controlador.

Abajo se muestra el esquema de comunicación de clases que se responsabilizan del manejo de datos.

- admin.jsp
- admin-edit-game.jsp
- admin-manage-copies.jsp
- game.jsp
- home.jsp
- index.jsp
- profile.jsp
- profile-collection.jsp
- profile-edit.jsp
- profile-loans.jsp
- profile-password-change.jsp
- register.jsp
- reset.jsp
- search.jsp

- controller
 - Admin.java
 - Login.java
 - ProfileCollection.java
 - ProfileDetailsUpdate.java
 - ProfileLoans.java
 - ProfilePasswordChange.java
 - ResetPassword.java
 - Search.java
 - SingleGame.java
 - UserRegister.java



Se observaban siguientes las reglas de programación:

- © Keep it Simple Stupid – por eso no he creado un controlador principal que requería meter más reglas IF en caso de extender el proyecto, en vez de eso, servlets se invocan según su caso de uso.
- © Don't Repeat Yourself – Abstraer invocaciones de métodos. Uno de casos interesantes, invocación de métodos de Bean desde Servlet de préstamos a través de reflexión de método⁷:

```
if( actionDataPresent) {  
    int loanId = Integer.parseInt(request.getParameter("id"));  
    Loan loan = loanDao.findById(loanId);  
    try {  
        method = loan.getClass().getMethod(action);  
    } catch (SecurityException e) {  
        request.setAttribute("message", e);  
    } catch (NoSuchMethodException e) {  
        request.setAttribute("message", e);  
    }  
  
    try {  
        method.invoke(loan);  
        loanDao.update(loan);  
        String message = "Operación realizada con éxito.";  
        request.setAttribute("message", message);  
    } catch (IllegalAccessException e) {  
        request.setAttribute("message", e);  
    } catch (InvocationTargetException e) {  
        request.setAttribute("message", e);  
    }  
}
```

De este modo podemos invocar métodos de Bean de modo menos explícito. Se podría extender con parámetros, si fuese necesario.

- © Sustituir “valores mágicos” por variables bien nombradas
- © Inicializar variables de uso concreto más cerca posible de su invocación.
- © No utilizar abreviaturas para nombres de variables salvo que sean de uso común (por ejemplo, **em** para **EntityManager**, pero **searchedGameTitle**).
- © Evitar posibles inyecciones SQL con control de parámetros a través de Predicatos de Criteria Query API (en clase GameDaoImpl).

Conclusiones/Resultado final del proyecto

Ha sido un camino largo y bastante abrumador para un equipo de sola persona. Sobre todo, cuando me atascaba con dilemas de Java, como por ejemplo tipo de dato dentro de HashMap de parámetros de un request (al parecer, valores en este mapa son arrays porque un parámetro puede aparecer más que una vez ⁸). Uno de errores principales fue no consultarme con el profesorado por asumir que son cosas que debería saber. Reconozco también que quizás debería investigar mejor mi posible API de proporcionar datos para evitar las complicaciones. Enfocarme en datos que lleva aquella base me quitó de vista la posibilidad de evolucionar el proyecto en cuando estén creadas las funcionalidades principales.

Consideraciones para el futuro fueron marcadas en cada parte respectiva. Pasos siguientes serían reforzar la seguridad o enlazarla con la del foro de club, acabar las secciones de administrador, meter la conexión con las APIs de boardgamegeek desde Front, luego mudar el proyecto a servicios Java con endpoints JSON y crear vistas en uno de frameworks/librerías de Front para agilizar la renderización y poder aprovechar del patrón SPA, que se ajusta mucho a la idea de la página.

Bibliografía

- 1** Krug S., No me hagas pensar, 2000,
https://www.disenomovil.mobi/multimedia_un/01_intro_ux/no_me_hagas_pensar_steve%20_krug_2da%20ed.pdf
- 2** The 2019 UI Design Crash Course for Beginners: https://www.youtube.com/watch?v=_Hp_dI0DzY4
- 3** Krug, 22
- 4** Krug, 41
- 5** Oakley, B, Learning How to Learn, <https://youtu.be/vd2dtkMINlw?t=1760>
- 6** Liss, Taylor A., BGG API Documentation - <http://www.tayloralliss.com/bggapi/introduction.html>
- 7** Henrik P., StackOverflow hilo 160970, Cómo invocar un método según su nombre:
<https://stackoverflow.com/questions/160970/how-do-i-invoke-a-java-method-when-given-the-method-name-as-a-string>
- 8** StackOverflow, hilo 27732133 : HttpServletRequest getParameterMap() vs getParameterNames
<https://stackoverflow.com/questions/27732133/httpServletRequest-getparametermap-vs-getparameternames>
 - ⊙ BoardgameGeek API Documentation: https://boardgamegeek.com/wiki/page/BGG_XML_API2
 - ⊙ [Java Persistence Mini Book – Herbert Coelho, Byron Kiyourtzoglu](#)
 - ⊙ Java Persistence Relationships: https://en.wikibooks.org/wiki/Java_Persistence/Relationships
 - ⊙ How to implement Forgot your Password Feature: <https://www.codejava.net/coding/how-to-implement-forgot-password-feature-for-java-web-application>
 - ⊙ JAVA-Send URL HTTP Request and Read XML Response
<https://www.youtube.com/watch?v=F3hWv6PimB8>
 - ⊙ JPA Self-referencing Relationships https://www.youtube.com/watch?v=GV2tA3_uKBE&t=565s
 - ⊙ JSTL Standard Tag Library https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm
 - ⊙ Documentación Criteria API <https://www.objectdb.com/java/jpa/query/criteria>
 - ⊙ StackOverflow hilo 4014390 - JPA/Criteria API - Like & equal problema:
<https://stackoverflow.com/questions/4014390/jpa-criteria-api-like-equal-problem>

-
- ◎ StackOverflow hilo 19815145: JAX-WS client without a WSDL document file:
<https://stackoverflow.com/questions/19815145/jax-ws-client-without-a-wsdl-document-file>
 - ◎ StackOverflow hilo 12579880: Java: Can we use DAO as a Singleton instance:
<https://stackoverflow.com/questions/12579880/java-can-we-use-dao-as-a-singleton-instance>
 - ◎ StackOverflow hilo 137975: What is so bad about singletons? [closed]:
<https://stackoverflow.com/questions/137975/what-is-so-bad-about-singletons>
 - ◎ StackOverflow hilo 272118: One or multiple servlets per webapp:
<https://stackoverflow.com/questions/272118/one-or-multiple-servlets-per-webapp>
 - ◎ StackOverflow hilo 8260881: What is the most elegant way to check if all values in a boolean array are true?
<https://stackoverflow.com/questions/8260881/what-is-the-most-elegant-way-to-check-if-all-values-in-a-boolean-array-are-true>
 - ◎ StackOverflow hilo 2860943: How can I hash a password in Java?
<https://stackoverflow.com/questions/2860943/how-can-i-hash-a-password-in-java>