

2014-  
2015

# Treivial



Arenal Pereira Adrián

Cabañeros Blanco José Antonio

Castañón Muñiz Borja

Díaz García Jaime

García García Marcos

Ordóñez González Carlos

Valdés CuervoAmable José

Arquitectura del software



## INDICE

PLANTEAMIENTO DEL PROBLEMA .....	3
Primera etapa .....	4
Segunda etapa .....	4
Tercera etapa .....	4
Cuarta etapa .....	4
METODOLOGIA USADA .....	4
STAKEHOLDERS Y ATRIBUTOS DE CALIDAD .....	5
1- Identificación de los interesados .....	5
1- UsuarioJugador .....	5
2- Equipo de proyecto .....	5
3- Gerente del proyecto .....	5
4- Desarrolladorescontratados .....	6
5- Técnico .....	6
6- Operadores de la empresa .....	6
2- Identificación de los atributos de calidad .....	6
1- Disponibilidad: .....	6
2- Modificabilidad: .....	6
3- Rendimiento: .....	6
4- Seguridad: .....	6
5- Testabilidad: .....	7
6- Depurabilidad: .....	7
7- Mantenibilidad: .....	7
3- Lista actualizada de los interesados .....	7
1- Lista final de los stakeholders: .....	7
2- Lista de atributos de calidad: .....	8
3- Atributos de calidad e interesados: .....	9
DESCRIPCION DE NEGOCIO DE LA SOLUCION .....	9
Solución .....	9
Control Central .....	10
Pantallas .....	10
ESCENARIOS DE CALIDAD .....	11

ARQUITECTURA Y CORRESPONDENCIA CON IMPLEMENTACION .....	12
DIAGRAMAS Y VISTAS .....	15
Diagrama de clases 1 .....	16
Resumen diagrama de clases 1 .....	16
Diagrama de clases 2 .....	18
Resumen diagrama de clases 2 .....	18
Diagrama de clases 3 .....	19
Resumen diagrama de clases 3 .....	20
Diagrama de clases 4 .....	21
Resumen diagrama de clases 4 .....	21
Diagrama entidad relacional usuarios.....	22
Resumen diagrama entidad relacion.....	23
Diagrama interacción ventanas .....	23
Resumen diagrama interacción de ventanas.....	23
MANUAL DE USUARIO .....	24
Paso 1 – Descargar el framework PLAY.....	24
Paso 2 – Realizar los ajustes necesarios en variables de entorno .....	24
Paso 3 – Arrancamos el servidor.....	25
Paso 4 – arrancamos base de datos.....	26
Paso 5 – La aplicación .....	27
OTROS DATOS DE INTERES .....	29

## PLANTEAMIENTO DEL PROBLEMA

La empresa NoGame desea desarrollar diferentes juegos del tipo preguntas/respuestas disponibles en diferentes plataformas.

Nos encontramos en la tercera fase del desarrollo que consiste en la creación de la propia aplicación del juego del Trivial a nivel web. Recogiendo las preguntas en el formato .GIFT y presentando una interfaz gráfica a nivel web para jugar al trivial. Además incorporando un sistema de registro y control de usuarios que nos permita manejar una gran cantidad a la vez.

Además se ha incorporado un chat de manera adicional para ofrecer la posibilidad de poder hablar con los demás jugadores que estén en la partida.

Se desarrolla esta aplicación en cuatro etapas:

## Primera etapa

Se creará una base de datos relacionada con la gestión de usuarios, de partidas y todo lo relacionado con la estadística de partida de los jugadores. Con ella podremos facilitar información al administrador de la aplicación, además de hacer un seguimiento de los usuarios.

Esta base de datos no ha sufrido ningún tipo de cambio con respecto a la aplicación de la versión web. Gracias a esto se podrá mantener la base de datos para jugar desde el escritorio o la web con un mismo usuario y que se guarden los resultados en la misma base de datos.

## Segunda etapa

Desarrollo de la interfaz gráfica de la aplicación. En la cual diseñamos todas las ventanas pertinentes al registro, menú principal, el tablero, registro, etc. Para ello, se hará uso de HTML para su representación, y mediante el Framework de PLAY nos apoyaremos para darle la lógica que queramos a cada ventana, de forma separada.

La lógica de la aplicación de escritorio se mantendrá en esta versión, al menos parte de ella, gracias a la separación de la anterior parte entre lógica e interfaz.

## Tercera etapa

Incorporar la aplicación a toda la base de preguntas para la posibilidad de mostrarlas en nuestra aplicación, tal y como hacíamos en nuestra aplicación de escritorio.

## Cuarta etapa

Implementación de un sistema de chat con el que poder comunicarse entre usuario. A través del uso de *javascript* y *scala*.

## METODOLOGIA USADA

Se va a realizar un estudio de la arquitectura siguiendo el modelo **Vista-Controlador** para separar las posibles vistas del sistema del controlador y del propio modelo. Pudiendo desarrollar independientemente la interfaz gráfica del juego del resto de la aplicación.

Además se dará la posibilidad de la creación de diferentes vistas para el usuario. Gracias a que será independiente del resto de aplicación. Por ejemplo, mediante los *look and feel* de la tecnología usada, podremos presentar nuestra aplicación de forma diferente sin necesidad de cambiar nada.

La extracción de datos para las preguntas se llevará a cabo a través de un proceso **ETL**. El cual se encarga primero de extraer las propias preguntas, transformarlas y cargarlas en nuestra base de datos.

Para integrar la base de datos se ha utilizado un patrón **DAO**, para las posibles diferentes implementaciones que pudiéramos querer utilizar. De esta forma, suministraremos un punto de conexión entre la interfaz de la aplicación las bases de datos que tengamos, en este caso una, consiguiendo así separar las responsabilidades.

## STAKEHOLDERS Y ATRIBUTOS DE CALIDAD

### 1- Identificación de los interesados

#### 1- UsuarioJugador

Se trata de cualquier persona que tenga la aplicación y disponga del tiempo libre suficiente para jugar una partida al Trivial

Entre sus objetivos están:

- Disponibilidad de la aplicación en cualquier momento.
- El juego cumpla sus reglas y la contabilidad de la puntuación sea correcta.
- El rendimiento de la aplicación sea óptimo, teniendo un transcurso de la misma con fluidez y sin parones.

#### 2- Equipo de proyecto

Se trata de la agrupación que ha decidido llevar adelante la idea de la creación del sistema y se encargará de contratar al personal adecuado para que la idea tenga éxito.

Entre sus objetivos están:

- Conseguir el máximo beneficio con la aplicación.
- Ganar prestigio en nombre de la empresa por el éxito de la aplicación → Popularidad.

#### 3- Gerente del proyecto

Se encarga de todos los aspectos financieros de la empresa. Entre ellos están los presupuestos y las tomas de decisiones que comprometen los fondos de dichos presupuestos.

Entre sus objetivos están:

- Bajo coste de desarrollo, esto es que el desarrollo del proyecto debe ser lo más corto y con el coste lo más reducido posible.
- Conseguir el funcionamiento esperado sin utilizar excesos en tecnología.
- Conseguir con el menor coste posible el mayor beneficio para los componentes del proyecto.

#### 4- Desarrolladores contratados

Este grupo de personas serán los responsables de desarrollar el sistema resultante de la arquitectura.

Entre sus objetivos están:

- Seguridad de acceso a los datos, no se puede acceder si no se tienen los permisos definidos en las correspondientes políticas de acceso.
- Comunicación ágil
- El sistema deberá ser fácil de mantener y de escalar, ya que la implementación de otros juegos dentro del sistema se está barajando.

#### 5- Técnico

Es el encargado de subsanar cualquier posible fallo que ocurra a nivel de hardware (o software) dentro del sistema informático que soporta la aplicación.

Entre sus objetivos están:

- El fallo sea fácil de localizar.
- El fallo sea fácil de subsanar ocupando el menor tiempo posible.

#### 6- Operadores de la empresa

Son el grupo de personas contratado por la empresa que se encargarán de realizar tareas mecánicas y repetitivas dentro de la empresa día a día.

Entre sus objetivos están:

- Simplicidad de su tarea lo máxima posible.
- En caso de un error, poder volver al estado correcto lo antes posible.

## 2- Identificación de los atributos de calidad

### 1- Disponibilidad:

- La disponibilidad debe ser 24x7.

### 2- Modificabilidad:

- Facilidad para elegir más de un formato en el que se pueden extraer las preguntas de los ficheros entrada de los contenedores.
- Escalabilidad del sistema, a la hora de ampliar más formatos de texto que acepte el sistema para suministrar las preguntas.

### 3- Rendimiento:

- La extracción de preguntas de los ficheros en la base de datos debe ser fluida.

### 4- Seguridad:

- Garantizar que el acceso de la base de datos donde almacena las preguntas estará solo autorizado al personal que lo mantenga (operadores del sistema) o administradores.

### 5- Testabilidad:

- Facilidad para probar que tanto el salvado como la recuperación de preguntas de la base de datos sucede satisfactoriamente.

### 6- Depurabilidad:

- Facilitar la labor de localización de fallos aplicando una arquitectura que modularice las funciones.

### 7- Mantenibilidad:

- Garantizar que la restitución o sostenibilidad del sistema vendrá acompañado de un esfuerzo lo menor posible.
- Tolerancia a fallos:
- Cumplir el intervalo de fallos que se especifica que el sistema puede provocar. Nuncasobrepasandoesatasa.

## 3- Lista actualizada de los interesados

### 1- Lista final de los stakeholders:

Código	Stakeholder	Intereses
ST-01	Usuario Jugador	Disponibilidad de la aplicación en cualquier momento. El juego cumpla sus reglas y la contabilidad de la puntuación sea correcta. El rendimiento de la aplicación sea óptimo, teniendo un transcurso de la misma con fluidez y sin parones.
ST-02	Equipo de proyecto	Conseguir el máximo beneficio con la aplicación. Ganar prestigio en nombre de la empresa por el éxito de la aplicación->Popularidad.
ST-03	Gerente de proyecto	Bajo coste de desarrollo, esto es que el desarrollo del proyecto debe ser lo más corto y con el coste lo más reducido posible. Conseguir el funcionamiento esperado sin utilizar excesos en tecnología. Conseguir con el menor coste posible el mayor beneficio para los componentes del proyecto.
ST-04	Desarrolladores contratados	Seguridad de acceso a los datos, no se puede acceder si no se tienen los permisos definidos en las correspondientes políticas de acceso. Comunicación ágil El sistema deberá ser fácil de mantener y de escalar, ya que la implementación de otros juegos dentro del sistema se está barajando.
ST-05	Técnico	El fallo sea fácil de localizar. El fallo sea fácil de subsanar ocupando el menor tiempo posible.
ST-06	Operadores del sistema	Simplicidad de su tarea lo máxima posible. En caso de un error, poder volver al estado correcto lo antes posible.

## 2- Lista de atributos de calidad:

Código	Descripción	Tipo de Atributo
AT001	La disponibilidad debe ser 24x7	Disponibilidad
AT002	Facilidad para elegir más de un formato en el que se pueden extraer las preguntas de los ficheros entrada de los contenedores.	Modificabilidad
AT003	Escalabilidad del sistema, ya que se podría ampliar el sistema añadiéndole más juegos del mismo estilo en el futuro.	Modificabilidad
AT004	Escalabilidad del sistema, a la hora de ampliar más formatos de texto que acepte el sistema para suministrar las preguntas.	Modificabilidad
AT005	La extracción de preguntas de los ficheros en la base de datos debe ser fluida.	Rendimiento
AT006	Garantizar que el acceso de la base de datos donde almacena las preguntas estará solo autorizado al personal que lo mantenga (operadores del sistema) o administradores.	Seguridad
AT007	Facilidad para probar que tanto el salvado como la recuperación de preguntas de la base de datos sucede satisfactoriamente.	Testabilidad
AT008	Facilitar la labor de localización de fallos aplicando una arquitectura que modularice las funciones.	Depurabilidad
AT009	Garantizar que la restitución o sostenibilidad del sistema vendrá acompañado de un esfuerzo lo menor posible.	Mantenibilidad
AT010	Cumplir el intervalo de fallos que se especifica que el sistema puede provocar. Nuncasobrepasandoesatasa.	Tolerancia a fallos



### 3- Atributos de calidad e interesados:

Los diferentes atributos de calidad son de interés para alguno de los Stakeholders. La siguiente tabla muestra la lista de intereses para el proyecto actual:

Atributos Vs Interesados	ST-01	ST-02	ST-03	ST-04	ST-05	ST-06
AT001	X	X	X	X	X	X
AT002		X		X		
AT003		X	X	X		
AT004		X		X		
AT005	X	X		X		
AT006		X		X		X
AT007		X		X	X	X
AT008		X		X	X	
AT009		X	X	X	X	
AT010		X	X	X	X	

## DESCRIPCION DE NEGOCIO DE LA SOLUCION

El modelo de negocio está basado en una página web que nos aporta la funcionalidad completa de nuestra aplicación y a la que el usuario podrá acceder y jugar desde su PC.

La aplicación se conectará a la base de datos principal, donde recuperará datos para por ejemplo registrarse o conectarse como un usuario dado de alta.

### Solución

- Control central:
  - Gestión de usuarios de la aplicación.
  - Recoge preguntas a cargar para el tablero de la operación, por categorías.
  - Recoge las respuestas del usuario por pregunta para almacenarlas.
- Pantallas
  - Presentación de cada interfaz del sistema y su gestión
  - *Look and feel* de la aplicación

## Control Central

Cuando el usuario se intenta conectar por primera vez a la aplicación tendrá que pasar antes por el registro de esta, introduciendo sus datos, y guardándose estos en la base de datos de los usuarios. Una vez hecho esto ya podrá entrar en el menú principal a través del login, comprobándose en la base de datos si existe o no.

Una vez empezamos a jugar, se cargaran las preguntas de todas las categorías de los ficheros .GIFT correspondientes, y estos se guardaran en nuestra aplicación de cara a mostrarlos de manera más eficiente durante el transcurso de ejecución.

Una vez terminada una partida, se guardará en la base de datos los resultados de este jugador.

## Pantallas

Presenta cada pantalla a través de html, con un *look and feel* común (con el uso *bootstrap* y los CSS generados).

Los usuarios en sesión podrán navegar con total libertad por la aplicación, pudiendo elegir a que pantallas desea ir. También tendrá la posibilidad de cerrar la sesión actual, llevando al usuario una vez elegida dicha opción al menú inicial de login.

También se presentará una sala de chat con la que comunicarse entre usuarios de la misma aplicación, mediante javascript.

## ESCENARIOS DE CALIDAD

Escenario Nº	Fuente de estímulo	Estímulo	Entorno	Artefacto	Respuesta	Medición de la respuesta	Atributos de calidad
1	Mantener el sistema siempre operativo	No perder dinero por una mala funcionalidad	Sistema	Equipoinformático	Atención 24x7	Horas desactivado = 0	AToo1
2	Diversidad de formatos	Variedad a la hora de obtener preguntas	Sistema de lectura de ficheros	Equipoinformático	Transformación a únicoformato	Ha de aceptar GIFT como mínimo	AToo2
3	Ampliación del sistema	Posibleaumento del negocio	Sistema y desarrollo	Equipo de desarrollo	Facilidad a la hora de añadir funciones	Tiempo de desarrollo< 1 mes	AToo3
4	Ampliación de formatos	Posiblesnuevosformatos	Sistema	Equipoinformático	Facilidad a la hora de añadir nuevos tipos de formatos	Tiempo de desarrollo< 1 mes	AToo4
5	Operaciones claves de la BBDD	Transmisión de datos	BBDD	Equipoinformático	Optimización del algoritmo de obtención de preguntas	Tiempo de extracción por pregunta < 15 segundos	AToo5
6	Seguridad del sistema	Imposibilidad de usuarios sin permisos	Sistema y empresa	Sistema	Accesorestringido	Horas con seguridaddesactivada = 0	AToo6
7	Funcionamientosatisfactorio	Impedirposiblesfallos	Sistema de lectura de ficheros	Sistema	Testeo de la aplicación por desarrolladores y tests JUnit	Tasa de aciertos > 95%, solo errores al encontrar caracteres extraños	AToo7
8	Fácillocalización de errores	Optimizar el tiempo de solución de errores	Sistema	Equipoinformático	Usar herramientas de localización de errores y encapsulamiento	Coste de herramientas = 0 € y tiempo de localización < 1 hora	AToo8
9	Sistema fácilmentemodificable	Facilitar trabajo de los desarrolladores ante futuros cambios	Equipo de desarrollo	Desarrollo	Uso de patrones y algoritmos óptimos	Tiempo de desarrollo< 3 mes	AToo9
10	Acción en el sistema incorrecta	Errores	Sistema	Equipoinformático	Análisis y solución de errores	Tiempo en generar la solución < 1 semana	AToo10

## ARQUITECTURA Y CORRESPONDENCIA CON IMPLEMENTACION

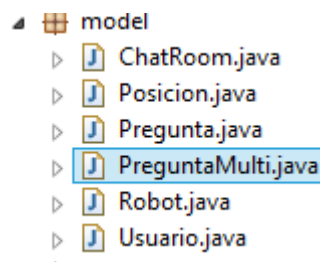
Tanto para la aplicación web como para la aplicación de escritorio se ha seguido una arquitectura MVC.

En ambos casos era necesario dar flexibilidad al usuario a la hora de elegir y personalizar el tablero y para ello se tomó esa decisión.

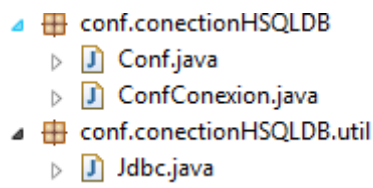
El MVC nos aporta esa separación entre lógica, modelo y vista de manera que podemos ofrecer al usuario lo anteriormente mencionado.

Se inició con una arquitectura en la aplicación de escritorio que luego se reutilizó y perfeccionó en la parte web. Esta arquitectura será la que se muestre a continuación.

Como se puede ver en la siguiente imagen, nuestra aplicación dispone de un paquete para el modelo donde se han metido las clases que conviven en el sistema. En esta capa manejaríamos todo lo que se refiere a la información que luego será transferida a la vista.

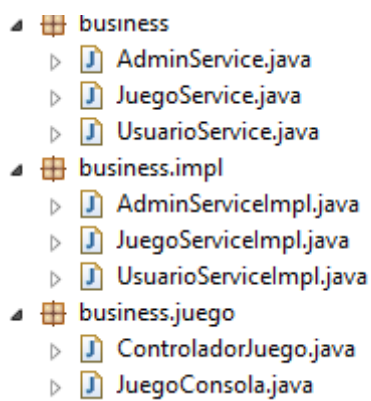


Se ha tomado la decisión de usar el patrón DAO para separar el acceso a la base de datos del modelo con lo cual se le proporciona a este un acceso.

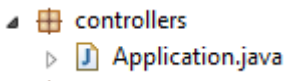


Así conseguimos quitar peso al modelo a la hora de acceder a la base de datos.

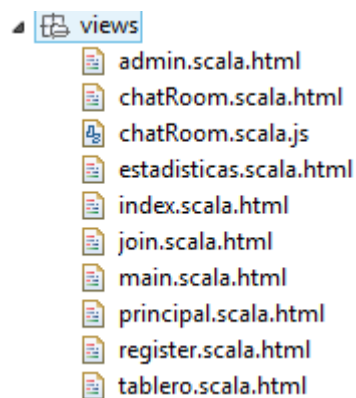
La capa de negocio también se encuentra separada.



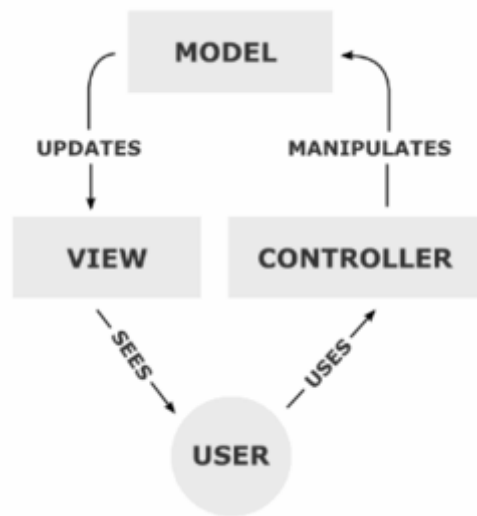
Aquí tenemos el controlador que será el encargado de recibir los eventos y enviar la petición al modelo para solicitarle información.



Y por último nos encontramos las vistas totalmente separadas de la lógica. De esta manera podríamos tener infinitas vistas que recibiesen información del modelo para mostrarlo en función de nuestros intereses y así darle una mayor flexibilidad al usuario que desea jugar la partida.



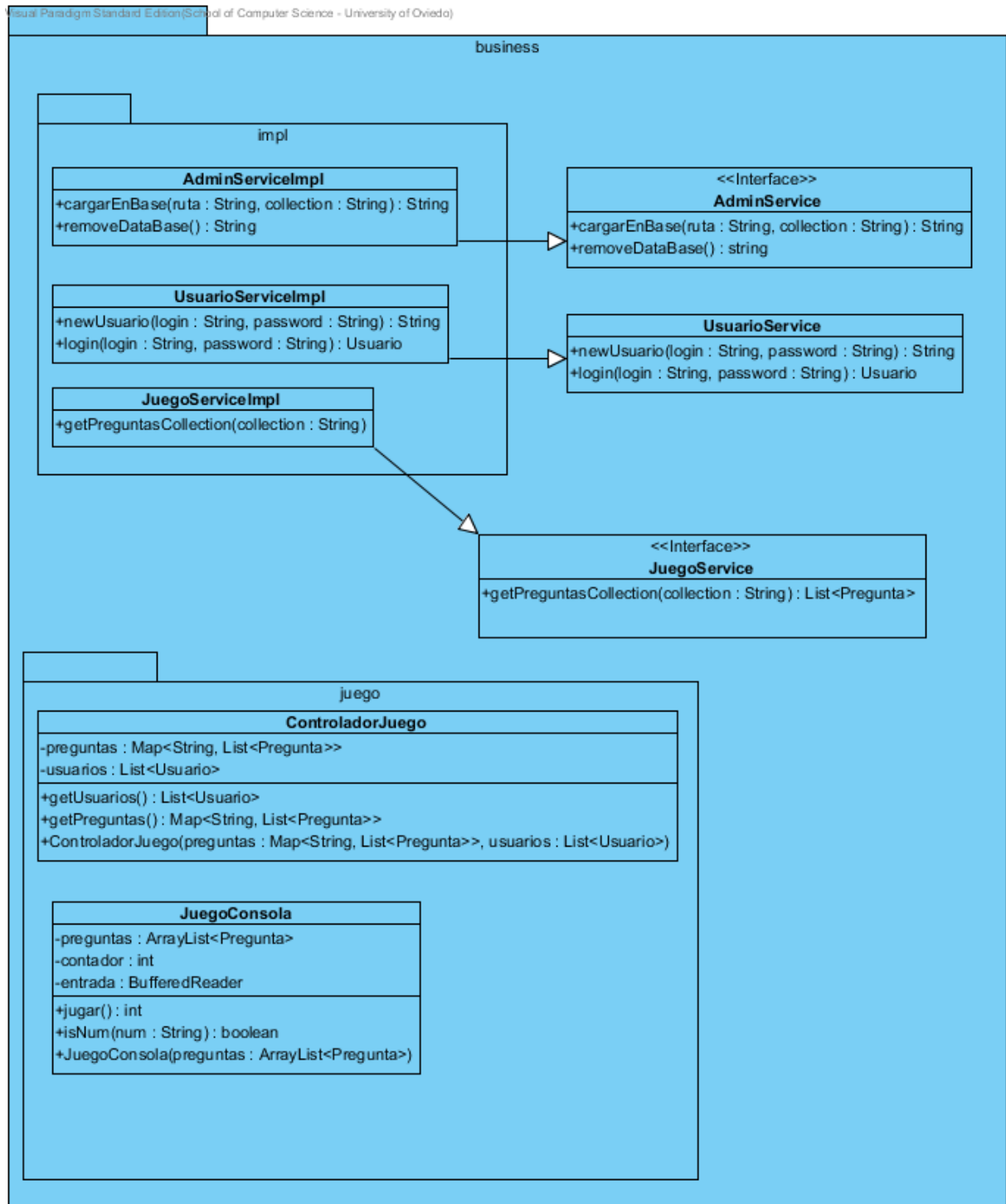
Como se ve la arquitectura sigue bastante bien un MVC como sería el siguiente.



Como se puede apreciar. El usuario que desea jugar, usaría el controlador pulsando sobre los distintos botones como podrían ser las casillas del tablero, la respuesta a una pregunta y toda la funcionalidad que se encuentra en la web. Este controlador enviaría la solicitud al modelo, por ejemplo solicitaría las estadísticas de un jugador y este actualizaría la vista para que el jugador fuese mostrado al usuario que está jugando.

## **DIAGRAMAS Y VISTAS**












## Diagrama de clases 1



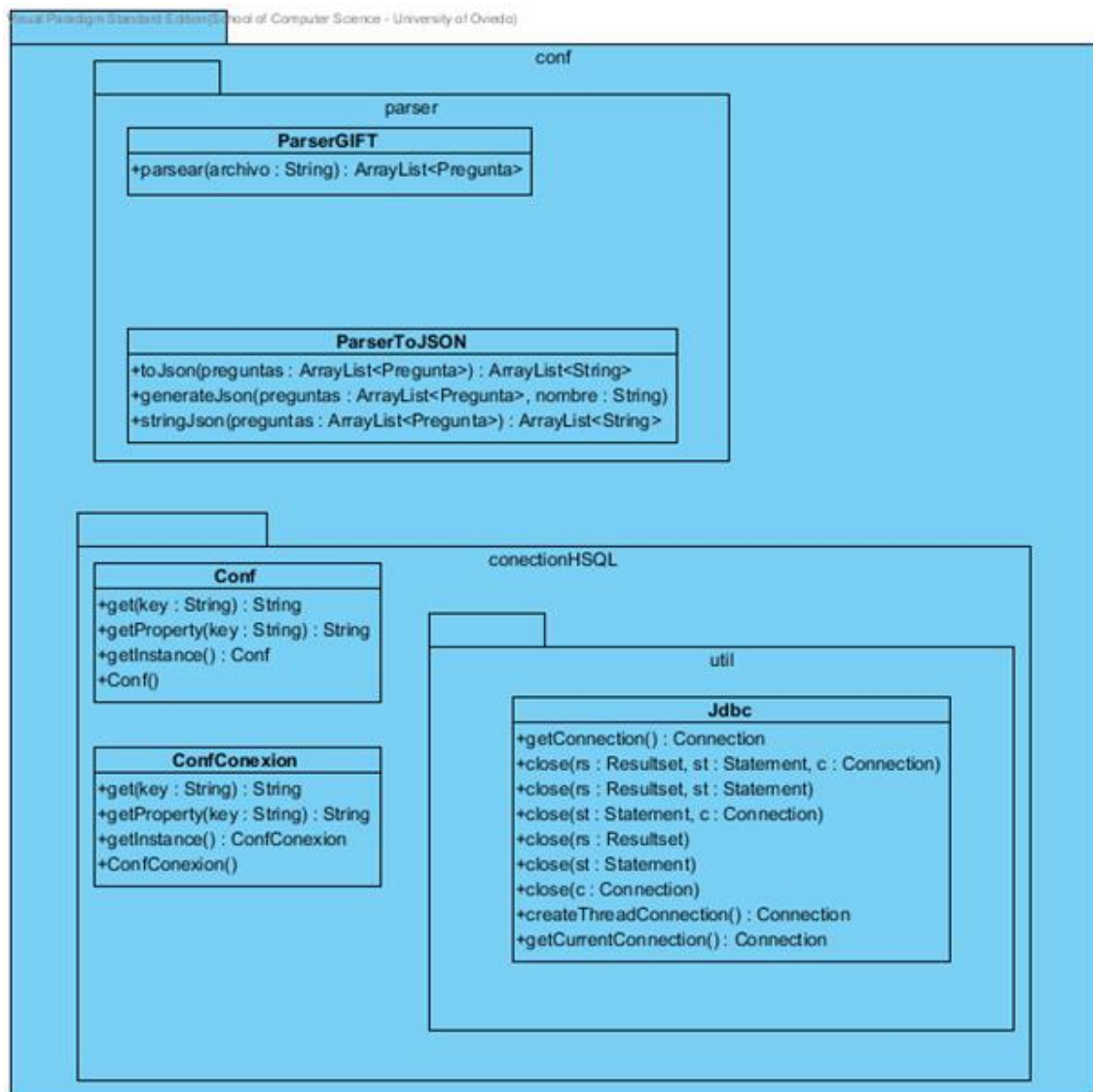
## Resumen diagrama de clases 1

Nombre	Documentación
--------	---------------












 business	Paquete obtención de datos de la base
 impl	Diferentes implementaciones para la obtención de datos
 AdminServiceImpl	Implementación concreta para cargar preguntas
 AdminService	Carga las preguntas de base
 UsuarioServiceImpl	Implementación concreta para listar las preguntas
 UsuarioService	Recuperación de la lista de preguntas
 JuegoServiceImpl	Implementación concreta para logear y crear usuarios
 JuegoService	Logea y crea usuarios
 juego	Clases encargadas del control del juego
 ControladorJuego	Guarda las preguntas y usuarios del sistema para ser usados
 JuegoConsola	Clase para jugar en línea de comandos, a modo de prueba

## Diagrama de clases 2

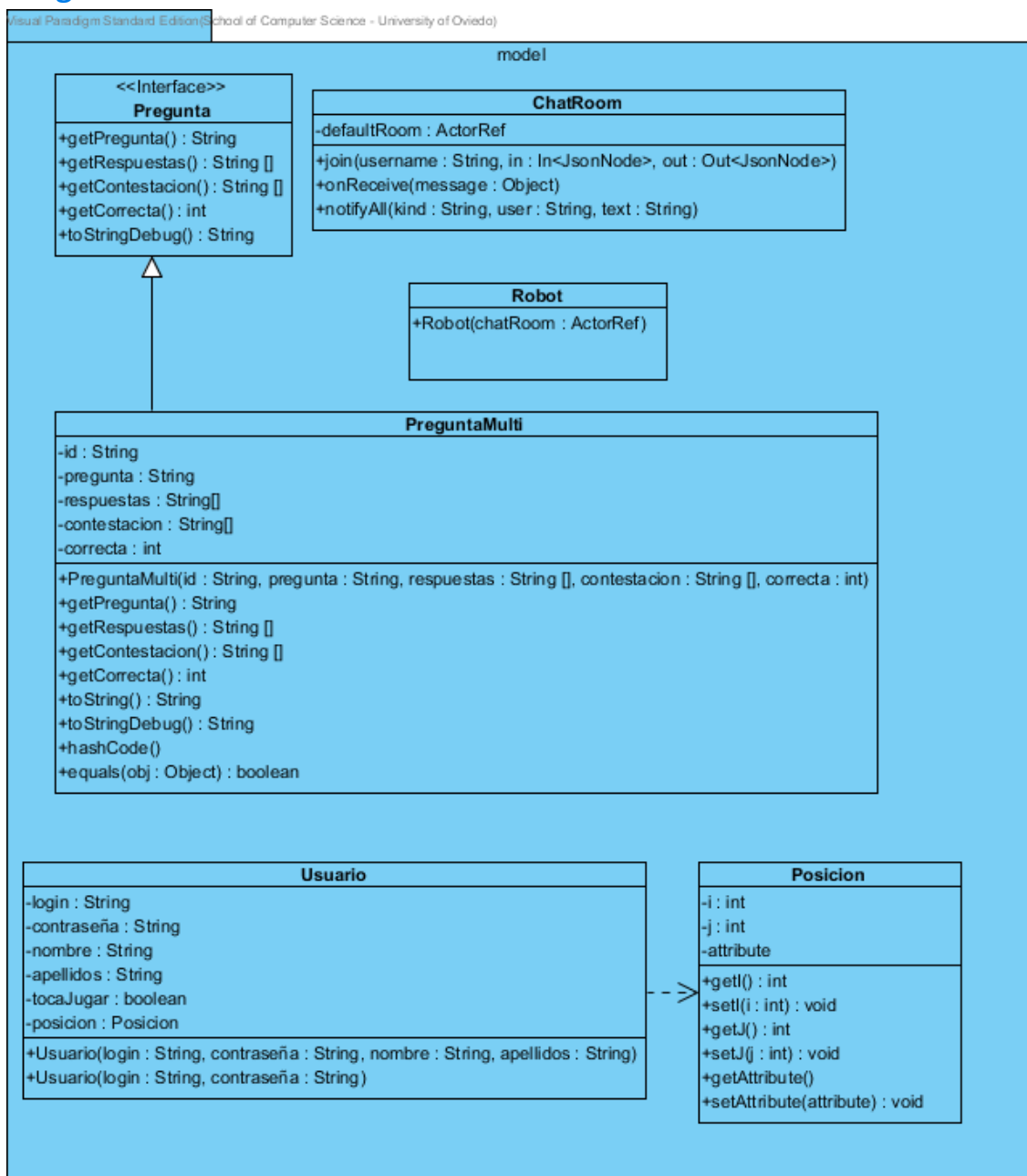


## Resumen diagrama de clases 2








Nombre	Documentación
 conf	Parseadores de JSON y mantenimiento base datos hsql
 parser	Parseadores
 ParserGIFT	Recupera archive de preguntas de GIFT
 ParserToJson	Crea un JSON con preguntas pasadas
 conexionHSQL	Uso de conexiones HSQL
 Conf	Recupera archive de propiedades para consultas

 util	Uso de la HSQL
 Jdbc	Gestiona base datos
 ConfConexion	Recupera archive de propiedades para la conexión

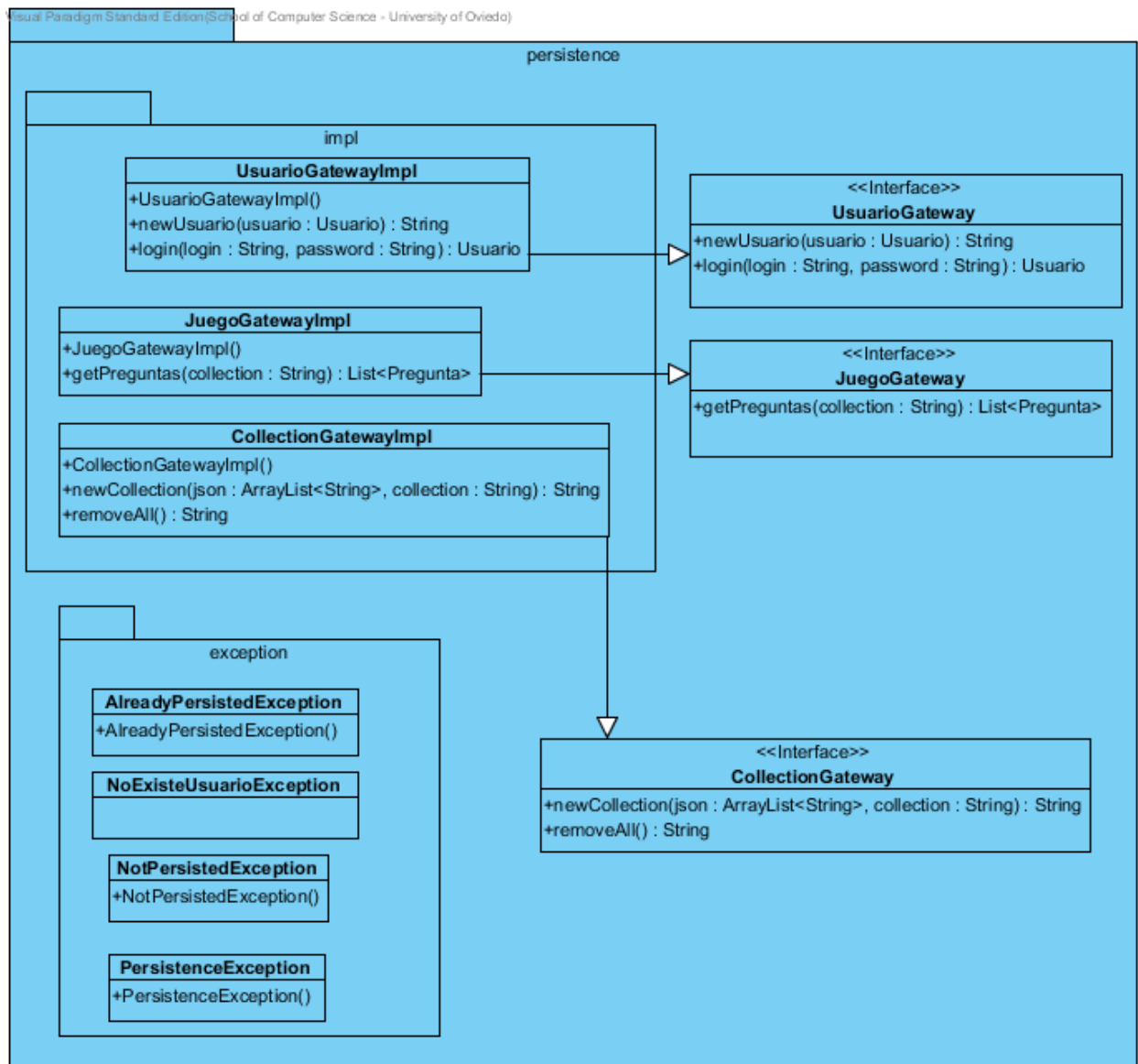
### Diagrama de clases 3









### Resumen diagrama de clases 3








Nombre	Documentación
 model	Gestión de entidades, como usuario y pregunta
 Pregunta	Recupera datos de una pregunta, como contestaciones.
 PreguntaMulti	Tipo de Pregunta específica
 Usuario	Modelo de usuario
 ChatRoom	Gestión de la sala de chat para los usuarios
 Robot	Clase que genera respuestas para usuario
 Posicion	Clase que proporciona coordenadas x,y

## Diagrama de clases 4

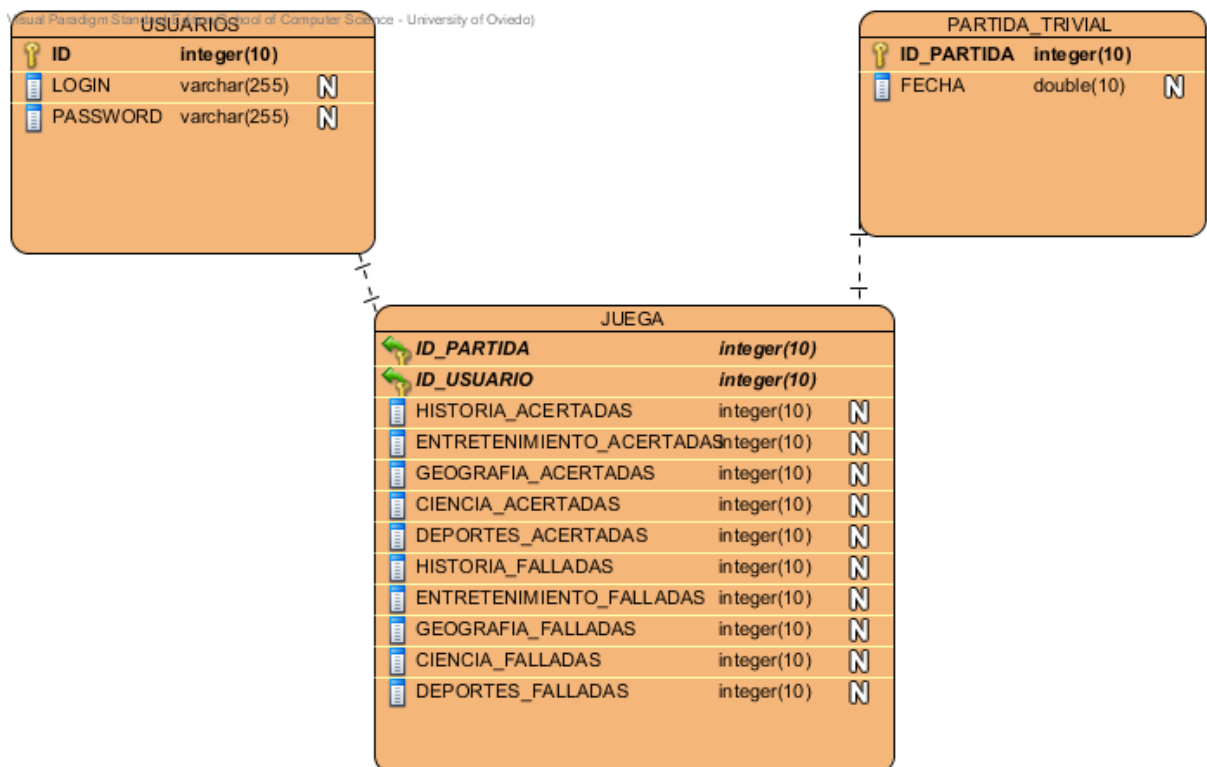


## Resumen diagrama de clases 4




Nombre	Documentación
 persistence	Paquete de persistencia para consultas
 impl	Diferentes implementaciones par alas gateway
 UsuarioGatewayImpl	Implementación concreta para gestión de usuarios
 UsuarioGateway	Gestión de usuarios, como creación de estos
 JuegoGatewayImpl	Implementación concreta para gestión de partidas
 JuegoGateway	Gestión de partidas recuperación de la lista de preguntas

 CollectionGatewayImpl	Implementación concreta para la gestión de colecciones.
 exception	Paquete de excepciones propias utilizadas
 AlreadyPersistedException	Excepción, ya ha sido introducido dato
 CollectionGateway	Gestión de colección de datos
 NoExisteUsuarioException	Excepción, no existe usuario en base
 NotPersistedException	Excepción, no ha sido introducido dato
 PersistenceException	Excepción genérica

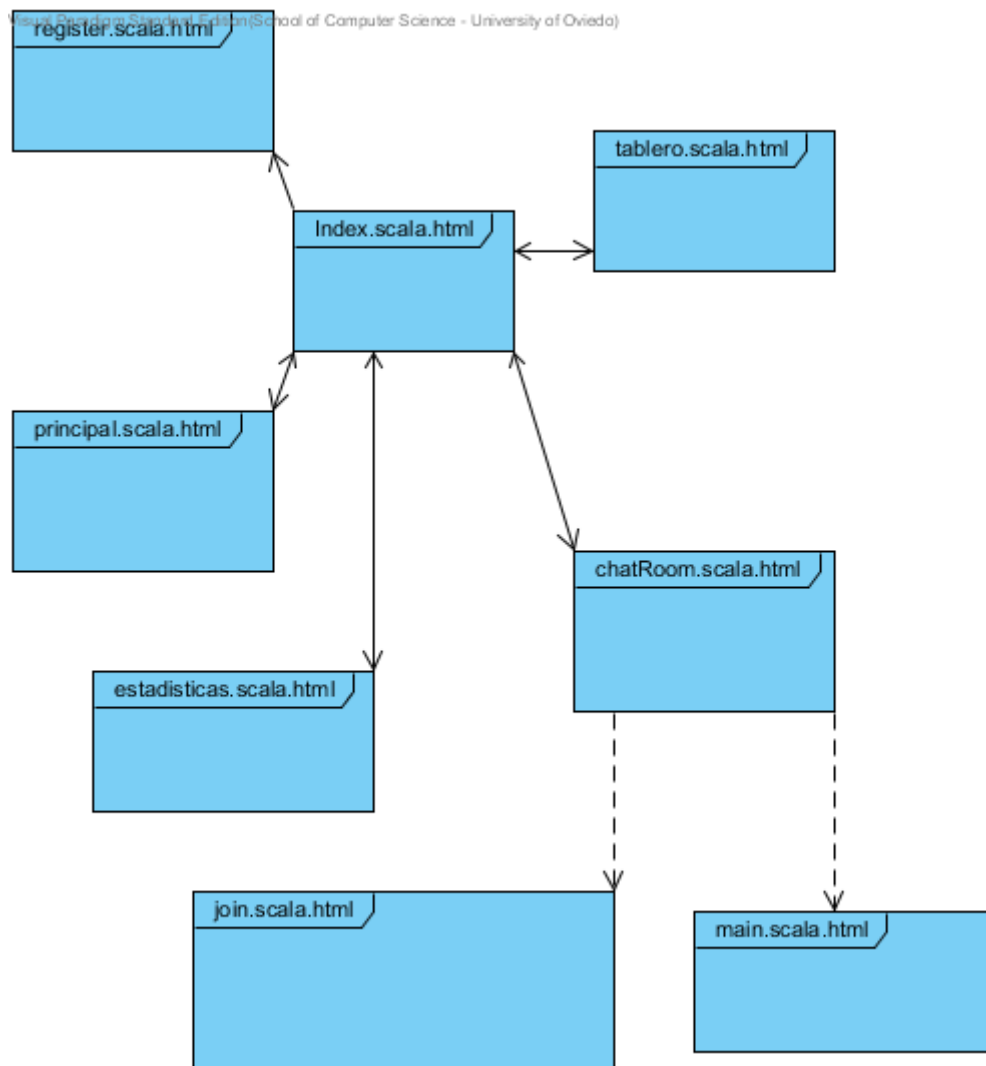
## Diagrama entidad relacional usuarios





### Resumen diagrama entidad relacion







Nombre	Documentación
 USUARIOS	Guarda datos de cada usuario, en principio el login y la password
 PARTIDA_TRIVIAL	Guarda la fecha de la partida
 JUEGA	Entidad intermedia entre las dos anteriores para relacionar partidas a usuarios.

### Diagrama interacción ventanas



### Resumen diagrama interacción de ventanas

Nombre	Documentación
 register.scala.html	Página para registrarse en el sistema
 tablero.scala.html	Página del tablero donde se jugará la partida

 Index.scala.html	Página donde uno se puede logear
 principal.scala.html	Página donde se pueden seleccionar todas las opciones disponibles
 chatRoom.scala.html	Página del chat de usuarios
 estadisticas.scala.html	Página donde se muestran las estadísticas de juego
 join.scala.html	Complemento para la página de chat
 main.scala.html	Complemento para la página de chat

## MANUAL DE USUARIO

A continuación se mostrará una guía con la cual el usuario podrá ejecutar sin problemas nuestro juego Treevial:

### Paso 1 - Descargar el framework PLAY

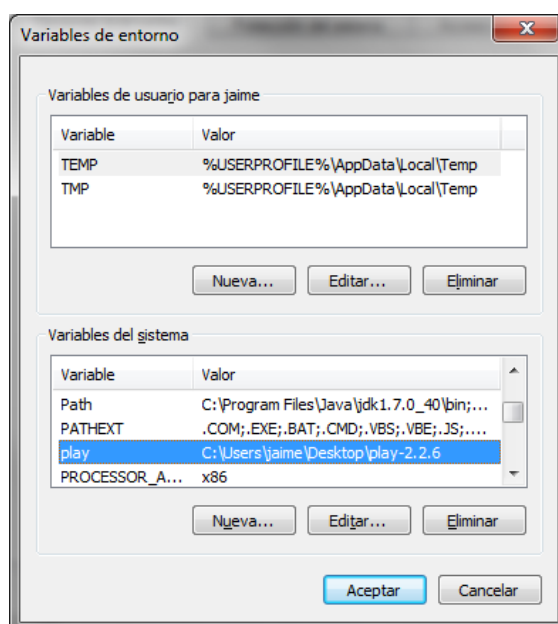
Para ello tendremos que tener descargado la versión de **PLAY 2.2.6**:

<http://downloads.typesafe.com/play/2.2.6/play-2.2.6.zip>

### Paso 2 - Realizar los ajustes necesarios en variables de entorno

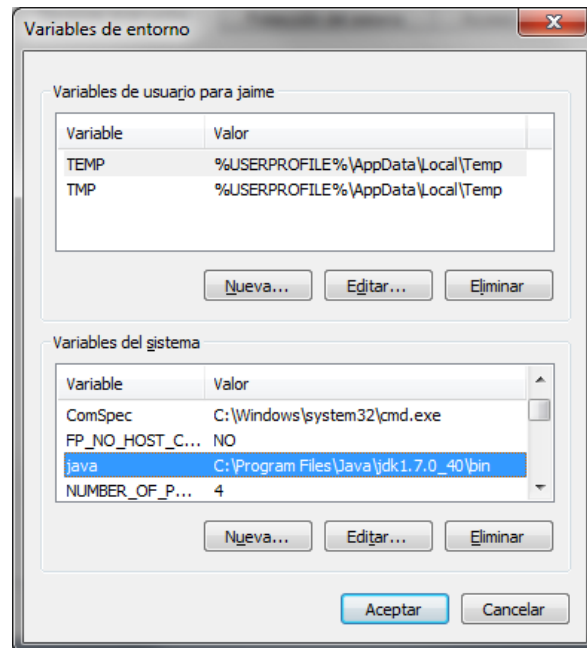
Es posible que encuentre problemas de cara ejecutar nuestro problema por culpa de la ubicación del Javac, por lo que tendremos que crear las siguientes variables de entorno:

Una para el play, dándole como ruta donde tengamos ubicada el framework que nos bajamos en el paso anterior:

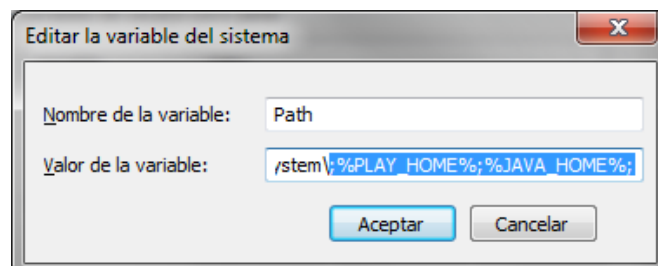




Otra variable de entorno para java, dándole como ruya el bin del jdk de la versión 1.7 (con la 1.8 no serviría)

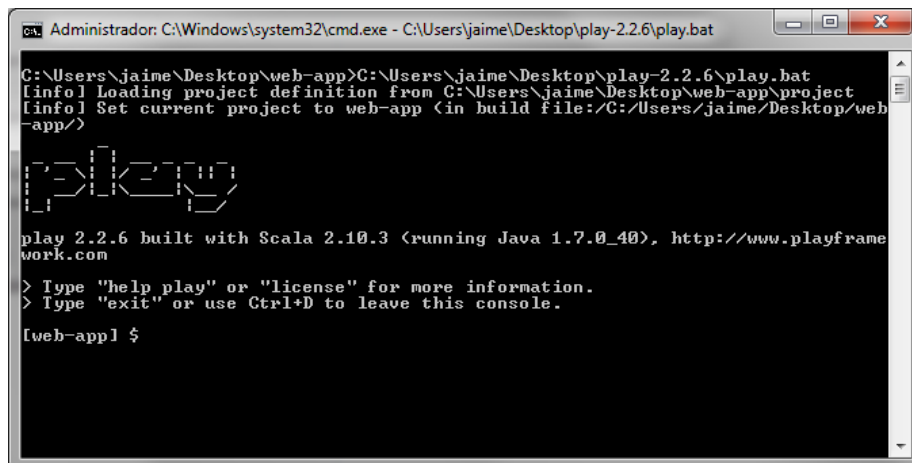


Por último, a la variable path le añadimos al final: ; %PLAY\_HOME%;%JAVA\_HOME%;



### Paso 3 - Arrancamos el servidor

- Sacamos fuera del proyecto maven el proyecto *web-app*
- En una línea de comandos, sobre la carpeta *web-app* ejecutamos el comando *play* del framework bajado



- Hacemos *eclipse*

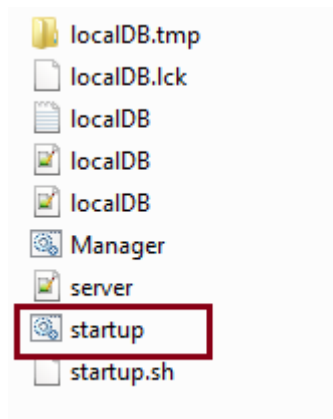
```
[web-app] $ eclipse
[info] About to create Eclipse project files for your project(s).
[info] Successfully created Eclipse project files for project(s):
[info] web-app
[web-app] $ _
```

- Una vez a finalizado, ejecutamos *run* .Con esto el servidor pasa a escuchar en el puerto 9000

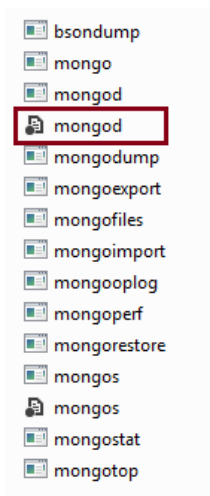
```
[web-app] $ run
--- (Running the application from SBT, auto-reloading is enabled) ---
[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000
(Server started, use Ctrl+D to stop and go back to the console...)
```

#### Paso 4 - arrancamos base de datos

Desde la carpeta *data* donde tenemos ubicada nuestra base de datos ejecutamos el *startup* para iniciarla. De ella es donde realizaremos toda la gestión de usuarios.

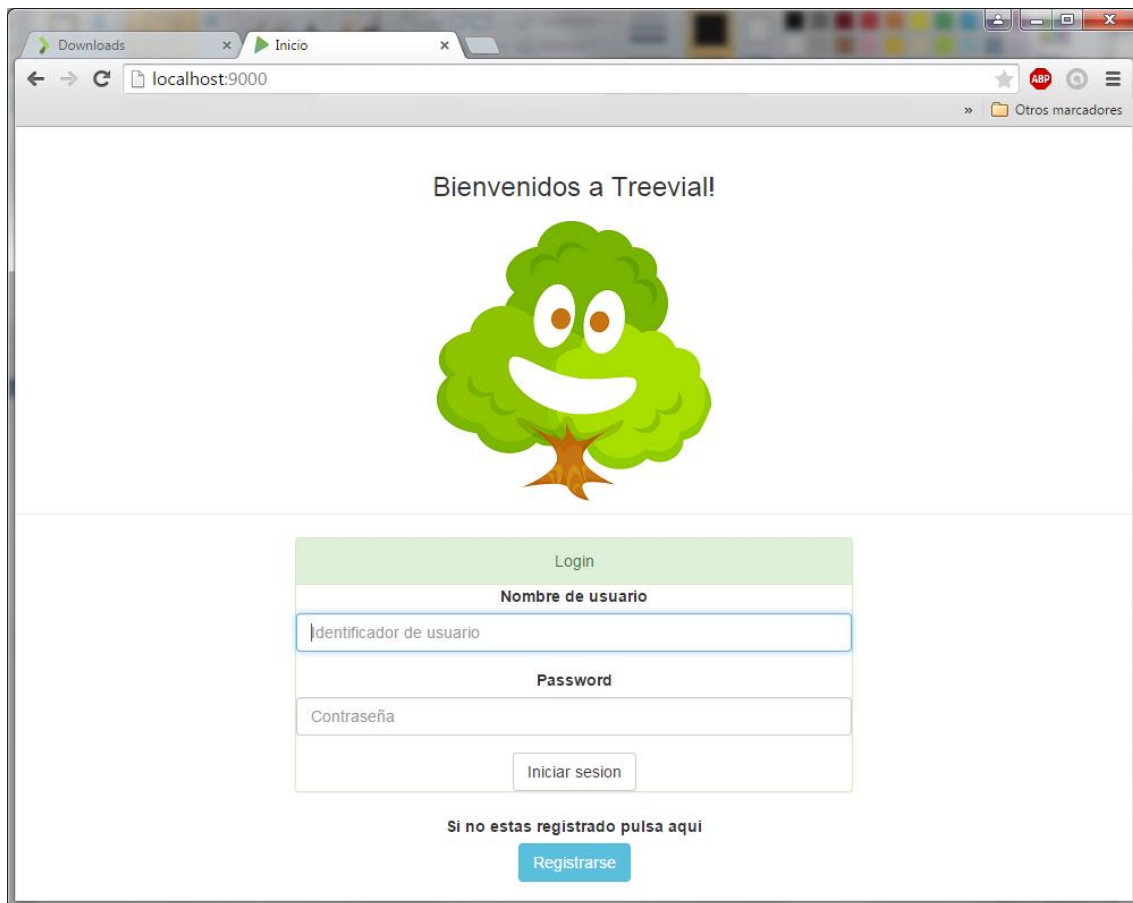


Arrancamos también el *mongod* de la carpeta donde este:



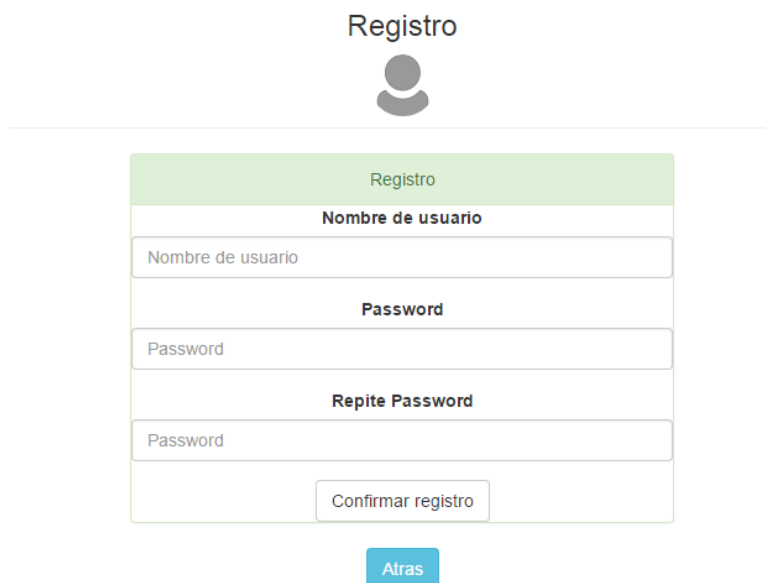
## Paso 5 - La aplicación

Una vez hecho esto, entrando en el puerto 9000 podremos comenzar a usar la aplicación



The screenshot shows a web browser window with the address bar set to `localhost:9000`. The page has a white background and a green header bar. Below the header, the text "Bienvenidos a Treevial!" is centered. Underneath is a large, stylized green tree with a smiling face. Below the tree is a login form with a green header bar labeled "Login". The form contains two input fields: "Nombre de usuario" (with a placeholder "Identificador de usuario") and "Password" (with a placeholder "Contraseña"). Below these fields is a button labeled "Iniciar sesion". At the bottom of the form, there is a link "Si no estas registrado pulsa aqui" and a blue button labeled "Registrarse".

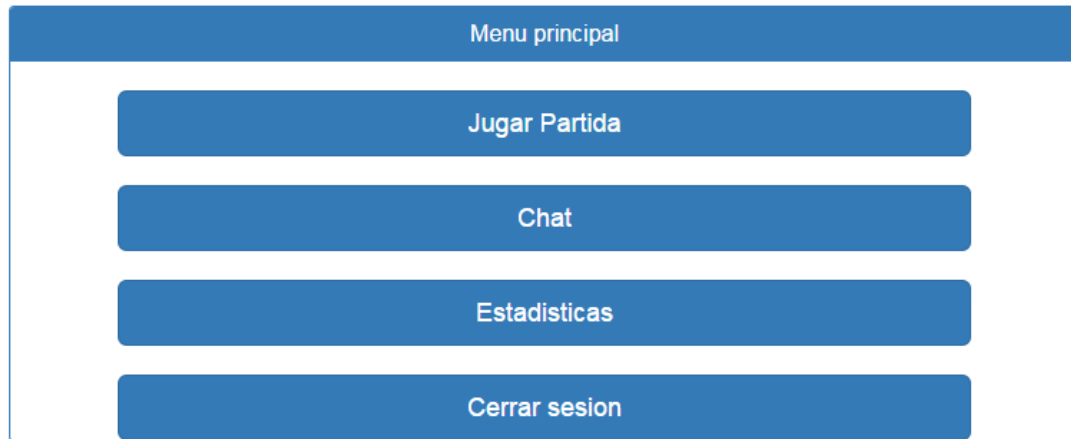
Desde la propia ventana principal podremos iniciar sesión con un usuario de la base de datos (por ejemplo Nombre: *jaim*e y Password: *jaim*e). O por otro lado registrarnos, dando nuestros datos.



The screenshot shows the registration page of the Treevial application. It has a white background and a green header bar. Below the header, the text "Registro" is centered, followed by a grey user icon. Below the icon is a registration form with a green header bar labeled "Registro". The form contains three input fields: "Nombre de usuario", "Password", and "Repite Password". Below these fields is a button labeled "Confirmar registro". At the bottom of the form, there is a blue button labeled "Atras".

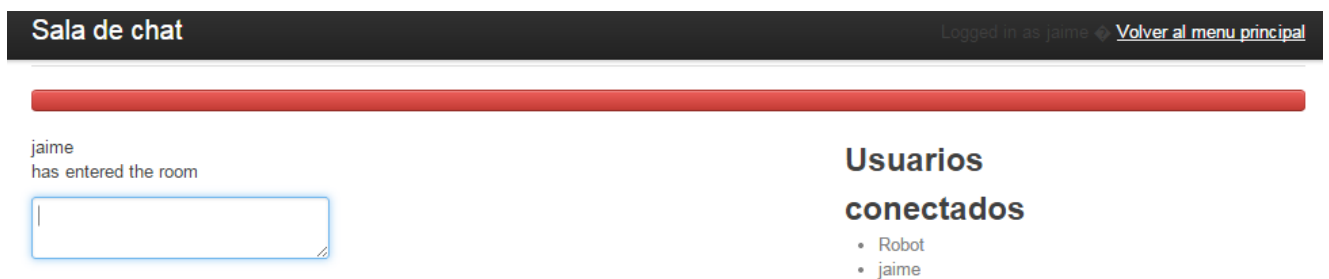
Una vez entremos en sesión podremos elegir diferentes opciones como usuario:

Bienvenido jaime

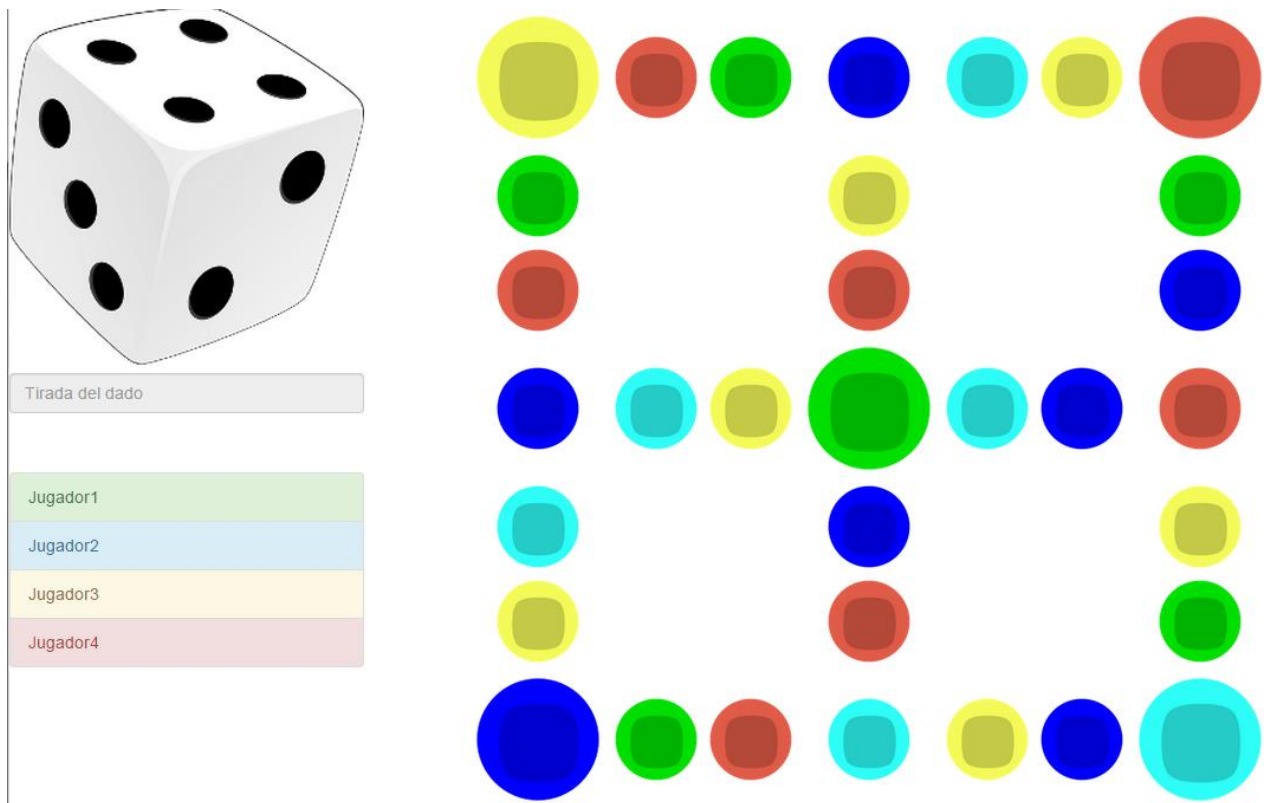


- **Jugar partida:** para comenzar una partida del trivial
- **Chat:** para comenzar un chat con otros usuarios. Aunque aún estaría en desarrollo y de momento se ofrece solo un bot con el que hablar.
- **Estadísticas:** las estadísticas que tenga el usuario en todas sus partidas
- **Cerrar sesión:** para cerrar la sesión en curso del usuario.

Desde la pestaña del chat, nos podremos unir y ver todos los usuarios conectados:



Y por último, desde la ventana de la partida podremos jugar como con nuestra aplicación de escritorio:



## OTROS DATOS DE INTERES

- <https://github.com/Arquisoft/Trivial2b/wiki> Lista de actas
- <http://arquisoft.github.io/Trivial2b/> Web de la aplicación