# 1. Planteamiento del problema

El departamento de marketing de la empresa NoGame señala que el juego del Trivial tiene gran potencial en un entorno Web, donde los usuarios puedan conectarse y jugar entre ellos.

Partiendo de la versión de escritorio del juego se trata de reutilizar lo máximo posible para portarlo a una versión Web, reduciendo costes y tiempo. Se desea que los jugadores puedan registrarse fácilmente y con su cuenta lleva un registro de sus estadísticas, tales como número de partidas jugadas, preguntas falladas, preguntas acertadas, etc. Una vez conectados los jugadores podrán ver si hay otros usuarios conectados e invitarlos para participar en una partida.

La forma de jugar seria análoga a la versión de escritorio , con ciertas particularidades derivadas del juego online, por ejemplo la perdida de conexión, el abandono de los jugadores de forma intencionada y otros supuestos que deberán solucionarse de la forma más elegante posible para que dentro de lo posible el resto de jugadores puedan seguir jugando sin interrupción.

La fuente de monetización principal será la publicidad, siendo necesaria una masa de usuarios importante y que la aplicación los soporte. Se requiere una arquitectura escalable y tiempos de respuesta razonables a las acciones de los usuarios.

Como en la versión de escritorio la empresa desea una separación de la lógica del juego y su capa de presentación para poder experimentar y ofrecer distintas visualizaciones a los usuarios y evolucionarla con las cambiantes exigencias del diseño web. Además la empresa está muy interesada en que la solución funcione en diferentes tipos de dispositivos, como la gran variedad de dispositivos móviles en auge, y por ello está buscando una solución que cumpla con los principios y estándares Web de diseño responsable.

# Prototipos

Se desarrollará una aplicación Web que permitirá realizar una gestión de usuarios y de partidas. La visualización gráfica en tiempo real de las partidas es opcional. Se recomienda la separación entre la lógica de la aplicación del servidor y los aspectos visuales mediante el uso de API REST. Se recomienda la utilización de tecnologías estándares.

En el prototipo se permite la posibilidad de que en cada partida concreta solamente haya un jugador. Esta opción puede facilitar bastante la arquitectura al no requerir que los clientes de los jugadores de una partida tengan que actualizar su información en función de los movimientos de otro jugador.

Si se considera necesario, se podrán entregar varios prototipos independientes que demuestren la diferente funcionalidad, aunque se recomienda que estén integrados.

# Diagrama Contextual

**(por hacer)**

# primer acercamiento a la solución

El uso de una API REST se apoya en el estándar HTTP y gracias a tener un protocolo cliente/servidor sin estado la aplicación será más fácilmente escalable.

También el ya conocido patrón de arquitectura MVC (modelo-vista-controlador) permite separar los datos y la lógica de negocio de la interfaz de usuario, nos basaremos en el framework Play, usándolo con Java ya que es el lenguaje más conocido por los alumnos de la escuela.

Para la persistencia, como aprendimos al finalizar el modulo anterior, usaremos MongoLab, un servicio de base de datos en la nube basado en MongoDB que facilita el trabajo y ofrece las ventajas de la nube, como una gran escalabilidad.

# riesgos relacionados con la solución

1. Desconocimiento de nuevas tecnologías.

Es posible que la mayoría de integrantes del grupo de desarrollo desconozcan total o parcialmente algunas de las tecnologías a utilizar, ya sea el framework Play, MongoLab u otras tecnologías web.

Debido al escaso margen de tiempo de desarrollo este punto puede ser crítico.

1. Problemas con la red.

Al ser una aplicación orientada a la web, deben gestionarse los riesgos de una caída de red, del abandono de los jugadores u de otras causas que puedan afectar al juego y que no se tienen en cuenta en una aplicación de escritorio como en la que se basa la actual.

1. Disponibilidad de MongoLab.

Aunque en un principio no debería dar problemas, dependemos de este servicio para la persistencia y su disponibilidad 24/7 no está en nuestras manos.

# 2. Metodología usada

Se va a realizar un estudio de arquitectura siguiendo el método de ADD (Atribute-Driven Design) y la norma del SEI (ANSI/IEEE 1471, 2000) el cual sigue los siguientes pasos:

Paso 1: Confirmar que hay suficiente información.

Paso 2: Elegir un elemento del sistema a descomponer.

Paso 3: Identificar candidatos a *drivers* arquitectónicos.

Paso 4: Elegir un concepto de diseño que satisfaga los *drivers* arquitectónicos.

Paso 5: Instanciar los elementos arquitectónicos y disponer responsabilidades.

Paso 6: Definir interfaces para los elementos instanciados.

Paso 7: Verificar y refinar los requisitos y convertirlos en restricciones para los elementos instanciados.

Paso 8: Repetir los pasos 2 a 7 para el siguiente elemento del sistema que se desee descomponer.

# 3. Stakeholders

* **Usuarios/Clientes**

Serán los que utilicen la aplicación y pueden tener varios roles

* + Usuario no registrado
  + Usuario registrado
  + Administrador
* **Directores del proyecto (NoGame)**

En este caso son los responsables de la empresa NoGame, los cuales deciden los requisitos, plazos y presupuesto para realizar la aplicación.

* **Equipo de desarrollo**

El equipo 5A formado por alumnos de la asignatura ASW de la EII de Oviedo.

# 3.1 Lista de Stakeholders

|  |  |  |
| --- | --- | --- |
| Código | Stakeholder | Intereses |
| ST-01 | Directores del proyecto | Que se realice en el tiempo estimado.  Que sea flexible y permita fáciles modificaciones de cara a futuro.  Que no exceda el presupuesto.  Que se cumplan los requisitos funcionales y no funcionales impuestos.  Amplia compatibilidad en diversos dispositivos.  Que la aplicación Web sea capaz de generar ingresos mediante publicidad gracias a la escalabilidad y uso masivo de usuarios. |
| ST-02 | Equipo de desarrollo | Que el proyecto sea rentable, lo cual quiere decir que este dentro del presupuesto establecido.  Proyecto flexible, lo cual quiere decir que las modificaciones futuras sean fáciles de introducir.  Que no requiera grandes esfuerzos en adaptar tecnologías nuevas y desconocidas, desarrollo controlado y predecible. |
| ST-03 | Usuarios/Clientes | Registro rápido y sencillo.  Usar la aplicación como herramienta de aprendizaje.  Estudiar las estadísticas de los demás usuarios (Administrador)  Uso lúdico y de entretenimiento.  Que la aplicación sea fácil de usar, segura, consistente e intuitiva. |

# 4 Requisitos funcionales

* La aplicación debe funcionar en un entorno Web multidispositivo.
* Tener un coste de desarrollo dentro del presupuesto.
* Tiempo de realización dentro del establecido por la empresa NoGame.
* Las reglas del juego serán las del Trivial Pursuit.
* El número de jugadores será de 2 a 6.
* Al tirar el dado deben indicarse las casillas a las que el jugador puede moverse.
* Deben registrarse las estadísticas del juego.
* Los jugadores deberán hacer login y estar previamente registrados.
* Distinción entre la parte administrativa y la del jugador.
* El juego deberá tomar las preguntas de una BBDD previamente creada.
* Debe permitirse el juego online.
* El juego debe continuar aunque alguno de los jugadores abandone.

# 4.1 Requisitos no funcionales

* Uso razonable de recursos del sistema y de la red.
* Facilidad de uso.
* Agradable visualmente.
* Cerrado para modificar, abierto para extender.
* Código legible y bien documentado.

# 5 identificación inicial de los atributos de calidad

* **Modificabilidad**

Facilidad a la hora de añadir nuevas funcionalidades como por ejemplo variar el número de jugadores, de categorías de preguntas, elementos visuales…

* **Disponibilidad**

Disponibilidad del sistema continua para permitir partidas online.

* **Rendimiento**

Uso razonable de los recursos de la maquina cliente y de la red.

* **Escalabilidad**

La aplicación debe estar preparada para admitir a una gran cantidad de usuarios.

* **Seguridad**.

Login para jugadores y administrador.

Los datos guardados no deben ser accesibles por elementos o personas ajenos al sistema.

* **Testabilidad**

Sencillez y automatización de las pruebas para el correcto funcionamiento del sistema.

Facilitar depuración de la lógica del juego.

Permitir la observación de los resultados intermedios.

* **Usabilidad**.

Sencillez de uso por parte de los usuarios.

Interfaz gráfica adaptada a la web y multidispositivo.

Disponibilidad de manual de usuario.

* **Mantenibilidad**.

Abierto para extender, cerrado para modificar.

Bajo acoplamiento.

Código documentado.

* **Time-to-market.**

Cumplir con los plazos de entrega requeridos.

* **Coste-beneficio**.

Cumplir con los presupuestos pactados.

# 5.1 lista de atributos de calidad

|  |  |  |
| --- | --- | --- |
| Código | Descripción | Tipo de atributo |
| AT001 | Modularidad del sistema, es posible que se reutilicen módulos en otros juegos o variantes del mismo. | Modificabilidad |
| AT002 | Facilidad para adaptar la lógica del juego a diferentes tipos de tablero, número de jugadores, etc. | Modificabilidad |
| AT003 | Flexibilidad para futuras adaptaciones a otros dispositivos, principalmente móviles (Smartphone/Tablet). | Modificabilidad |
| AT004 | Facilidad para cambiar la representación visual y bajo acoplamiento de esta con la lógica de la aplicación. | Modificabilidad |
| AT005 | Integridad en los datos almacenados de los jugadores y estadísticas. | Seguridad |
| AT006 | Evitar accesos no autorizados (login). | Seguridad |
| AT007 | Poder comprobar la fiabilidad del sistema usando integración continua. | Testabilidad |
| AT008 | Mantenible por los desarrolladores. | Mantenibilidad |
| AT009 | Facilidad para añadir nuevos algoritmos al código. | Modificabilidad |
| AT010 | Tiempo de desarrollo muy bajo. | Time to market |
| AT011 | Coste de desarrollo muy bajo. | Coste-Beneficio |
| AT012 | Uso razonable de los recursos del sistema. | Rendimiento |
| AT013 | Uso razonable de red. | Rendimiento |
| AT014 | Facilidad de uso por el usuario. | Usabilidad |
| AT015 | Disponibilidad de manuales. | Usabilidad |
| AT016 | Disponibilidad del sistema para su uso online. | Disponibilidad |
| AT017 | Preparado para una gran cantidad de usuarios. | Escalabilidad |

# 5.2 atributos de calidad e interesados

|  |  |  |  |
| --- | --- | --- | --- |
| Atributos vs interesados | ST-01 | ST-02 | ST-03 |
| AT001 | **X** | **X** |  |
| AT002 |  | **X** |  |
| AT003 | **X** | **X** |  |
| AT004 |  | **X** |  |
| AT005 | **X** |  |  |
| AT006 | **X** |  |  |
| AT007 |  | **X** |  |
| AT008 |  | **X** |  |
| AT009 |  | **X** |  |
| AT010 | **X** |  |  |
| AT011 | **X** | **X** |  |
| AT012 | **X** |  | **X** |
| AT013 | **X** |  | **X** |
| AT014 | **X** |  | **X** |
| AT015 | **X** |  | **X** |
| AT016 | **X** |  | **X** |
| AT017 | **X** |  |  |

# 5.3 escenarios de calidad

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nº escenario | Fuente de estímulo | Estímulo | Entorno | Artefacto | Respuesta | Medición Respuesta | Atributo de calidad afectado |
| 1 | Desarrollador/Nueva necesidad | Añadir funcionalidad | Desarrollo | Módulos del sistema | Funcionalidad implementada | No es necesario modificar código existente (abierto para extender, cerrado para modificar) | AT001, AT008, AT009 |
| 2 | Desarrollador | Modificar lógica de juego | Desarrollo | Lógica del sistema | Lógica modificada (tablero, nº jugadores) | No es necesario modificar código existente (abierto para extender, cerrado para modificar) | AT002, AT008,  AT009 |
| 3 | Desarrollador | Adaptar a nuevo dispositivo | Desarrollo | Sistema | Sistema funcionando en entorno nuevo | Esfuerzo menor en adaptar que en crear de cero | AT003 |
| 4 | Desarrollador | Cambio de interfaz | Desarrollo | UI | UI nueva o cambios configurables sobre la existente | No hay que tocar código de lógica del programa | AT004 |
| 5 | Usuario | Uso de la aplicación | Explotación | Datos | Se mantiene la integridad de los datos | Corrupción de datos = 0 | AT005 |
| 6 | Usuario | Login al inicio de la aplicación | Explotación | Seguridad | No permite accesos no autorizados | Accesos no autorizados = 0 | AT006 |
| 7 | Desarrollador | Probar Test | Desarrollo | Sistema - Pruebas | Pasa las pruebas unitarias | Pasan todas las pruebas | AT007 |
| 8 | Travis CI | Integración continua | Desarrollo | Sistema | Compilación y pruebas | Cobertura de código >=90% con resultado positivo | AT007 |
| 9 | Director Proyecto | Fin del desarrollo | Pre-explotación | Sistema | Bajo tiempo de desarrollo | < 4 semanas  Fecha limite 5 de mayo | AT010 |
| 10 | Director Proyecto | Fin del desarrollo | Pre-explotación | Sistema | Bajo coste de desarrollo | <= que el presupuesto (0€) | AT011 |
| 11 | Usuario | Uso de la aplicación | Explotación | Equipo local | El sistema funciona fluidamente | Uso de CPU y RAM <30% en un equipo estándar de 400€ (al menos doble núcleo y 2GB RAM) | AT012 |
| 12 | Usuario | Uso de la aplicación | Explotación | Red | Sistema online | No hay interrupción de red y la latencia es <500ms | AT013 |
| 13 | Usuario | Uso de la aplicación | Explotación | UI | El usuario no necesita ayuda externa | Manual de usuario disponible y necesidad de ayuda externa = 0 | AT014, AT015 |
| 14 | Usuario | Abandona la partida antes de tiempo | Explotación | Sistema | El resto de jugadores pueden seguir | El juego no se bloquea y puede continuar | AT016 |
| 15 | Usuarios | Gran cantidad de conexiones | Explotación | Sistema | El sistema continua funcionando correctamente | El sistema funciona con más de 100 usuarios | AT017 |