

# Máster Universitario en Ingeniería Informática

## TRABAJO FIN DE MÁSTER

DESARROLLO DE UN SISTEMA DE JUEGO SOCIAL SOBRE UN  
SISTEMA DE MENSAJERÍA. CHATBOT DE JUEGO SOCIAL.



**UNIVERSIDAD  
DE GRANADA**



Carlos Morales Aguilera

*30 de noviembre de 2020*

# Índice

<b>1. Introducción.</b>	<b>3</b>
<b>2. Estado del arte.</b>	<b>3</b>
2.1. Industria del videojuego . . . . .	3
2.2. Juego social . . . . .	5
2.3. Juegos para mayores . . . . .	8
2.4. Chatbots . . . . .	10
2.5. Plataformas . . . . .	12
2.6. Herramientas . . . . .	15
2.7. Análisis del juego en los chatbots . . . . .	20
2.8. Conclusiones . . . . .	23
<b>3. Análisis inicial del problema.</b>	<b>24</b>
3.1. Arquitectura . . . . .	28
<b>4. Tecnologías a usar.</b>	<b>29</b>
4.1. Gestor de proyecto . . . . .	29
4.2. Lenguaje de programación y APIs . . . . .	29
4.2.1. API chatbot . . . . .	29
4.2.2. Lenguaje de programación . . . . .	30
4.2.3. Librería de programación de API . . . . .	30
4.3. Gestor de tareas . . . . .	32
4.4. Test . . . . .	32
4.4.1. Marco de pruebas . . . . .	32
4.4.2. Biblioteca de aserciones . . . . .	33
4.5. Control de versiones . . . . .	33
4.6. Integración Continua . . . . .	33
4.7. Almacenamiento . . . . .	33
4.8. Contenerización . . . . .	34
4.9. Despliegue . . . . .	34

<b>5. Metodologías a usar en el proyecto.</b>	<b>35</b>
5.1. SCRUM . . . . .	35
5.2. Kanban . . . . .	37
5.3. Diseño centrado en el usuario . . . . .	37
5.4. GDD . . . . .	38
5.5. Aplicación de las metodologías . . . . .	39
<b>6. Plan de entregas.</b>	<b>41</b>
<b>7. Desarrollo. Entrega e iteraciones.</b>	<b>44</b>
7.1. Entrega 1 . . . . .	44
7.2. Entrega 2 . . . . .	48
7.3. Entrega 3 . . . . .	54
7.4. Entrega 4 . . . . .	59
7.5. Entrega 5 . . . . .	64
7.6. Entrega 6 . . . . .	69
<b>8. Conclusiones y Trabajo futuro.</b>	<b>70</b>
<b>9. Bibliografía</b>	<b>71</b>

## **1. Introducción.**

## **2. Estado del arte.**

### **2.1. Industria del videojuego**

Desde hace muchos siglos, los juegos se han utilizado como una herramienta de educación, interacción y formación, a través de la interacción social y como un medio fundamental para la estructuración del lenguaje y el pensamiento. Entre las principales motivaciones se encuentran la reducción de la sensación de gravedad frente a errores, la invitación a la participación activa por parte del jugador y desarrollo de la creatividad, competencia intelectual, fortaleza emocional y estabilidad personal. Por otra parte se encuentra el fin principal de un videojuego: la diversión.

A su vez, desde el comienzo de los videojuegos online, se ha ido demostrando los diversos efectos que estos producen en los jugadores, como pueden ser la creación de lazos emocionales de amistad, la inclusión social y otras ventajas que se obtienen a través de la práctica de los mismos. Los juegos se utilizan para motivar a los jugadores a desenvolverse en entornos virtuales con un fin determinado, y lograr una serie de objetivos que le permitan aprender a desenvolverse en un determinado entorno y socializar.

Por otra parte, dentro de los diferentes sectores de la industria del videojuego se pueden encontrar los videojuegos sociales, que consisten en la interacción de diferentes jugadores con un fin colaborativo o competitivo, con el objetivo de alcanzar una determinada meta o logro. El objetivo principal de estos videojuegos consiste en el desarrollo de una serie de capacidades cognitivas e interacción entre personas con el fin de alcanzar objetivos como la inclusión social o el desarrollo de determinadas habilidades mediante el entretenimiento.

Por otro lado, se ha demostrado mediante estudios como el de la Universidad de Montreal [1], que las personas con una participación regular en videojuegos disminuyen su deterioro cognitivo, ayudan a la prevención de enfermedades como el Alzheimer o incluso permiten una inclusión social mayor, ofreciendo una compañía virtual. Además, está la cualidad prosocial implícita que poseen los videojuegos, la cual permite no solo conocer a nuevas personas que aporten nuevos estímulos, sino también compartir determinadas emociones, vivencias y momento. Estos dos aspectos son dos grandes puntos de partida a la hora de plantear un entorno de juego para personas mayores, con el fin de que socialicen y encuentren un entretenimiento diferente al tradicional.

El enfoque principal a la hora de diseñar, desarrollar y lanzar un videojuego es el de buscar un público joven, juvenil o adolescente en su mayoría, con el fin de obtener las mayores ganancias posibles, dejando de lado un sector como puede ser el de las personas mayores, el cual es un público algo más complicado de cara a la realización de videojuegos. Estos requieren juegos con los que se sientan cómodos y no requiera un gran aprendizaje por parte de los usuarios. Es por este motivo por el que la industria actual del videojuego está más enfocada en un público joven, más acostumbrado al uso de las tecnologías y con mayor facilidad de aprendizaje de las diferentes tecnologías.

Sin embargo, la industria del juego se remonta a mucho antes de la existencia de los videojuegos, donde los principales juegos eran o bien de cartas, de mesa o bien imaginati-

vos, donde la tecnología no cobraba ningún papel. Es por esto que la inclusión a las nuevas tecnologías de las personas mayores requiere un mayor esfuerzo y la necesidad de un atractivo diferente de los estímulos necesarios en los jóvenes [1]. La pregunta que cabría hacer entonces es: ¿Cuáles son las características idóneas para un juego al que jueguen personas mayores? Si bien es una pregunta compleja, a lo largo de la historia de la industria del videojuego, cabe destacar que los primeros juegos consistían en juegos sencillos, fáciles de utilizar y haciendo en muchos casos uso de juegos tradicionales trasladado a las nuevas tecnologías de forma sencilla.

Es por este motivo que las principales características deseadas son una temática o juego que resulte familiar al usuario, que sea fácil de comprender mediante una interfaz de un móvil y sencillo de aprender, ya que el más mínimo esfuerzo es motivo de rechazo por este sector. Además, diferentes estudios indican que las personas mayores prefieren jugar en dispositivos sencillos como puede ser un móvil con una interfaz simple, a juegos muy desarrollados y con las características más punteras.

Por otro lado, diferentes estudios demuestran las ventajas de jugar a videojuegos tanto por parte de los jóvenes como de los más mayores, a pesar de tener diferentes enfoques respecto a los mismos. En los jóvenes por ejemplo se desarrollan habilidades a través de la diversión como principal objetivo, mientras que en las personas más mayores el enfoque es diferente, ya que es una herramienta o bien de aprendizaje o bien de evasión de las obligaciones o situaciones diarias [2].

Además, existen múltiples beneficios en el empleo de videojuegos en las personas mayores, ya que estos no solo mantienen activas determinadas zonas del hipocampo, mantienen una vida cerebral activa, sino que también generan una mayor retención de memoria y capacidades cognitivas, a la vez que permiten que los mayores se sientan más satisfechos y genera una sensación de bienestar [3].

## 2.2. Juego social

El juego social se define en la industria del videojuego como un tipo de videojuegos que se juegan de forma social en redes sociales, o que requieren de la interacción de varios jugadores para colaborar o competir con una serie de objetivos a alcanzar, comunes o individuales. Este tipo de juego enfatiza el papel de la plataforma en la que se distribuye y la comunicación entre los diferentes individuos que participan. Por lo tanto se pueden destacar tres conceptos principales:

- Son juegos en línea.
- Hacen uso de la red social del jugador, o pretenden crear una red nueva.
- Hace uso de los servicios de redes sociales, apoyando su uso cultural.



Imagen 1: Juego Farmville: Colaborativo

Estos juegos se han vuelto muy populares de la mano de las diferentes mejoras tecnológicas con el paso de los años, aproximándose cada vez más a una serie de experiencias comunicativas a través de plataformas multijugador y con fines sociales. Estos juegos buscan como fin principal la socialización y entretenimiento de los diferentes individuos que lo componen, haciendo uso de las interacciones humanas.



Imagen 2: Juego Grepolis: Competitivo

Por lo tanto cabe destacar que un juego social, al depender directamente de los individuos que lo jueguen, debe estar orientado a determinados factores que engloben un posible conjuntos de individuos interesados, con rasgos similares como pueden ser la edad, el género, personalidad o preferencias de juego, entre otros. Además cabe destacar que influyen factores como la extroversión, ya que los individuos más extrovertidos tienden a socializar más, y los más introvertidos tienden a socializar en entornos que implican una interacción menos directa.



Imagen 3: Juego social Habbo

Por otro lado, los juegos online cobran un papel bastante interesante dentro de la educación, ya que el tiempo que una persona dedica a jugar a juegos digitales e identificarse con los personajes puede ser aprovechable para explorar nuevas condiciones, situaciones y toma de decisiones en las que ofrecer un aprendizaje no directo sobre los usuarios [4]. Los juegos educativos deberían superar la barrera de resultar lo suficientemente interesantes

para involucrar a los jugadores en base a las características del público destino, pero se puede observar de que se trata de una herramienta innovadora, motivadora y divertida [5].



Imagen 4: Juego World of Warcraft: Competitividad y colaboración

Se han considerado los principios básicos de la teoría de la actividad, que parece ser un fundamento teórico sólido para una valoración de los juegos educativos, en conjunto con la principal tendencia en los videojuegos multijugador, para proponer el desarrollo de juegos educativos multijugador en línea, que crearían comunidades de jugadores que aprenderían a través de una experiencia social. Esto solo puede lograrse si se tienen en cuenta una serie de factores y se aprovecha tanto la experiencia técnica y la creatividad de la industria informática como los conocimientos aportados por las teorías y la investigación educativas.

### 2.3. Juegos para mayores

Tal y como se comentaba, el sector deseado es el de las personas mayores, las cuales suelen ir asociadas a una pérdida de la actividad, siendo este un enfoque equivocado, ya que los juegos colaborativos les mantienen activos física y mentalmente. Además de los beneficios previamente mencionados, es importante conocer qué tipo de juegos son utilizados por el sector de los grupos de animación para personas mayores en grupos o en residencias.

Dentro de los tipos de juegos existentes, nos centraremos a continuación en aquellos que poseen una mejor aceptación entre las personas mayores [6] [7]:

- **Juegos de mesa:** Las actividades más comunes entre las personas mayores suelen ser juegos de habilidad como las cartas, bingo, dominó, etc.
- **Veo veo:** Al igual que con los más pequeños, este juego es aceptado ampliamente entre personas mayores ya que implica un punto imaginativo y de investigación.
- **Juegos de memoria:** Tal como cartas, asociativos a ideas, etc.
- **Simón dice:** Juego sencillo de seguir una serie de normas que sigue un moderador común, donde todos los jugadores realizan una serie de tareas en mismas condiciones.
- **Palabras encadenadas:** Juego clásico consistente en encadenar palabras mediante las sílabas que estas las componen creando una cadena infinita de palabras.
- **Puzzles:** Ya sean juegos sencillos de puzzles o rompecabezas, que implican un determinado razonamiento por parte del juego y puede conllevar una parte colaborativa.
- **Cada oveja con su pareja:** Consistente en asociar dos conceptos diferentes en base a una misma idea, pudiendo ser colaborativo entre varios enfoques de distintos jugadores.
- **Adivinanzas:** Se basan en el ingenio y los dobles sentidos, los cuales implican un razonamiento, pudiendo ser este una exposición de ideas colaborativas.
- **¿Quién es quién?**: Consistente en adivinar una persona en base a una serie de preguntas o descripciones.
- **Crucigramas:** Consiste en la asociación de descripciones y términos a palabras.

Como se puede comprobar, son juegos realmente sencillos que radica su éxito en su simplicidad y capacidad de hacer razonar a los diferentes jugadores, conllevando a veces un razonamiento colaborativo y exposición de ideas, lo cual produce una interacción entre los jugadores y permite establecer o fortalecer relaciones sociales.

Por otro lado, en el escenario de las personas mayores entran factores importantes como la actividad, socialización, individualización, situación personal y otras características propias de este grupo [8]. Se deben diseñar por lo tanto de forma especializada tantos los objetivos, como contenidos y metodología orientados a este sector mediante una evaluación posterior que determine tanto los beneficios obtenidos, como la utilidad de estas herramientas.

Las actividades lúdicas entre las personas mayores en su mayoría es tiempo de ocio, por lo que cabe realizar un estudio sobre estas personas y realizar un análisis de las necesidades, características personales e inquietudes que se puedan poseer en estos grupos [8] [9].

Por último cabría destacar la importancia de los problemas frecuentes que se encuentran en diferentes análisis realizados para determinar el diseño, análisis y evaluación de utilización de juegos por mayores [10]. En su mayoría, la mayoría de los problemas encontrados son problemas de visión, comprensión o deterioros de sentidos como la vista o el oído, por lo que realmente la dificultad residiría en encontrar un entorno que les pueda resultar cómodo y comprensible sin necesidad de un esfuerzo excesivo.

Como cabe esperar, la mayoría de las dificultades encontradas por los usuarios conlleven un mayor desinterés tanto por los juegos propuestos como por la utilización de nuevas tecnologías como herramientas de ocio, por lo que evitar estos posibles problemas es una tarea primordial en el diseño y elección de herramientas de cara a diseñar y construir proyectos lúdicos para este sector.

## 2.4. Chatbots

Los chatbots son máquinas que permiten la comunicación con personas de forma que estas interactúan con el objetivo de obtener un determinado fin o servicio, lo cual se desarrolla partiendo del concepto de la interacción hombre-máquina (*HCI*) [11]. La automatización de los procedimientos que se tratan mediante un agente conversacional obtiene numerosos beneficios, pero no consiste en una toma de decisiones triviales, puesto que el conocimiento sobre el dominio del problema es uno de los factores más determinantes.

Dentro de este ámbito, se distinguen dos grandes tipos dentro de lo que son los chatbots, en los que se encuentran los chatbots basados en reglas y los chatbots desarrollados mediante inteligencia artificial, ambos con sus propios entornos de trabajo y sus propias características. Para poder comprender ambos tipos es conveniente explicar en más detalle las diferencias entre los mismos [12]:

- **Chatbots basados en reglas:** Son agentes conversacionales con una amplia base de conocimiento sobre el dominio del problema o asunto. Estos agentes ofrecen una serie de respuestas predefinidas en base a las acciones del usuario de forma que se otorgue la mejor respuesta a cada interacción del usuario. Delimitan el escenario y fuerzan al usuario a seguir unas determinadas pautas o pasos para evitar que se desvíe del fin que pretende obtener, sin permitir realizar acciones ajenas a las que se contemplan en el escenario sobre el que se trabaja.
- **IA Chatbots:** A través de un motor de inferencia de información estos agentes son capaces de aprender y expresarse con un lenguaje más natural, dotando de mayor flexibilidad las posibles acciones del usuario, y aprendiendo de él, aunque esto realmente supone un riesgo, ya que un aprendizaje inadecuado de determinados comportamientos puede dar lugar a posibles fines diferentes del ideado.

Por otro lado, la aplicación de los chatbots en el sector de los videojuegos ha sido destacado desde su inicio mediante juegos de aventuras por texto y decisiones, como pueden ser los juegos desarrollados por *Infocom* o *Artic Computing*. Incluso en la actualidad, los videojuegos con comportamientos más realistas y que aplican tecnologías relacionadas con la IA poseen una serie de comportamientos predefinidos, compartiendo esta característica con los chatbots, por lo que realizar un juego sencillo no es complicado, y a su vez es sencillo de utilizar, tal y como se desea en el sector de las personas mayores.

Al mismo tiempo, existen diferentes formas de desarrollar un chatbot, ya sea mediante aplicaciones propias, interfaces o incluso en una de las formas más populares como son en la actualidad las propias aplicaciones de mensajería, con las cuales la familiarización es completa y el proceso de aprendizaje casi inexistente debido a la constante utilización que estas poseen en la actualidad [11]. Dentro de este trabajo se pretende enfocar el desarrollo de un videojuego de forma que sea amplio a todos los públicos en cuanto a utilización y sencillo de aprender, por lo que la utilización de una herramienta como son los chatbots es un punto de interés considerable y aplicable a dicho fin.

Considerando por lo tanto la descripción de un chatbot, se deben considerar por lo tanto una serie de características propias de los mismos, como son:

- Un chatbot está disponible 24/7.
- Un chatbot puede conversar con ilimitadas personas a la vez.
- La respuesta del chatbot siempre es inmediata.
- La respuesta del chatbot siempre es correcta, en el sentido de que responda exactamente lo que la empresa quiere que responda, sin despistes o confusiones que a veces tenemos los humanos.
- Abre otro canal de consumo de información y de suscripción por parte de clientes.
- Supone un ahorro de tiempo para el personal encargado de conversaciones repetitivas, eliminando además la frustración que suelen provocar este tipo de tareas.

## 2.5. Plataformas

Dentro de los diferentes servicios de mensajería instantánea se puede comprobar que los servicios principales son Telegram [Im. 5], WhatsApp [Im. 6] y Facebook Messenger [Im. 7] [13, 14]. Aunque WhatsApp es más utilizado en la actualidad, es cierto que hasta el año 2020 no implementó entre sus funcionalidades los chatbots, mientras que Telegram en ese sentido lleva siendo puntero desde casi sus comienzos, y la alternativa de Facebook es interesante debido al alcance que esta ofrece y su integración con un chat sencillo como Messenger. Por otro lado, aunque las tecnologías a la hora de diseñar un bot son similares, es cierto que WhatsApp es una plataforma más restringida a la hora de permitir el diseño e implementación de bots, mediante su API propia la cual requiere darse de alta como un proveedor de soluciones y pasar una serie de filtros [14].

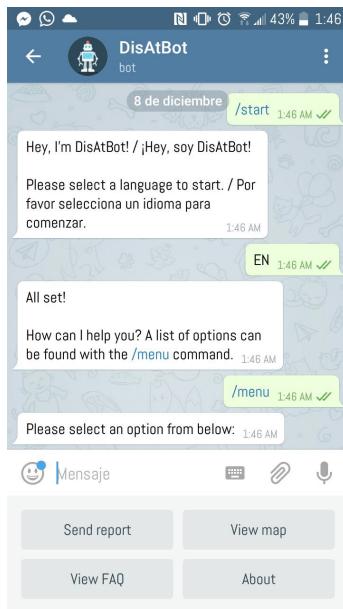


Imagen 5: Chatbot Telegram

Por otro lado Telegram presenta una mayor facilidad en la inclusión natural de bots dentro de la aplicación, sin requisitos adicionales ni filtros que dificultan la realización y desarrollo de los bots en la aplicación, por lo que es una plataforma ideal a la hora de desarrollar un chatbot con el objetivo de ser fácilmente utilizado por personas mayores en el propio servicio de mensajería instantánea. Por último, Facebook es una opción realmente interesante ya que utiliza estándares más parecidos a los de Telegram, pero mantiene una interfaz sencilla también y una accesibilidad similar [13]. Si bien son opciones menos populares entre la gente mayor, al poseer una interfaz prácticamente idéntica a WhatsApp, este aspecto quizás quede en un segundo plano, frente al amplio abanico de posibilidades que ofrecen estas opciones de cara al desarrollo, implantación y prueba del bot diseñado.



Imagen 6: Chatbot WhatsApp

Dentro del diseño de diferentes chatbots, Telegram ofrece la posibilidad de crear chatbots abiertos o dirigidos, que interactúen con el usuario de la forma más razonable según el contexto o el fin que se pretende alcanzar, ya sea de forma implícita o explícita. Además, la comunidad de desarrollo de bots en Telegram es inmensamente superior en la actualidad frente a la de WhatsApp o Facebook, aunque se pudieran implementar en todos estos servicios. Esto se debe al soporte que ofrece cada aplicación y a las herramientas proporcionadas por cada uno de los servicios, además de por el tiempo que lleva existiendo esta tecnología en cada una de estas plataformas. El debate se encontraría principalmente entre usar Telegram o Facebook Messenger, pero considerando la facilidad de instalación y darse de alta, y el público objetivo, Facebook puede suponer una sobrecarga de información en el usuario, y por lo tanto saturarle, mientras que Telegram se da de alta como WhatsApp, por lo que se escogerá Telegram como la opción más idónea.

Por otro lado, existe otra gran alternativa como es el uso de asistentes virtuales como podrían ser Alexa, o Google Home, entre otros. Si bien es una alternativa que permitiría una interacción más natural con el usuario mediante control por voz mediante *skills* o aplicaciones, esta opción puede llegar a resultar un poco estresante y causar rechazo en su inicio al usuario, ya que muchos de ellos consideran que el aprendizaje es una tarea innecesaria si desean buscar un entretenimiento, más aún con nuevas tecnologías, que requieren una curva de aprendizaje superior [15, 16, 17].

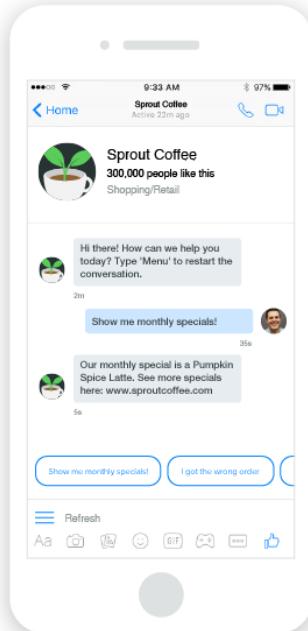


Imagen 7: Chatbot Facebook Messenger

En la actualidad existen numerosos juegos que pretenden socializar y ofrecer un entretenimiento de diversas maneras, aunque nuevamente suelen estar más enfocados en un público más joven, por lo que el objetivo de este trabajo será ofrecer una alternativa para personas más mayores tratando de no excluir ningún tipo de público.

## 2.6. Herramientas

A la hora de diseñar, desarrollar e implementar un agente conversacional, existen una infinidad de herramientas que permiten realizar estas funciones, de forma que se puede desarrollar un chatbot a la medida de las necesidades requeridas para el mismo. Por otro lado, existen ya diferentes herramientas denominadas *builders* que permiten automatizar este procedimiento de una forma sencilla e intuitiva.

Dentro de las diferentes herramientas se distinguen entre diferentes grupos [18] según si se tratan de frameworks oficiales, generales o herramientas de apoyo. Dentro de los frameworks oficiales, la mayoría de sistemas de mensajería incluyen los suyos propios como Telegram [Im. 8], Whatsapp [Im. 9], Facebook Messenger [Im. 10] o Microsoft [Im. 11].

### Telegram Bot API

The Bot API is an HTTP-based interface created for developers keen on building bots for Telegram.  
To learn how to create and set up a bot, please consult our [Introduction to Bots](#) and [Bot FAQ](#).

### Recent changes

Subscribe to [@BotNews](#) to be the first to know about the latest updates and join the discussion in [@BotTalk](#)

November 4, 2020

Introducing Bot API 5.0

Run Your Own Bot API Server

- Bot API source code is now available at [telegram-bot-api](#). You can now run your own Bot API server locally, boosting your bots' performance ([check this out](#) to see if this will benefit your project).
- Added the method `logOut`, which can be used to log out from the cloud Bot API server before launching your bot locally. You **must** log out the bot before running it locally, otherwise there is no guarantee that the bot will receive all updates.
- Added the method `close`, which can be used to close the bot instance before moving it from one local server to another.

Transfer Bot Ownership

- You can now use [@BotFather](#) to transfer your existing bots to another Telegram account.

Imagen 8: Telegram Bot API



Imagen 9: WhatsApp Business API

**Plataforma de Messenger**

- Introducción
- Primeros pasos
- Mensajes
- Webhooks
- Vistas web
- Sugerencias y nueva interacción
- Identificadores y perfil
- Procesamiento de lenguajes naturales
- Analytics & Feedback
- ¡Envía tu bot!
- Política y normas de uso
- Referencia
- Recursos útiles
- PREGUNTAS FRECUENTES
- Registro de cambios

Se actualizó este documento.  
La traducción en español no está disponible todavía.  
Actualización del documento en inglés: 13 oct. 2020  
Actualización del documento en español: 6 nov. 2018

[Volver al documento en español](#)

**Introduction**  
Learn basics and best practices [Learn More](#)

**Getting Started**  
Build your first Messenger bot fast [Start Building](#)

**Platform Features**  
Great Tools for Building Awesome Experiences

<b>Messaging</b> Send and receive text, media, structured templates, and so much more	<b>Webview</b> Build web-based experiences with the dev tools and frameworks you already love
<b>Discovery</b> Reach new people and re-engage ones you know on Messenger, Facebook, and the web	<b>IDs &amp; Profile</b> Personalize conversations, link with your existing auth, and create unified experiences

Imagen 10: Facebook Messenger Developer API

**Microsoft**

My Bots Documentation Blog Emulator LUIS QnA Maker Sign in

**Microsoft Bot Framework**

A comprehensive framework for building enterprise-grade conversational AI experiences.

[Try Azure Bot Service for Free](#) [Download SDK from Github](#)

Customers Cognitive Services Bot Life Cycle Quick Starts

**AI and natural language**  
Create a bot with the ability to speak, listen, understand, and learn from your users with Azure Cognitive Services.

**Open & Extensible**  
Get started with open source SDK and tools to build, test, and connect bots that interact naturally with users, wherever they are.

**Enterprise-grade solutions**  
Build secure, global, scalable solutions that integrate with your existing IT ecosystem.

**Ownership and control**  
Create an AI experience that extends your brand and keeps you in control of your own data.

Imagen 11: Microsoft Bot API

Por otro lado, existen determinados SDK no oficiales, o generales que sirven como herramientas de desarrollo especializadas en la construcción de chatbots independientemente de la plataforma [19, 20, 21]:

- **Chatfuel:** Es un *builder* gratuito que permite la creación de chatbots para Facebook Messenger sin requerir experiencia previa programando. Actualmente se utiliza en diferentes plataformas y empresas como la NBA o Forbes [Im. 12].

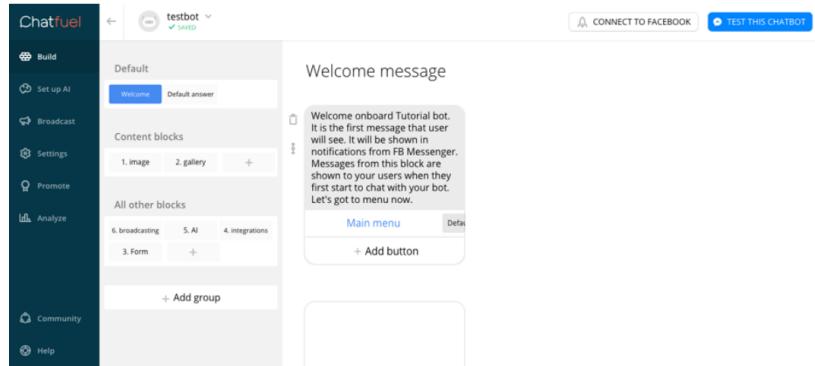


Imagen 12: Chatfuel Platform

- **Botsify:** Permite crear chatbots web o para Facebook Messenger sin conocimientos de programación previos. Posee una interfaz sencilla consistente en *drag and drop*, en la que se arrastran los elementos conformando un comportamiento. Es una plataforma reconocida utilizada por grandes empresas como Apple o Shazam [Im. 13].

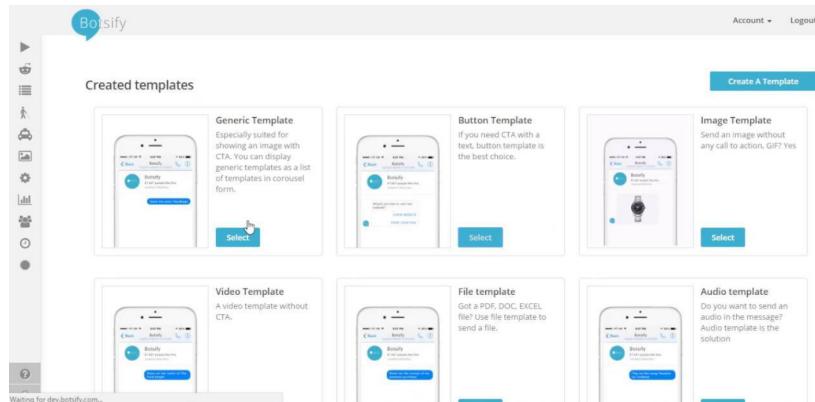


Imagen 13: Botsify Platform

- **MobileMonkey**: Es una plataforma reconocida para el diseño de chatbots en Facebook Messenger, la cual se encuentra orientada para el diseño de bots con fines comerciales y de crecimiento de audiencia. Estos chatbots permiten responder preguntas, seguir estados de compras y búsqueda de contenidos mediante palabras clave [Im. 14].

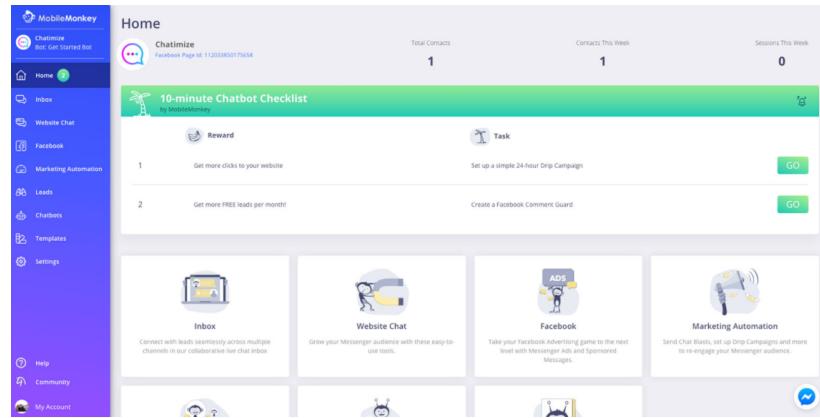


Imagen 14: MobileMonkey Platform

- **HubSpot**: Al igual que muchos de sus competidores, no requiere experiencia previa con la programación, y permite el diseño de chatbots con fines comerciales y personalización de mensajes basados en la información obtenida de los contactos del usuario. Esta plataforma es completamente gratuita [Im. 15].

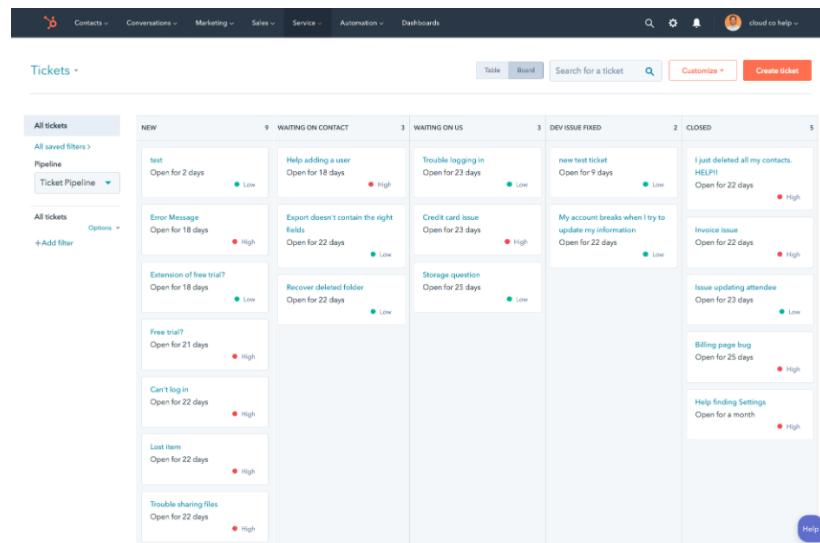


Imagen 15: HubSpot Platform

- **Dialogflow:** Perteneciente a Google, permite una gran escalabilidad y se encuentra hosteado en la plataforma Google Cloud, por lo que se trata de una de las plataformas con mayor escalabilidad. Además posee un soporte de una gran cantidad de lenguajes. Es una plataforma que permite el soporte de una gran cantidad de plataformas entre las que se incluyen Facebook Messenger, Slack o Telegram entre otras. Posee diferentes planes, pero existe la posibilidad de utilizar un plan gratuito [Im. 16].

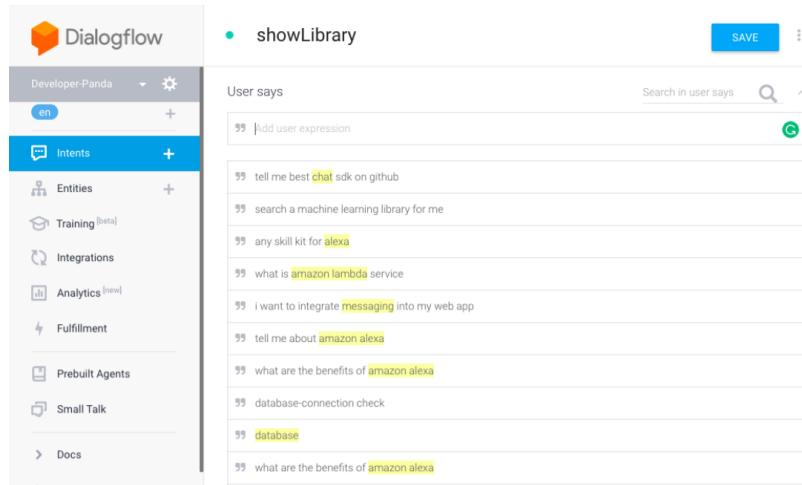


Imagen 16: Dialogflow Platform

- **Pandorabots:** Es una plataforma que está orientada a la interacción humana, y sin embargo requiere de cierto grado de experiencia a la hora de programar. Por lo tanto ofrece una gran grado de flexibilidad, siempre que se posea la experiencia suficiente. En cuanto a los planes, permite obtener una cantidad de 1.000 mensajes por mes gratis y se permite la construcción de hasta dos bots [Im. 17].

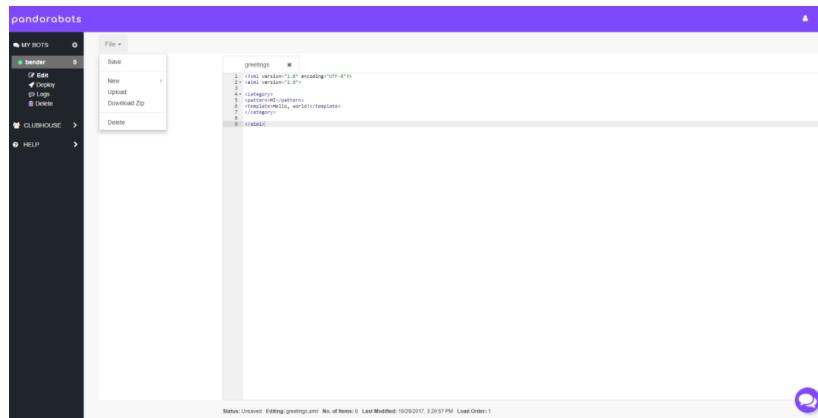


Imagen 17: Pandorabots Platform

## 2.7. Análisis del juego en los chatbots

A continuación el objetivo es ver diferentes aplicaciones de chatbots a la hora de diseñar diferentes juegos sociales, en los que se involucren diferentes jugadores. En este caso se distinguirán entre juegos desarrollados en aplicaciones y juegos desarrollados para plataformas de mensajería. En primera instancia se revisaran algunos juegos desarrollados directamente como una aplicación propia [22, 23].

- **Replika:** Si bien no es considerado un juego en ciertos aspectos, consiste en la creación de una réplica de nuestra personalidad mediante la realización de preguntas y actividades en conjunto. Evalua el comportamiento del usuario y la forma de interactuar de este, para realizar una réplica, obteniendo como resultado la conversación con uno mismo [Im. 18].



Imagen 18: Replika App

- **Mydol:** Consiste en un simulador que emula una conversación de uno mismo con una celebridad, simulando como esta interactuaría en el supuesto caso de que se tratase de una conversación real con el usuario, donde se entabla una relación de amistad y supone un entretenimiento diferente mediante la conversación [Im. 19].

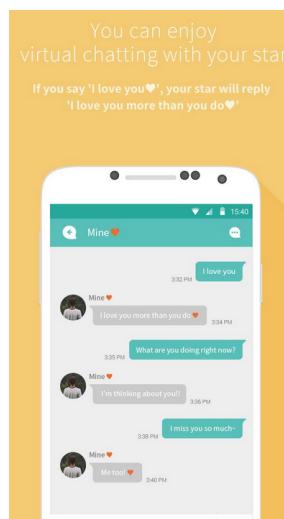


Imagen 19: Mydol App

- **Akinator:** Más conocida que las anteriores, consiste en un bot con forma de genio árabe que trata de adivinar un determinado personaje con preguntas sencillas de opciones sí o no. El objetivo es tratar de que el bot realizando preguntas sea capaz de adivinar el personaje que tiene el usuario en su mente, y en caso de no acertarlo reconocer su derrota [Im. 20].



Imagen 20: Akinator App

Si bien las aplicaciones anteriormente mencionadas tratan diferentes temáticas y tratan de simplificar los diferentes procedimientos que se llevan a cabo, existen otras alternativas como los denominados chatbots o bots de Telegram/WhatsApp [24], los cuales son bots que se definen como usuarios de la aplicación y que poseen su propio comportamiento al interactuar con ellos.

- **Werewolf:** Emula el conocido juego de *el hombre lobo*, el cual consiste en un reparto de personajes de una aldea en la que los diferentes usuarios realizan una serie de tareas con el fin de ganar la partida. Los hombres lobo han de matar al resto de jugadores y los jugadores mediante votaciones deben tratar de eliminar a los lobos sin saber quienes son, interviniendo diferentes personajes con habilidades [Im. 21].

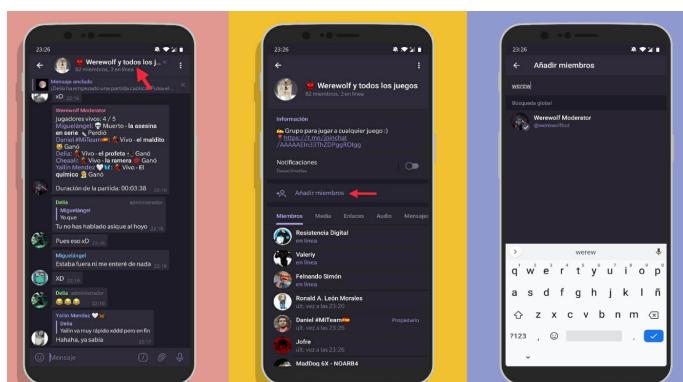


Imagen 21: Werewolf Telegram Bot

- **Quizarium:** Es una réplica de los clásicos juegos de mesa de preguntas y respuestas en los que se trata de poner a prueba el conocimiento general del usuario, utilizando para ello una serie de pistas y tiempos límites, con otras posibles opciones [Im. 22].



Imagen 22: Quizarium Telegram Bot

- **Pokerbot:** Básicamente traslada un juego de cartas como es el póker a un formato digital, donde indica al usuario las diferentes opciones e informa sobre el estado de la mesa virtual y acciones del resto de jugadores [Im. 23].

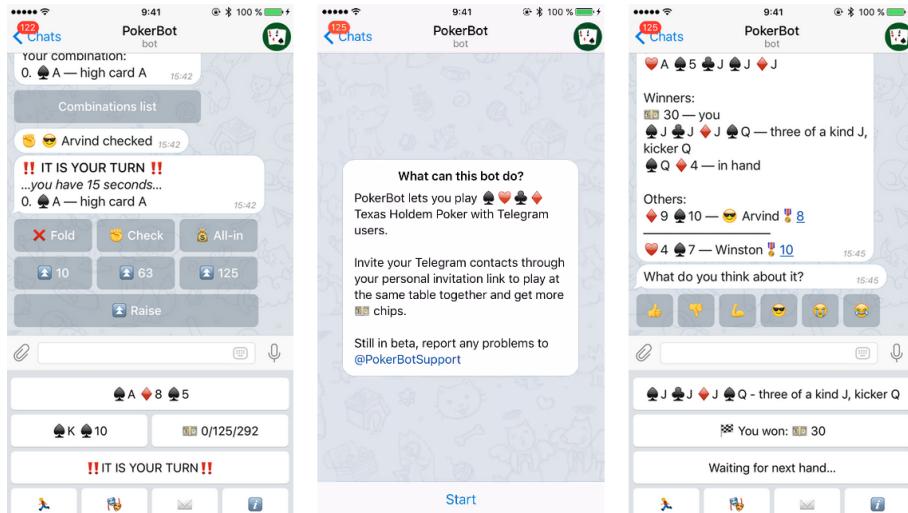


Imagen 23: Pokerbot Telegram Bot

Como se puede observar realmente existe un amplio abanico de ofertas, aunque estos no estén realmente orientados a personas mayores, existen juegos como el *Quizarium* o *Pokerbot* en los que la curva de aprendizaje requerida por el usuario es prácticamente mínima, por lo que la implementación de un determinado chatbot en una plataforma de mensajería instantánea es realmente una opción considerable para el fin de este trabajo.

## **2.8. Conclusiones**

En los diferentes trabajos e informaciones estudiadas se ha podido observar los diferentes beneficios que producen los videojuegos tanto en los jóvenes como en las personas mayores, centrándonos en estas últimas. Dentro de estos beneficios se destacan por lo tanto mantener a la persona activa, con una vida cerebral activa, retención de memoria, mejoras de las capacidades cognitivas y evitar su deterioro, generación de una sensación de bienestar, etc.

Se ha descubierto de las ventajas que tiene el videojuego social y el papel que puede cobrar tanto en la educación de un individuo como en la adquisición de competencias del mismo a través de un aprendizaje mediante el entretenimiento y como una herramienta de distracción y evasión de las preocupaciones diarias, permitiendo centrarse en tareas más divertidas para el jugador y centrando su atención.

Por otro lado, se han comentado diferentes enfoques de los posibles juegos y como un chatbot puede ser una gran herramienta por su sencillez para la realización de juegos sociales, que permitan mediante la comunicación de forma sencilla transmitir conceptos a los jugadores para que interactúen y generar un entretenimiento de forma que no suponga dificultad para el aprendizaje.

En cuanto a las diferentes plataformas observadas, es obvio que WhatsApp es la plataforma preferida de la mayoría de los usuarios en España, sin embargo, Telegram permite una mayor flexibilidad y gestión de los chatbots de forma que se integren fácilmente y se puedan comunicar con el usuario de una forma más natural, por lo que resulta la plataforma ideal de cara a desarrollar un proyecto de juegos sociales mediante chatbots para personas mayores.

Por último, cabe destacar algunas de las posibles mejoras o campos de estudios observados en la investigación, como puede ser la interacción de personas mayores con chatbots, las ventajas que posee la utilización de un dispositivo móvil como medio lúdico y educativo, el enfoque de la industria del videojuego hacia las personas mayores o incluso los beneficios que posee la utilización de asistentes virtuales en el entretenimiento de personas mayores.

### 3. Análisis inicial del problema.

Tras analizar el estado del arte, y comentar los principios y herramientas disponibles para poder desarrollar un juego mediante tecnología de chatbots, la idea es realizar un juego social, intergeneracional y que sea sencillo para poder atraer también al sector de las personas mayores. Considerando estos tres principales aspectos, hay que considerar un juego que a la vez que sencillo sea atractivo para todo tipo de edades y haga partícipes a ambos.

Por otro lado, otra de las cuestiones importantes que se pretenden trabajar en este trabajo es la aportación no solo del entretenimiento sino de una serie de retos que permitan ofrecer una serie de beneficios cognitivos a los jugadores, estimulando diferentes tipos de inteligencias en los jugadores y ayudando a mantener activo el razonamiento, especialmente de las personas mayores.

Tras este breve análisis el juego a desarrollar será por lo tanto un juego de preguntas, que combine diferentes temáticas, tomando como ejemplo juegos del tipo *Trivial Pursuit*, *Preguntados* o *¿Quién quiere ser Millonario?*.

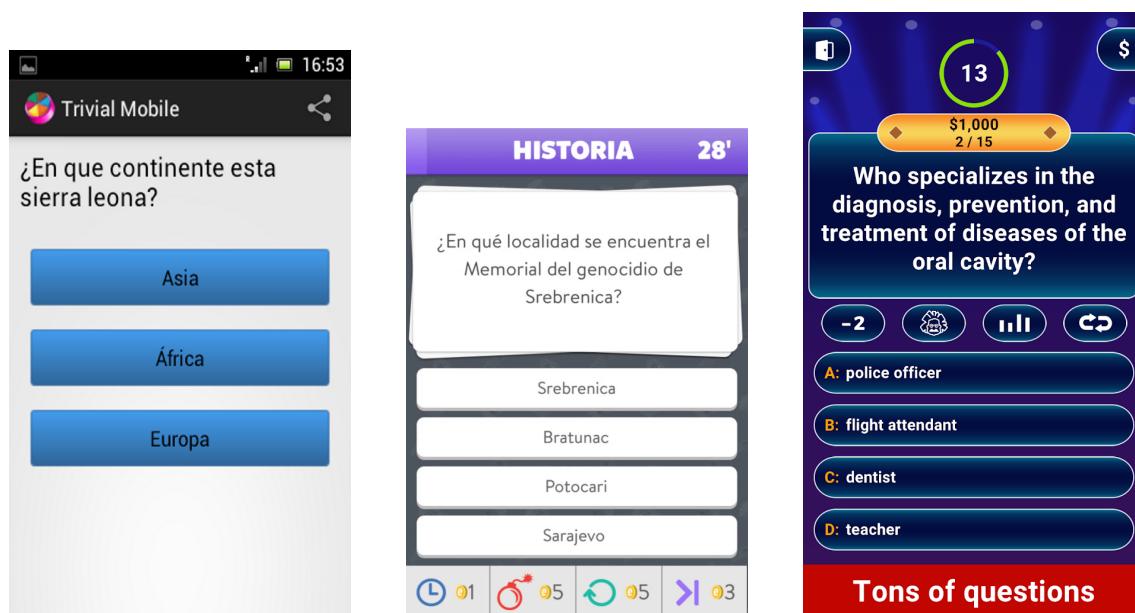
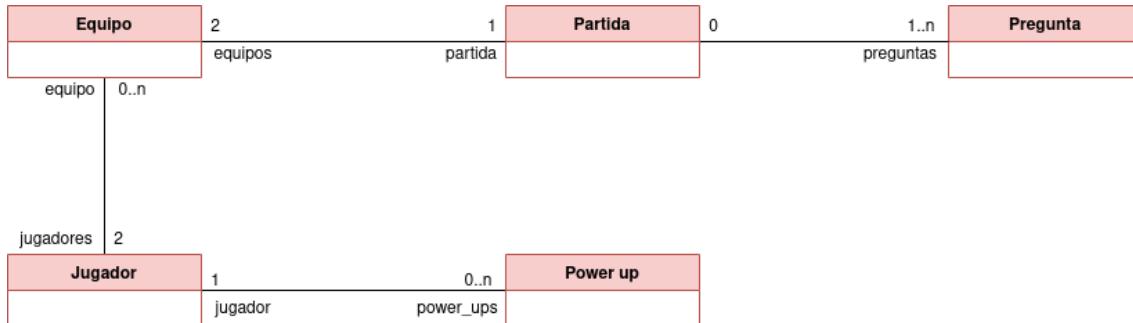


Imagen 24: Trivial Pursuit, Preguntados y Millonario

Por lo tanto la solución ofrecida será un juego de preguntas y respuestas, pero con el fin de añadir una serie de novedades que impulsen a los jugadores a participar, puedan atraer a un público intergeneracional y resulte motivador para las personas mayores, se proponen a continuación una serie de características especiales:

- **Competitividad:** Siendo un juego de preguntas y conocimiento general, con el fin de motivar a superarse, se realizará una competición entre dos bandos, el juego se realizará considerando dos equipos.
- **Colaboración:** Resulta más motivador para las personas mayores cuando se realizan tareas de forma colaborativa y se pueden compartir con otras personas, por lo que en vez de jugar individualmente, se realizará mediante parejas.
- **Intergeneracional:** Independientemente de las diferentes temáticas que se tratarán en el juego, uno de los objetivos es conseguir que tanto el público más joven como el más mayor se sientan valorados e involucrados en el juego, por lo que las preguntas deberán estar orientadas para todos los públicos y a su vez deberán existir preguntas orientadas a las diferentes generaciones con el fin de valorar el conocimiento de cada sector y fomentar la colaboración.
- **Ausencia de tiempo límite de respuesta:** Aunque en un principio podría parecer un problema, está comprobado que las personas mayores responden mejor a juegos en los que no encuentra una gran presión o no se ven obligados a responder bajo un tiempo límite. Con esta característica los mayores se pueden ver liberados de esa presión y podrán jugar cuando quieran, aunque duren las partidas días.
- **Logros:** Otro de los puntos importantes es el de poder obtener un sistema de logros que motive a los jugadores a jugar y llegar más lejos. Además estos logros se traducirán a su vez en un sistema de ventajas o *power ups*.
- **Notificación por ausencia:** Otra de las ideas propuestas para mejorar la comunicación en el juego, es en caso de una ausencia prolongada del jugador, si el resto de jugadores así lo desean, poder notificarle por correo electrónico, de forma que si el jugador lleva más de 24 horas sin jugar sea notificado personalmente en el correo, y en caso de llevar más de 48 horas conceder la victoria al rival.

Un posible diagrama de conceptos inicial sería:



En este diagrama se remarcán una serie de conceptos principales que se engloban bajo clases donde se especifica en un principio la siguiente información:

- **Pregunta:** Las preguntas poseerán su correspondiente enunciado con cuatro respuestas posibles, y a su vez la respuesta correcta a dicha pregunta. Es una entidad sencilla.
- **Jugador:** Es una entidad con sus propios atributos como nick de usuario, nombre, edad y correo electrónico. A su vez esta entidad contiene una serie de **power ups** que se explican a continuación.
- **Power up:** Existirán tres tipos diferentes (50 %, pasar turno o cambiar pregunta) de power ups que contendrán la cantidad disponible para ser usados por el jugador asociado.
- **Equipo:** Una entidad con nombre o identificador que reúne a dos jugadores en cada equipo y recoge las puntuaciones del mismo de forma unificada.
- **Partida:** Es la entidad sobre la que gira todo, contiene un conjunto de preguntas y dos equipos y lleva a su cargo toda la mecánica asociada al juego de forma clara y unificada.

El objetivo será mediante un controlador de la lógica de la aplicación recoger toda esta información y procesarla haciendo uso finalmente de la API propia de Telegram, como se explicará en las posteriores secciones.

Toda esta información será almacenada y manejada, y la responsabilidad final recaerá sobre el controlador y la API con la que se comuniquen los usuarios para participar en el juego desarrollado. Este esquema se detallará en mayor medida en las diferentes entregas que se realicen en este proyecto.

Una idea inicial de como se vería la conversación sería la siguiente, para las principales partes del juego:

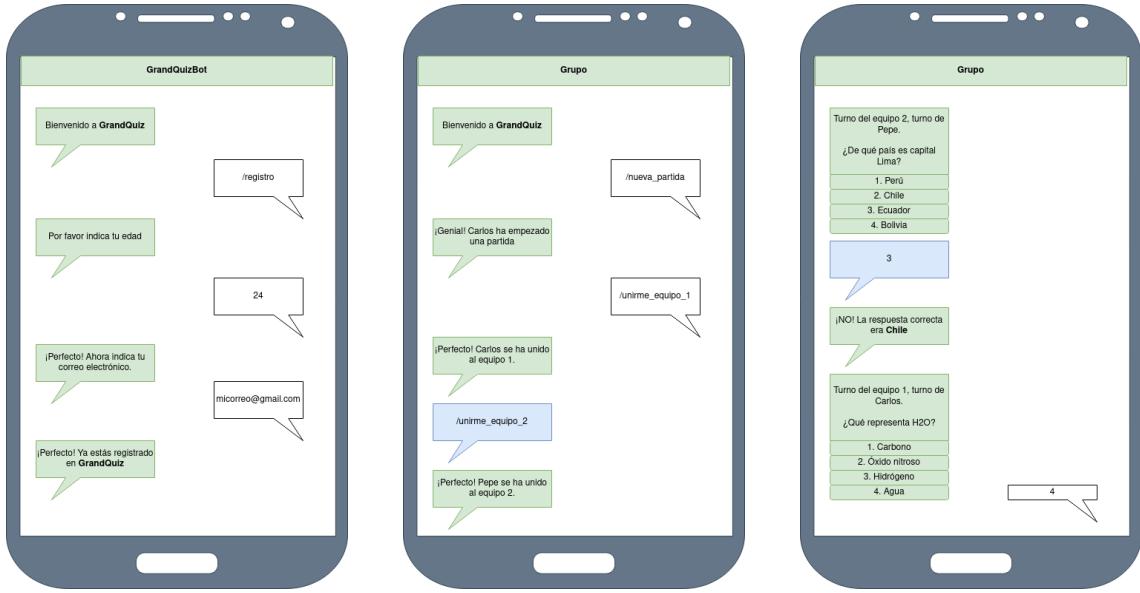


Imagen 25: Registro, Creación de partida y Juego

Como se puede observar, en la primera captura se ve un registro que se realiza de forma sencilla, aunque más adelante se valoraría un registro más intuitivo que incluya botones. En un principio la información que se contempla es la edad y su correo electrónico para futuros usos en las organizaciones de partida y las notificaciones al usuario.

Por otro lado, la creación de una partida deberá ser con un comando sencillo e intuitivo, mientras que unirse a una partida debería ser una tarea también sencilla mediante un comando que indique explícitamente a que equipo unirse.

También se contemplan diferentes opciones como ofrecer una versión más visual o incluso integrar la inclusión en un equipo de forma sencilla mediante botones de forma que sea más intuitivo y menos complejo para el usuario.

Por último, en la parte relativa a la mecánica del juego, la idea sería ofrecer una pregunta con las posibles opciones al jugador y que este indicara de alguna forma (bien mediante texto o bien mediante botones de forma más intuitiva) la opción escogida.

A continuación se mostraría si el jugador ha acertado o no y se le irían notificando de sus progresos, a la vez que en todo momento se indica el turno del jugador actual para evitar confusiones y controlando posibles trampas.

Nuevamente toda esta interfaz se detallará más adelante en posteriores secciones.

### 3.1. Arquitectura

El proyecto, al tratarse de un juego desarrollado mediante una API, necesitará una determinada arquitectura que contemple los diferentes elementos y servicios que interactúan en la misma. Para ello se han de considerar los diferentes elementos que intervendrían en esta y como se comunican.

La arquitectura propuesta para el proyecto consiste en:

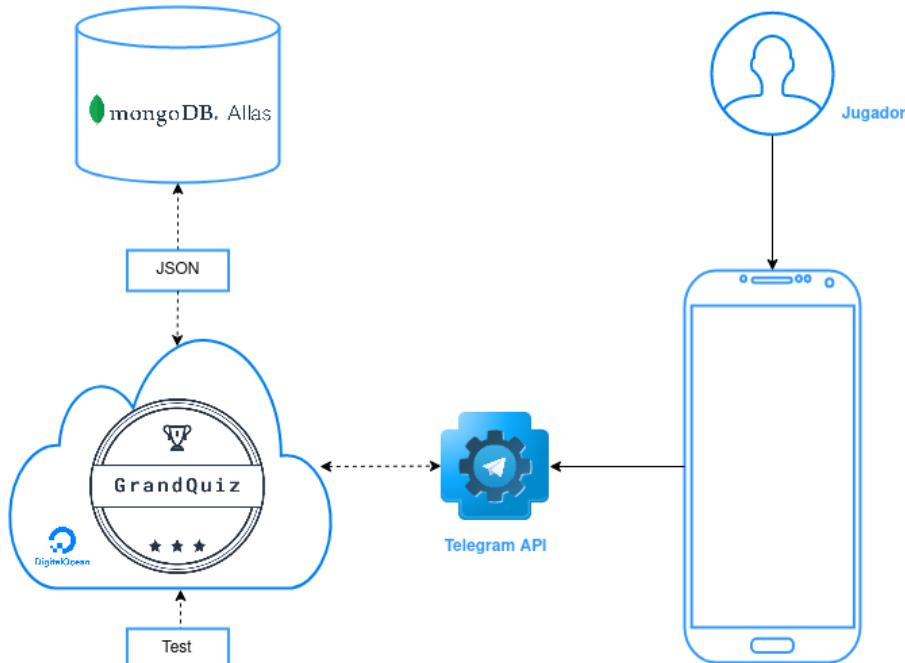


Imagen 26: Arquitectura de GrandQuiz

Dentro de la arquitectura se distinguen los principales elementos consistentes en:

- Una **API**, para la cual se utiliza la API oficial de **Telegram**, la cual permite definir flexiblemente con diferentes lenguajes de programación el comportamiento de la misma e interactuar con el sistema. Este API se comunica directamente con el dispositivo móvil (o cualquier otro dispositivo con Telegram) y permite una comunicación con el sistema del bot de forma flexible y personalizable.
- La lógica de la aplicación **GrandQuiz**, la cual deberá estar con una disponibilidad completa las 24 horas. Para este fin se utilizará un **droplet** de **Digital Ocean**, con el fin de desplegar el proyecto y la lógica del bot, el cual mediante la API previamente definida atenderá las peticiones.
- Un sistema de test sobre la aplicación, que permita ejecutar los tests tanto de forma local como con un sistema de Integración Continua.
- Un sistema de base de datos, donde almacenar la información de la aplicación, utilizando para ello un servicio con total disponibilidad, siendo este **MongoDB Atlas**, permitiendo una comunicación entre la base de datos y la aplicación mediante formato **JSON**.

#### 4. Tecnologías a usar.

El proyecto se compone de diferentes tecnologías y herramientas que se van a utilizar para las diferentes tareas que componen el desarrollo del juego. A continuación se detallan cada una de las herramientas que se utilizan:

#### 4.1. Gestor de proyecto

**Trello** es el gestor de proyectos que se utilizará para la organización del proyecto, permitiendo organizar mediante un sistema de organización por tarjetas o *Kanban*. Se configura como un tablero sencillo e intuitivo en el que se dispondrán las diferentes tareas y sus estados en base a su ámbito. Además permite establecer anotaciones, clasificaciones, fechas límites y un seguimiento de los mismos [Im. 26].

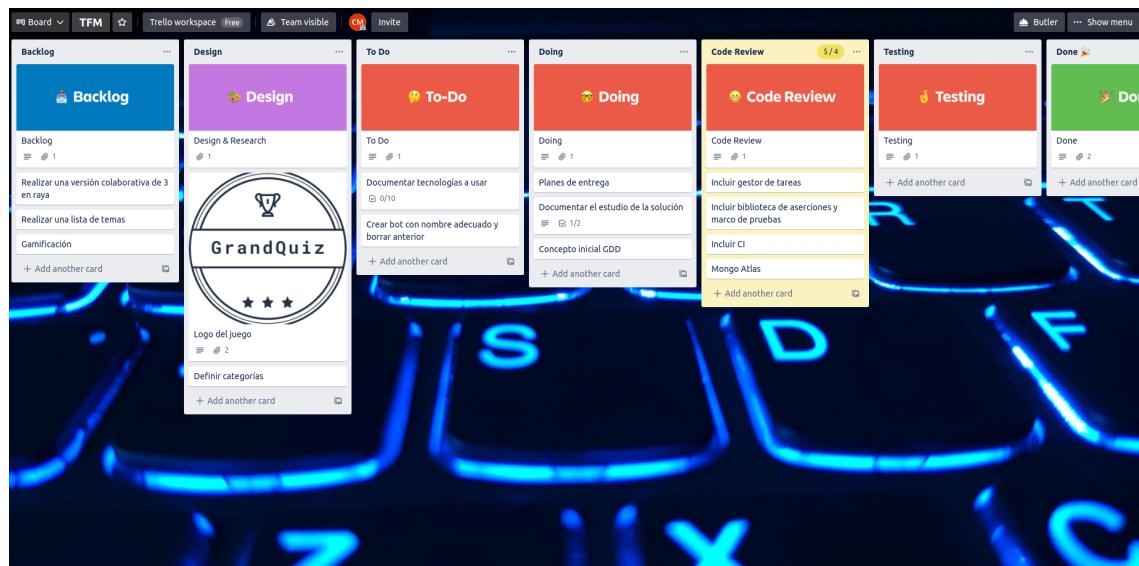


Imagen 27: Trello

## 4.2. Lenguaje de programación y APIs

#### 4.2.1. API chatbot

La **Telegram API** define una interfaz de programación sobre la que desarrollar diferentes bots. Esta API destaca por su libertad en el desarrollo ya que soporta diferentes lenguajes de programación. Debido a esta API es posible definir bots o usuarios autónomos que realizan tareas y responden a determinados comandos previamente programados. Por último cabe destacar que para desarrollar un bot es tan sencillo como solicitar un token al administrador propio de telegram conocido como *@BotFather* y la gestión del bot se realiza a través del mismo.

Para poder realizar un bot en **Telegram**, es necesario hablar inicialmente con **@BotFather** [Im. 27]. A continuación se le indica entre las disponibles opciones la de creación de

un nuevo bot con el comando `/newbot`. Una vez obtenemos el token de conexión a nuestro chatbot, es tan sencillo como definir el comportamiento con la herramienta que deseemos utilizando el token de acceso generado, el cual no se publicará para la seguridad del bot.

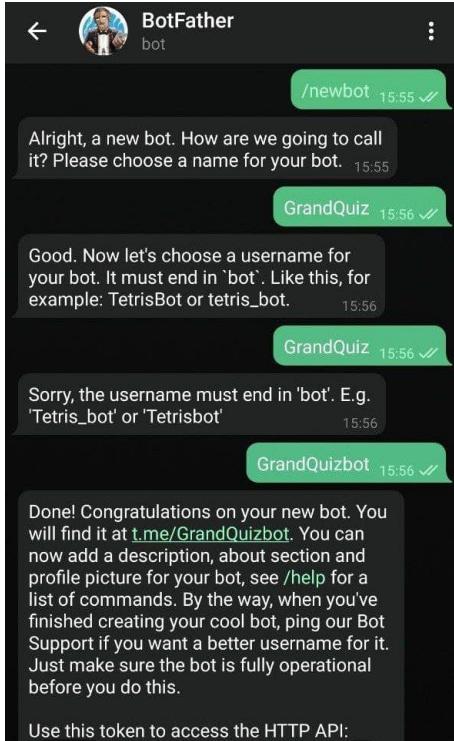


Imagen 28: Conversación con @BotFather

Por otro lado cabe destacar que todas las gestiones o modificaciones asociadas al bot (no a su comportamiento) se deben llevar a cabo a través de este bot de Telegram, ya que es la API que nos facilita la gestión de los mismos. Un ejemplo de posible modificación que se ha utilizado es el cambio de imagen de perfil.

#### 4.2.2. Lenguaje de programación

**Python** es un lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación. Se escoge principalmente por que se trata de un lenguaje ampliamente soportado por múltiples plataformas, por la enorme comunidad que lo soporta y porque es un lenguaje multiparadigma. Además existe una gran documentación sobre como realizar chatbots con Python.

#### 4.2.3. Librería de programación de API

**Telebot:** Se trata de una librería de Python la cual ofrece herramientas para poder definir la API de un chatbot de Telegram, permitiendo con decoradores definir las diferentes rutas, respuestas y comportamientos del bot.

Para comprender un poco, se muestra un tutorial que realiza las siguientes tareas:

1. **Contestar a un mensaje:** Esta funcionalidad es sencilla y consiste en indicar mediante un comando que se recoge e indicar la funcionalidad deseada dentro de una función que maneja el mensaje detectando dicho comando. A continuación podemos enviar un mensaje con **send\_message** o responder al mensaje con **reply\_to** siempre indicando el mensaje o el id del chat, obteniendo ambos del cuerpo del objeto **message**. Para ello definimos en el bot nuestro **message\_handler**.

```
# Comando start como saludo
@bot.message_handler(commands=['start'])
def saludar(message):
    bot.reply_to(message, f"¡Hola! Soy el Bot de tutorial,
                           bienvenido a Telebot.")
```

2. **Realizar preguntas mediante botones:** Esta funcionalidad es similar a la anterior, salvo porque podemos indicar **markups**, los cuales son objetos que contienen una serie de botones (**InlineKeyboardButton** en el caso de botones flotantes que no estén en el texto) y a través de ello indicar opciones y recoger **callbacks** al pulsarlos.

Un ejemplo de definición de un *markup* para escoger entre cuatro colores sería (en este caso recogido por modularidad en una función):

```
# Markup de colores
def get_markup():
    # Keyboard
    markup = types.InlineKeyboardMarkup(row_width = 1)
    # Buttons
    bt1 = (types.InlineKeyboardButton("rojo", callback_data="rojo"))
    bt2 = (types.InlineKeyboardButton("azul", callback_data="azul"))
    bt3 = (types.InlineKeyboardButton("verde", callback_data="verde"))
    bt4 = (types.InlineKeyboardButton("amarillo", callback_data="amarillo"))
    markup.add(bt1, bt2, bt3, bt4)

    return markup
```

A continuación, para utilizar el *markup* definido, podríamos preguntar por los colores al utilizar el comando */color* de la siguiente forma:

```
# Comando color para preguntar por el color favorito
@bot.message_handler(commands=['color'])
def preguntar_color(message):
    bot.send_message(message.chat.id, f"Por favor indicame tu color
                                         favorito de los siguientes:",
                                         reply_markup=get_markup())
```

3. **Obtener callbacks y procesarlos:** Este paso puede resultar ligeramente más complicado ya que no se manejan mensajes como tal sino **callbacks** del *markup* definido. Podemos obtener la información del *callback* directamente con **call.data** y procesarla, y a su vez si accedemos a **call.message** podríamos trabajar como si de un objeto *message* se tratara. En la función indicada al **callback\_query\_handler** podemos indicar una función con una comprobación para recoger solo determinados mensajes. En el siguiente ejemplo vemos como simplemente contestamos cuando nos indican el color escuchando el *callback* y modificamos el mensaje haciendo uso de **edit\_message\_text** para no crear un mensaje nuevo sino aprovechar el existente indicando el id del chat y del mensaje en particular:

```
# Funcion para recoger respuesta de color favorito
@bot.callback_query_handler(func=lambda call: True)
def escoger_color(call):
    if call.message:
        bot.edit_message_text(f"¡NO ME DIGAS! Mi color favorito tambien
                             es el {call.data}." ,
                             call.message.chat.id,
                             call.message.id,
                             parse_mode = 'Markdown')
```

4. **Recoger mensajes sin comandos ni callbacks:** Otra de las opciones más sencillas consiste en simplemente leer mensajes con el mismo **message\_handler** pero en lugar de indicar un comando como hemos visto previamente, aplicaríamos una función sobre el mensaje (por ejemplo una expresión regular). En el siguiente ejemplo simplemente repetimos lo que nos diga el usuario:

```
@bot.message_handler(func=lambda message: True)
def echo_all(message):
    user = message.from_user.first_name
    bot.send_message(message.chat.id, f"¡{user} ha dicho: '{message.text}'")
```

Por último, cabe indicar que existen una gran serie de funciones que podemos usar a nuestra conveniencia como puede ser enviar fotografías u otros elementos y procesar de diferentes maneras, el comportamiento general del bot a desarrollar en este trabajo se basará en la utilización de los puntos explicados previamente más el uso de algunas de estas ocasionalmente.

### 4.3. Gestor de tareas

Se ha escogido **Invoke** como gestor de tareas ya que permite realizar las diferentes tareas asociadas al proyecto de forma Pythonica y a su vez también es independiente de la plataforma en la que se ejecute. Posee un enfoque heredado de GNU Make y es dirigido por código, lo que permite una fácil definición y manipulación de tareas.

### 4.4. Test

#### 4.4.1. Marco de pruebas

**Pytest** se trata de una librería que actúa como framework para testing del código. Utilizar un marco de pruebas nos permite garantizar la calidad del código de forma au-

tomática y así poder obtener un desarrollo más fiable y robusto. Utilizar esta librería nos permite una mayor flexibilidad y además posee herramientas que se incorporan de forma natural con el código en Python.

#### 4.4.2. Biblioteca de aserciones

La biblioteca de aserciones escogida es **Assertpy**, la cual se trata de una librería de aserciones que permite obtener una serie de aserciones más complejas de forma sencillas con un lenguaje similar al lenguaje natural. Por otro lado, a diferencia de otras posibles bibliotecas con este fin, no tiene una sintaxis demasiado complicada y con demasiada verborrea, sino que posee un enfoque de estilo Python, que nos permite obtener un código más limpio y comprensible.

### 4.5. Control de versiones

Se ha escogido Git como sistema de control de versiones y concretamente **GitHub** como la plataforma de almacenamiento del repositorio asociado al proyecto, ya que posee la mayor comunidad frente a otros servicios de alojamiento y posee determinadas características como la gestión de issues, y el soporte de aplicaciones externas que hacen ideal este servicio para el proyecto.

### 4.6. Integración Continua

Otra de las metodologías útiles aprendidas ha sido la de utilizar un sistema de Integración Continua que permite determinar los fallos en el proyecto y determinar si se cumplen las condiciones de satisfacción sobre el proyecto.

Para ello se ha decidido utilizar las propias workflows de GitHub denominadas GitHub Actions ya que se trata de un servicio propio por lo que la configuración es sencilla y la sincronización también. Además, a diferencia de los sistemas de Integración Continua convencionales, no posee limitaciones de cara a un proyecto de Software libre como es este proyecto.

### 4.7. Almacenamiento

Para el servicio de almacenamiento y base de datos se ha decidido escoger **MongoDB Atlas** el cual es un servicio cloud de almacenamiento de una base de datos no relacional. MongoDB ofrece la posibilidad de configurar un proyecto de forma libre en su servicio, el cual se aprovechará para almacenar las diferentes preguntas y elementos del proyecto.

Las principales motivaciones de cara a escoger este servicio es la alta escalabilidad, el hecho de utilizar una base de datos no relacional ya que las diferentes preguntas tendrán información variable y resulta deseable este tipo de bases de datos. Por último la disponibilidad y gestión de la base de datos queda a responsabilidad del propio MongoDB, por lo que es un factor positivo de cara a la utilización de este servicio de almacenamiento.

```

_id: ObjectId("606dd14900d2b85156a65c65d")
nombre_usuario: "pepitocg"
nombre: "Pepe"
edad: "edad0"
email: "pepe@gmail.com"
avatar: "av3"

_id: ObjectId("606dd4fc223fd9aa18029a7c")
nombre_usuario: "Carlosma"
nombre: "Carlos"
edad: "edad0"
email: "carlosma@correo.ugr.es"
avatar: "av7"

```

Imagen 29: Visualización en MongoDB Atlas de una colección de jugadores

## 4.8. Contenerización

Con el objetivo de poder ofrecer un servicio de alta disponibilidad, se ha de realizar el despliegue del proyecto en un servicio cloud. Una de las metodologías aprendidas, ha sido la de realizar una contenerización del proyecto, permitiendo la implantación del mismo en una máquina proporcionada por un proveedor.

Con el fin de realizar una contenerización y que esta se encuentre en constante actualización respecto al desarrollo del proyecto en **GitHub**, se ha escogido **Docker** como herramienta para dicha labor.

Por otro lado, cabe destacar que se ha decidido escoger un repositorio de contenedores, como es **Docker Hub**, el cual no solo permite llevar un historial de las diferentes versiones, modificaciones y entregas del proyecto, sino que además permite sincronizar la creación de contenedores a los diferentes cambios que sufra el proyecto en el repositorio de **GitHub**. Finalmente la idea es utilizar **Docker** como herramienta de contenerización y **Docker Hub** como repositorio.

## 4.9. Despliegue

Para el despliegue del proyecto se han analizado diferentes propuestas de proveedores de servicios de IaaS y PaaS, pero tras analizar los requisitos y la naturaleza del proyecto, se ha decidido finalmente utilizar una implantación en un servicio de IaaS (Infraestructura como Servicio).

Entre las diferentes propuestas que se encuentran en el mercado, tras hacer un breve análisis, se ha decidido escoger **Digital Ocean** como proveedor, ya que sus **Droplet** son máquinas virtuales realmente sencillas de gestionar y que ofrecen todas las cualidades que se desean en el proyecto.

## 5. Metodologías a usar en el proyecto.

Dentro de la organización del proyecto, la elaboración del mismo se realiza por una única persona, siendo esta el alumno autor del proyecto, por lo que para la realización del proyecto se aplicarán diferentes metodologías ágiles que permiten realizar un proyecto de calidad, adaptativo y con constantes revisiones.

Entre las características que se destacan se encuentran:

- Una metodología rápida, específica al proyecto y que se comporta de forma dinámica, ya que no todos los requisitos del proyecto se encuentran definidos desde el inicio, y se producirán cambios durante la realización del mismo y es necesario un sistema adaptable.
- Se trabaja de forma constante con el cliente, siendo en este caso usuarios sin conocimiento alguno de la estructura del proyecto, pero interesados en la participación en las pruebas del mismo, siendo la comunicación clave en este procedimiento.
- Las diferentes acciones que se llevan a cabo en el proyecto siguen un formato ajustable y simple. Por lo tanto todas las decisiones tomadas serán lo más sencillas posibles intentando cubrir toda la estructura del proyecto y a su vez permitiendo una flexibilidad en el desarrollo del mismo.
- Se sigue un sistema de entregas tempranas y continuas, proporcionando un sistema que permita ser flexible y ofrecer un producto durante diferentes etapas, promoviendo la realización de un proyecto adaptable y con la posibilidad de ser probado.
- Se ofrece un sistema flexible laboralmente, y que no requiere de una planificación exacta, ya que aunque sigue un sistema programado de entregas, la realización de las tareas queda a la decisión según las necesidades de cada instante del desarrollo del proyecto.

Dentro de las diferentes metodologías ágiles existentes, para la realización del proyecto se utilizarán diversas metodologías ágiles, las cuales se explican a continuación.

### 5.1. SCRUM

Si bien es cierto que **SCRUM** es una metodología orientada a equipos de desarrollo, en este caso se utilizan determinados conceptos del mismo y se aplican en un perfil único y multidisciplinar que aplique dichos perfiles y reúna las tareas de los mismos, considerando las siguientes responsabilidades:

- **Product owner:** Este perfil es necesario para conocer la lógica del proyecto y establecer relaciones con el cliente (en este caso, usuarios que realizan las pruebas del proyecto).
- **Scrum master:** Encargado de que se apliquen las diferentes técnicas **SCRUM** por parte de la organización y ayuda en la adopción de esta metodología, siendo en este

caso del proyecto la función de comprensión y aplicación de las diferentes técnicas en el desarrollo del mismo.

- **Desarrollo:** Todo el desarrollo, organizado con el perfil de **Product Owner**, son realizadas por el autor del proyecto de forma que se comprendan las diferentes funcionalidades requeridas por el proyecto, pero siendo a su vez un sistema auto-organizativo en el que el desarrollo es flexible cumpliendo con los requisitos indicados previamente.

En cuanto a las principales herramientas empleadas se destacan:



Imagen 30: Backlog SCRUM

- Un **backlog de producto** consistente en un listado de tareas que engloba el proyecto, mediante una estimación de tiempo. Estas tareas deben ser realizadas y para ello previamente han de ser programadas, organizadas y ordenadas por el **Product Owner**.
- **Sprint backlog** siendo el conjunto de tareas obtenidas del **product backlog** escogidas en el **sprint planning** para generar un objetivo. Estas tareas son realizadas por el desarrollador de cara al siguiente **meeting**.

Finalmente cabe destacar que el uso de esta metodología permite una organización efectiva en la que se desarrolla un producto de forma rápida y con un mayor valor. Además esta planificación asegura menos imprevistos ya que se evalúa constantemente el proyecto y quedan claramente definidas las tareas que se realizan durante el mismo.

## 5.2. Kanban

Se aplica también la metodología **Kanban** con el fin de visualizar los diferentes flujos de trabajos que se aplican en diferentes etapas del desarrollo del proyecto, para las diferentes tareas indicadas previamente mediante metodología **SCRUM**. Estas dos metodologías se utilizan de forma complementaria con el objetivo de mantener una traza efectiva, visible y organizada del desarrollo del proyecto mediante sus diferentes etapas y tareas.

Por lo tanto las principales prácticas que se realizan al aplicar **Kanban** son:

- **Visualización del flujo de trabajo:** Mediante un sistema de tableros, columnas y notas (tareas) se puede visualizar claramente el flujo de trabajo que se realiza en el desarrollo del proyecto, resultando una herramienta efectiva para el seguimiento del proyecto.
- **Eliminar interrupciones:** Se establecen en conjunto con **SCRUM** límites de trabajo en las diferentes tareas, de forma que además del seguimiento de las mismas, se pueda seguir una traza y no se dediquen etapas muy variantes de esfuerzo en el desarrollo.
- **Gestión del flujo:** La principal ventaja es la de crear un flujo continuo e ininterrumpido, de forma que el movimiento de elementos sirva de rastreo de las tareas, de forma veloz y continua.
- **Retroalimentación:** Mediante las diferentes reuniones planteadas en **SCRUM**, además de la visualización, permite la definición de nuevas tareas o flujos de trabajo, de forma que una metodología alimente a la otra, creando un procedimiento de retroalimentación.

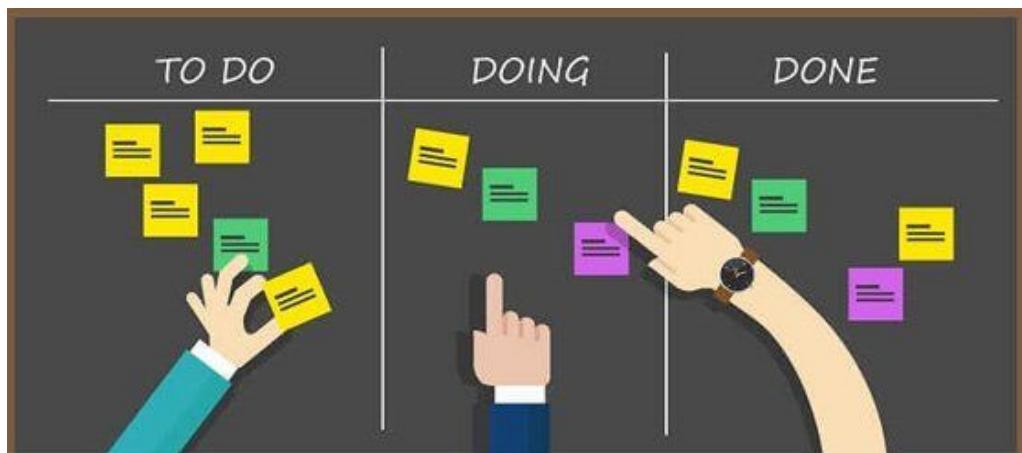


Imagen 31: Metodología Kanban

## 5.3. Diseño centrado en el usuario

El **Diseño Centrado en el Usuario** o **DCU** se utiliza a la hora de comprender a los usuarios, necesidades, comportamiento, entorno y tareas que estos realizan, por lo que

es una metodología necesaria al tratar con personas de diferentes generaciones, haciendo especial hincapié en las personas mayores al no estar acostumbradas a estas nuevas tecnologías.

Entre las características deseadas para el proyecto, se encuentran una serie de ventajas de utilizar esta metodología de **DCU**:

- **Especificación del contexto:** Se pueden identificar los usuarios, y en qué condiciones actúan, para comprender las necesidades de los mismos y el uso que realizarán del proyecto.
- **Requisitos:** Permite establecer una serie de requisitos más específicos, que añaden una mayor robustez y fiabilidad al sistema de cara al uso por los diferentes usuarios.
- **Evaluaciones del diseño:** Se realizarán diferentes evaluaciones durante el desarrollo del proyecto con diferentes usuarios con el fin de obtener un producto no solo funcional, sino usable y centrado en el modo de actuar de los usuarios.

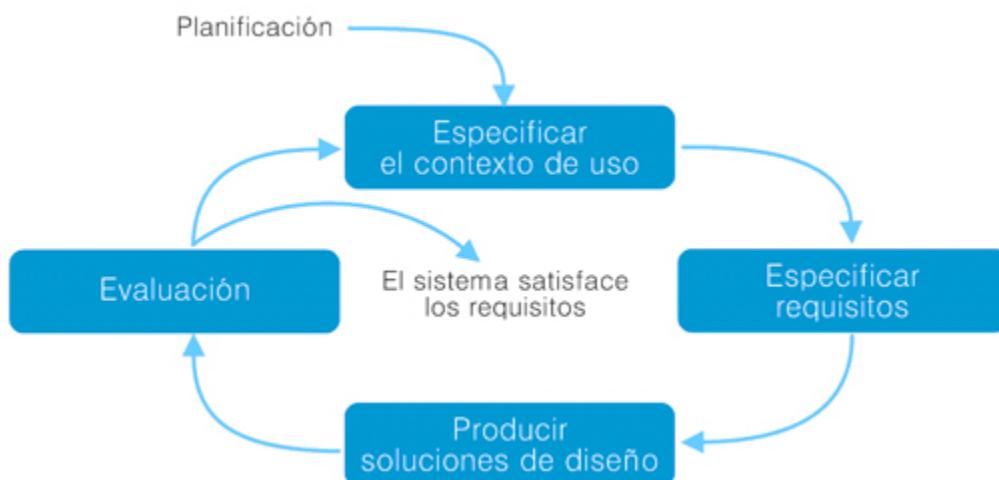


Imagen 32: Metodología DCU

#### 5.4. GDD

Si bien no es una metodología como tal, se considera importante mencionar las pautas seguidas para la elaboración de dicha documentación. El **GDD** o **Game Design Document** es el documento guía que marca el rumbo del proceso de la creación del videojuego. Es por ello que debe estar actualizado y establecer una visión de lo que será el videojuego. Al leerse debe quedar bien claro por qué los usuarios deberían jugar al juego.

En él se recoge información sobre:

- Título del videojuego.
- Nombre del estudio o equipo de diseñadores.

- Género del videojuego.
- Plataformas donde estará disponible.
- Versión del documento.
- Sinopsis de jugabilidad y contenido.
- Categoría del juego.
- Licencia.
- Mecánica del juego.
- Tecnología que se requiere para producir el juego.
- Público objetivo.

Los documentos de diseño de videojuegos pueden contener texto, imágenes, diagramas, dibujos conceptuales o cualquier contenido multimedia que pueda ilustrar mejor las decisiones de diseño. Algunos documentos de diseño pueden incluir prototipos funcionales o motores de juego seleccionados para determinadas partes del juego.

Aunque muchas empresas consideran que esto es necesario, el **GDD** no sigue ningún estándar. Por ejemplo, los desarrolladores pueden optar por mantener el documento como un documento de texto formateado, o pueden optar por mantenerlo en una herramienta de colaboración en línea. En el caso del proyecto, mediante un documento PDF que será actualizado constantemente.

## 5.5. Aplicación de las metodologías

En este proyecto se deben destacar una serie de puntuaciones respecto a las metodologías mencionadas previamente, ya que al tratarse de un trabajo de final de máster, únicamente se trata de un desarrollador del mismo, por lo que algunas metodologías no se aplicarán exactamente como se aplicaría en un entorno con múltiples desarrolladores.

En este proyecto se aplicará la metodología **SCRUM** para organizar las diferentes tareas y *sprints* a realizar en el proyecto apoyándose de una metodología visual como **Kanban**, con el objetivo de tener una trazabilidad del flujo de trabajo, y poseer una organización visual que permita optimizar el reparto de tareas desde nuestro **backlog de producto** con las diferentes tareas a realizar.

A su vez, estas tareas, siguiendo **metodologías ágiles** se han organizado en una serie de entregas que se organizan con el fin de obtener una serie de **prototipos funcionales** que van construyendo en diferentes etapas la solución final que se pretende obtener, tal y como se analizará posteriormente en este documento.

Cabe indicar que este proyecto se llevará a cabo siguiendo una metodología de **DCU**, realizando pruebas tras la finalización de cada entrega, con el fin de admitir posibles mejoras, o encontrar fallos y obtener un *feedback* de diferentes usuarios con el fin de establecer no solo unos requisitos funcionales en base al proyecto, sino poder obtener evaluaciones del diseño y analizar las necesidades y preferencias del usuario.

Por último, en cuanto a la elaboración del juego y la documentación del mismo, se utilizará un **GDD** el cual recoja los aspectos más creativos y funcionales en cuanto a la mecánica del juego, el cual se incluye como anexo dentro de este documento. El objetivo es establecer y definir de forma clara los aspectos que se pretenden englobar en este proyecto y dejar una clara trazabilidad de como se alcanzan esos objetivos usando las metodologías mencionadas previamente.

## 6. Plan de entregas.

En el siguiente apartado se detallarán las diferentes entregas que se realizarán a lo largo de los distintos avances del desarrollo del proyecto, indicando los diferentes objetivos de cada una de ellas, iteraciones y fechas de cada una de las entregas.

Al tratarse de un videojuego, la planificación es un procedimiento progresivo que puede ir variando durante la realización del mismo, denominándose esta planificación como *Just-In-Time*. En un principio las iteraciones durarán en base a unas 3 a 4 semanas, por lo que las iteraciones durarán aproximadamente 2 semanas.

Finalmente, las entregas se dividirán en base a las distintas necesidades, por lo que la planificación de la misma podrá sufrir ligeros cambios durante la realización del proyecto.

El desarrollo del proyecto comenzará el 9 de Marzo de 2021. La descomposición de las entregas se divide de la siguiente forma:

Entrega	Objetivo: Poseer una versión inicial del estilo del juego.	Fecha de la entrega
0	Estado del arte, tecnologías a usar, análisis inicial del problema, creación del GDD y documento de visión.	22 de Marzo de 2021

Entrega	Objetivo: Poseer un chatbot funcional con sesión volátil y realizar pruebas.	Fecha de la entrega						
1	<table border="1"><thead><tr><th>Iteración</th><th>Objetivo</th></tr></thead><tbody><tr><td>1</td><td>Prototipo juego de preguntas</td></tr><tr><td>2</td><td>Conexión con BD</td></tr></tbody></table>	Iteración	Objetivo	1	Prototipo juego de preguntas	2	Conexión con BD	12 de Abril de 2021
Iteración	Objetivo							
1	Prototipo juego de preguntas							
2	Conexión con BD							

Entrega	Objetivo: Obtener una versión funcional con la lógica de registro y mecánica del juego final.	Fecha de la entrega						
2	<table border="1"><thead><tr><th>Iteración</th><th>Objetivo</th></tr></thead><tbody><tr><td>1</td><td>Implementación sistema de registro en GrandQuiz</td></tr><tr><td>2</td><td>Implementación de la mecánica del juego e inyección de preguntas en BD</td></tr></tbody></table>	Iteración	Objetivo	1	Implementación sistema de registro en GrandQuiz	2	Implementación de la mecánica del juego e inyección de preguntas en BD	10 de Mayo de 2021
Iteración	Objetivo							
1	Implementación sistema de registro en GrandQuiz							
2	Implementación de la mecánica del juego e inyección de preguntas en BD							

Entrega	Objetivo: Ampliación de la mecánica del juego.		Fecha de la entrega
3			10 de Junio de 2021
	Iteración	Objetivo	
	1	Implementación del sistema de desafíos	
	2	Implementación del sistema de gamificación	

Entrega	Objetivo: Definición de una versión de juego individual.		Fecha de la entrega
4			16 de Junio de 2021
	Iteración	Objetivo	
	1	Implementación de la versión para un jugador en chat individual	

Entrega	Objetivo: Comunicación de eventos por correo electrónico.		Fecha de la entrega
5			20 de Junio de 2021
	Iteración	Objetivo	
	1	Implementación del sistema de notificación por registro	
	2	Implementación del sistema de notificación por ausencia en partida	

Entrega	Objetivo: Despliegue del proyecto y análisis de usuarios y usabilidad.		Fecha de la entrega						
6	<table border="1"> <thead> <tr> <th>Iteración</th> <th>Objetivo</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Despliegue del proyecto en cloud</td> </tr> <tr> <td>2</td> <td>Realizar pruebas y encuestas de usuarios y usabilidad</td> </tr> </tbody> </table>		Iteración	Objetivo	1	Despliegue del proyecto en cloud	2	Realizar pruebas y encuestas de usuarios y usabilidad	
Iteración	Objetivo								
1	Despliegue del proyecto en cloud								
2	Realizar pruebas y encuestas de usuarios y usabilidad								

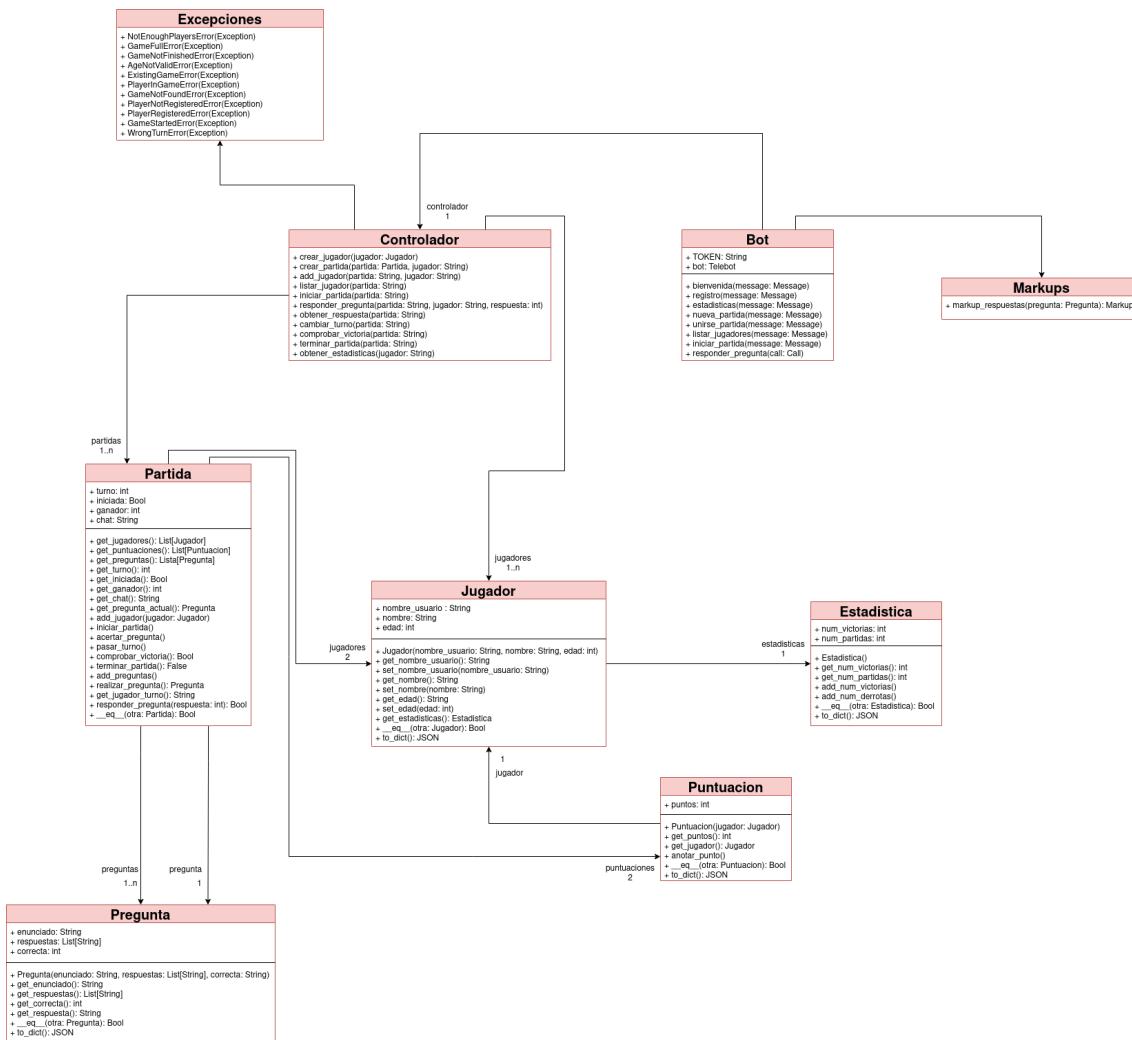
## 7. Desarrollo. Entrega e iteraciones.

### 7.1. Entrega 1

En esta entrega el propósito es entregar una versión del chatbot funcional con una sesión volátil, donde se pueda obtener una primera estructura funcional y comprobar el funcionamiento del Chatbot. En esta entrega, denominada *Questions*, se puede observar un juego de preguntas sencillo donde el usuario realizará una serie de funciones sencillas que dan pie al futuro desarrollo de la versión final.

Se debe considerar que esta primera entrega ya contiene determinados detalles que se incluirán en versiones finales, como puede ser la inclusión de fotos, la contestación de preguntas mediante botones o la visualización de estadísticas.

Por lo tanto, se propone en dicha versión el siguiente diagrama de clases:



En este diagrama se destacan por lo tanto las siguientes entidades:

- **Estadística**: Esta entidad contiene toda la información relacionada al número de

victorias y partidas de un jugador, por lo que contiene una referencia al mismo.

- **Puntuación:** Esta entidad contiene básicamente la cantidad de puntos que posee un determinado jugador en el contexto de una única partida, por lo que un mismo jugador tendrá diferentes puntuaciones en diferentes partidas.
- **Jugador:** Es la entidad principal de usuario que contiene los elementos que hacen referencia al mismo como su nombre o su *nick* de Telegram (*nombre\_usuario*) además de un factor importante en el juego como es la edad.
- **Pregunta:** Representa de forma sencilla una pregunta de tipo test con su enunciado y cuatro opciones disponibles, guardando también el índice de la respuesta correcta.
- **Partida:** Es la entidad sobre la que gira el juego, la cual contiene dos jugadores inicialmente con sus puntuaciones asociadas, además de una serie de preguntas predefinidas para poder realizar, posee toda la mecánica del juego y dirige el mismo.
- **Controlador:** Es la entidad que controla todo el sistema del proyecto, el cual se encarga de crear las diferentes entidades que componen el juego y administra los jugadores y las partidas de forma que se gestionen de forma individual.
- **Excepciones:** Asociadas al **controlador** se definen los diferentes tipos de excepciones del juego de forma específica.
- **Bot:** Es la interfaz del juego con Telegram, para ello hace uso de las funciones propias de la librería **Telebot** y utiliza el **controlador** como controlador de lógica del juego y del sistema, haciendo uso de los diferentes **markups** definidos.

En cuanto a las diferentes rutas de interacción con la API del bot, se distinguen:

- **/start:** Con este comando podemos recibir la bienvenida al juego por parte del bot y nos indica como proceder con el registro.
- **/registro:** Este comando inicia el proceso de registro, para ello es necesario indicar la edad de forma numérica tras el comando.
- **/estadisticas:** Este comando solicita la obtención de las estadísticas del jugador que lo solicita para visualizarlas.
- **/nueva\_partida:** Este comando (que solo puede ser utilizado en un grupo) crea una partida e indica el procedimiento para unirse a la misma.
- **/unirme:** Este comando (que solo puede ser utilizado en un grupo) solicita la unión a la partida del grupo (si existe) y comprueba si hay huecos disponibles, en caso afirmativo le une a la partida.
- **/lista:** Este comando (que solo puede ser utilizado en un grupo) muestra la lista de participantes en la partida del grupo.
- **/jugar:** Este comando (que solo puede ser utilizado en un grupo) inicia la partida comprobando que hay suficientes jugadores y establece los turnos, realizando la primera pregunta.

- **Responder pregunta:** Este manejador de callbacks obtiene la respuesta a una pregunta, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de victoria de un jugador finaliza la partida y lo indica.

A continuación, con el objetivo de comprender la funcionalidad propuesta, se muestran una serie de capturas donde se puede observar el comportamiento del juego, para poder comprender la realización del mismo.

Inicialmente se observa la inicialización del bot y el registro de un usuario:

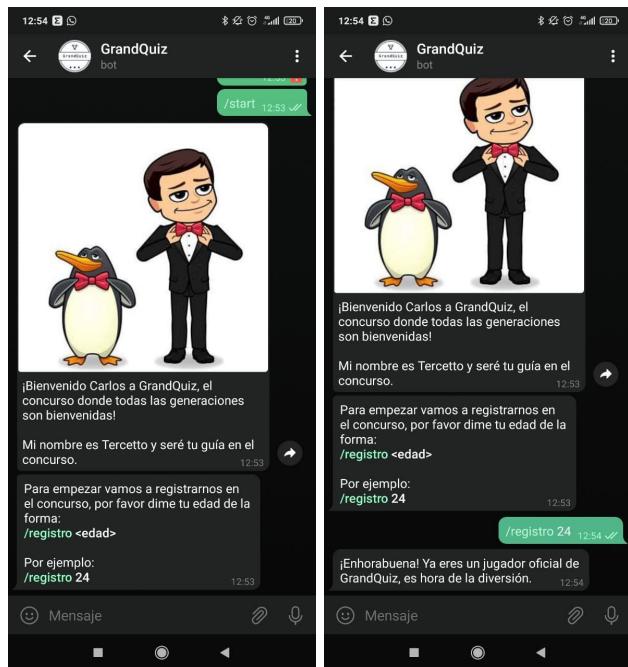


Imagen 33: Inicialización y registro

A continuación se observa la creación de una partida en un grupo, como se unen los jugadores, listado de los mismos y el inicio de la partida:

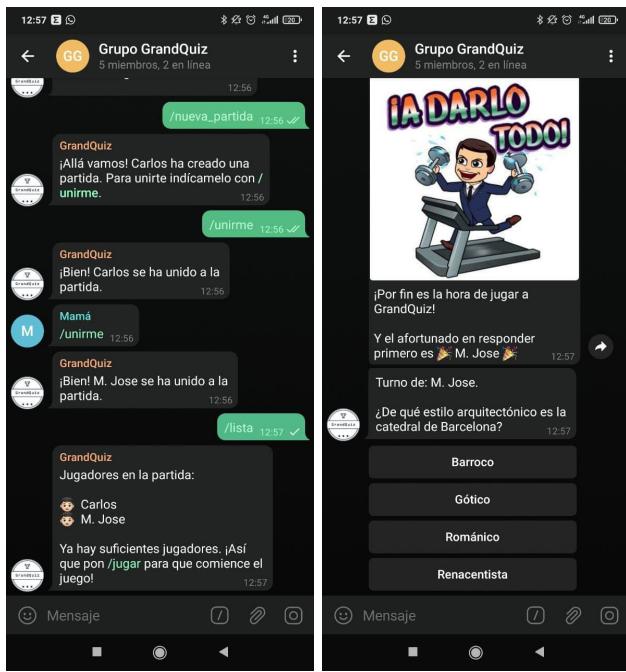


Imagen 34: Creación partida, listado e inicio

Por último se observa como sigue el flujo de la partida, como se finaliza la misma y las estadísticas de los jugadores:

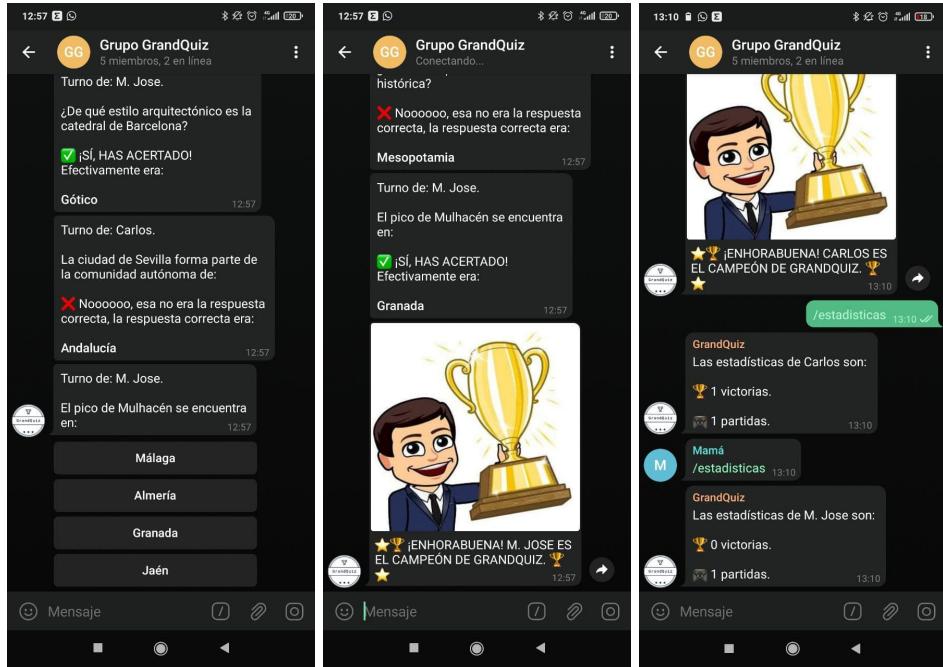


Imagen 35: Flujo partida, fin y estadísticas

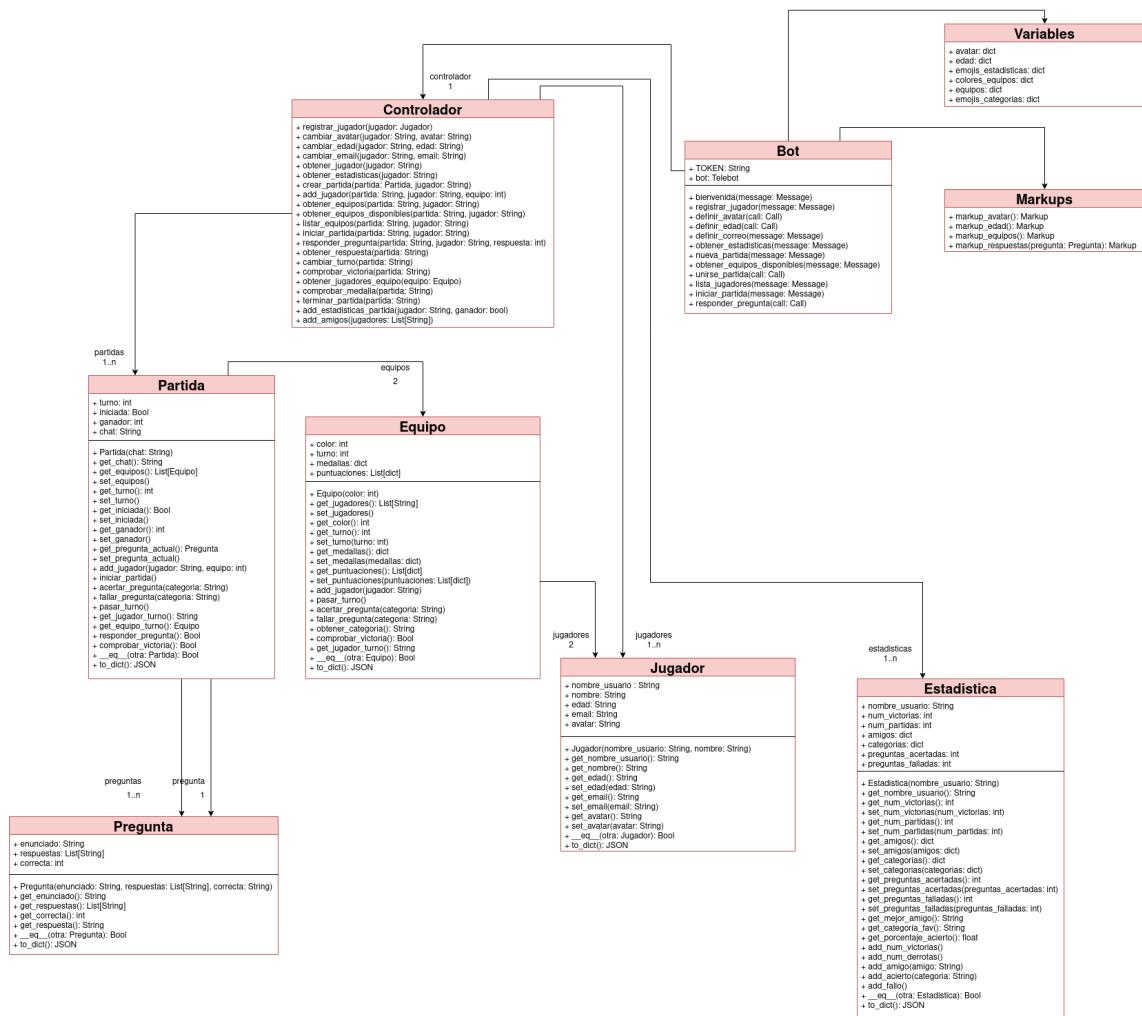
Finalmente, tras las diferentes características observadas en esta entrega, se han tomado una serie de anotaciones para el desarrollo de las diferentes entregas donde se orientará el desarrollo a la versión final. Además se incluirá el desarrollo del sistema haciendo uso de una base de datos no relacional altamente disponible.

## 7.2. Entrega 2

En esta entrega el propósito es entregar una versión del chatbot funcional con sesión permanente, donde se defina la mecánica básica de juego en *GrandQuiz*. En esta entrega, se puede observar la estructura básica de la partida con dos equipos, donde jugando por turnos los jugadores deben colaborar para obtener una serie de medallas de equipo de las diferentes categorías.

Se considera que a partir de esta entrega inicial de la primera versión del juego de *GrandQuiz* se desarrollarán las diferentes versiones como agregaciones de diferentes elementos del juego sobre esta mecánica simple inicial, como serán la comunicación de notificaciones, los duelos individuales, etc.

Por lo tanto, se propone en esta versión el siguiente diagrama de clases:



En este diagrama se destacan por lo tanto las siguientes entidades:

- Estadística:** Esta entidad contiene toda la información relacionada al número de victorias, partidas y otras características relacionadas con el rendimiento o relaciones de un jugador, por lo que contiene una referencia al mismo.
- Jugador:** Es la entidad principal de usuario que contiene los elementos que hacen referencia al mismo como su nombre o su *nick* de Telegram (*nombre\_usuario*) además de un factor importante en el juego como es la edad, además se incluyen en esta versión el correo electrónico y un avatar que añade cierta personalización.
- Pregunta:** Representa de forma sencilla una pregunta de tipo test con su enunciado y cuatro opciones disponibles, guardando también el índice de la respuesta correcta.
- Equipo:** Es la entidad que engloba a los equipos del juego, los cuales contienen los jugadores que forman parte del mismo, la mecánica asociada a las puntuaciones de los jugadores de forma individual y como equipo mediante medallas comunes, además del turno actual del jugador dentro del equipo.
- Partida:** Es la entidad sobre la que gira el juego, la cual contiene dos equipos

inicialmente con sus puntuaciones asociadas, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.

- **Controlador:** Es la entidad que controla todo el sistema del proyecto, el cual se encarga de crear las diferentes entidades que componen el juego y administra los jugadores y las partidas de forma que se gestionen de forma individual.
- **Variables:** Asociadas al **controlador** se definen las diferentes variables asociadas a visualizaciones de elementos en Telegram.
- **Bot:** Es la interfaz del juego con Telegram, para ello hace uso de las funciones propias de la librería **Telebot** y utiliza el **controlador** como controlador de lógica del juego y del sistema, haciendo uso de los diferentes **markups** definidos.

En cuanto a las diferentes rutas de interacción con la API del bot, se distinguen:

- **/start:** Con este comando podemos recibir la bienvenida al juego por parte del bot y nos indica como proceder con el registro.
- **/registro:** Este comando inicia el proceso de registro, indicandonos a continuación que introduzcamos el grupo de edad al que pertenecemos.
- **Definir edad:** Nos ofrece tres grupos de edades para asignarnos a un grupo con los que se harán las restricciones de equipos posteriormente.
- **Definir avatar:** Nos ofrece una lista de emojis para escoger como avatar, con el fin de añadir cierta personalización.
- **Definir correo:** Nos indica que escribamos nuestro correo, al enviarlo lo detecta y lo almacena, terminando el registro.
- **/estadisticas:** Este comando solicita la obtención de las estadísticas del jugador que lo solicita para visualizarlas.
- **/nueva\_partida:** Este comando (que solo puede ser utilizado en un grupo) crea una partida e indica el procedimiento para unirse a la misma.
- **/unirme:** Este comando (que solo puede ser utilizado en un grupo) solicita la unión a la partida del grupo (si existe) y comprueba si hay huecos disponibles, en caso afirmativo muestra los equipos disponibles.
- **Elegir equipo:** Permite seleccionar mediante botones alguno de los equipos con huecos disponibles y si se cumplen las restricciones de edades de los miembros que lo integran, lo acepta dentro.
- **/lista:** Este comando (que solo puede ser utilizado en un grupo) muestra la lista de participantes en la partida del grupo.
- **/jugar:** Este comando (que solo puede ser utilizado en un grupo) inicia la partida comprobando que hay suficientes jugadores y establece los turnos, realizando la primera pregunta.

- **Responder pregunta:** Este manejador de callbacks obtiene la respuesta a una pregunta, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de victoria de un equipo finaliza la partida y lo indica. Además incorpora la funcionalidad de notificar cuando un equipo obtiene una medalla.

A continuación, con el objetivo de comprender la funcionalidad propuesta, se muestran una serie de capturas donde se puede observar el comportamiento del juego, para poder comprender la realización del mismo.

Inicialmente se observa la inicialización del bot y el registro de un usuario:

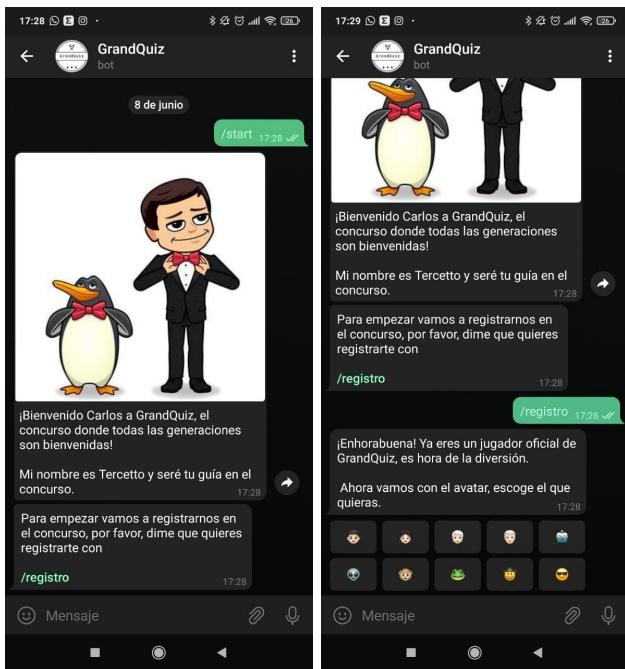


Imagen 36: Inicialización

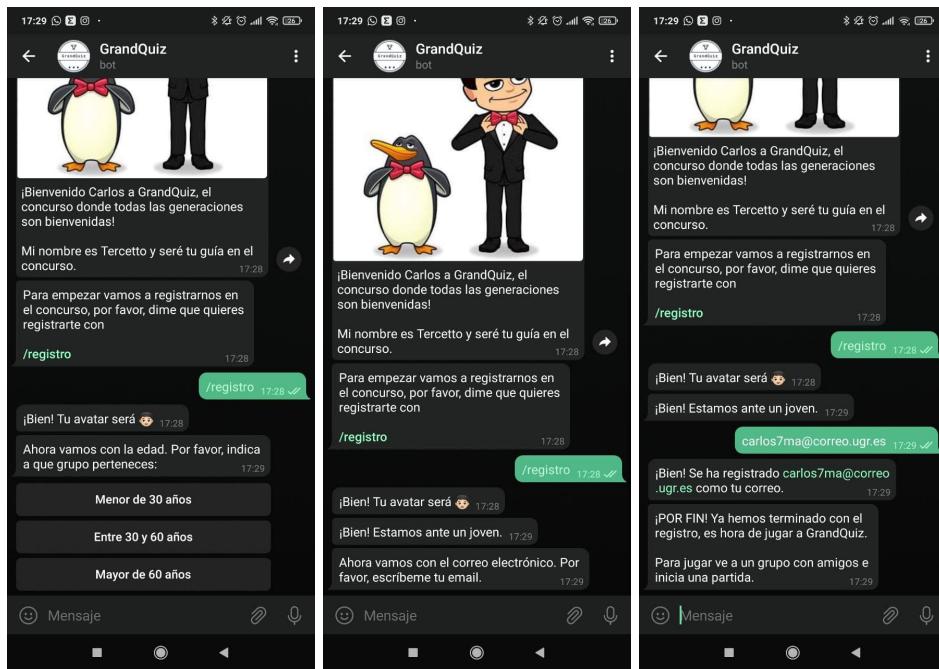


Imagen 37: Registro

A continuación se observa la creación de una partida en un grupo, como se unen los jugadores a los equipos, listado de los mismos y el inicio de la partida:

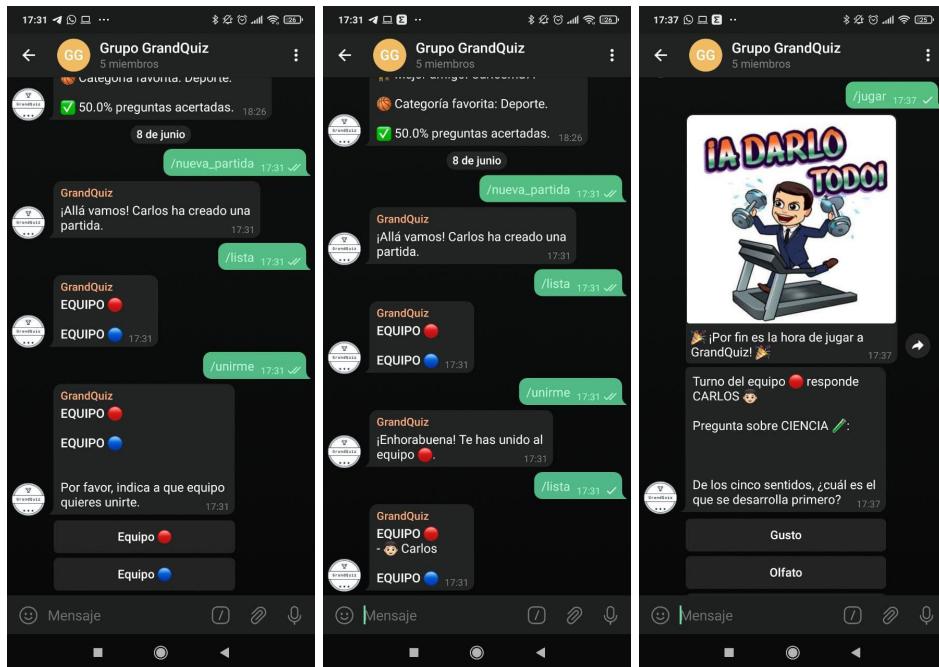


Imagen 38: Creación partida, listado e inicio

Por último se observa como sigue el flujo de la partida, como se obtienen las medallas de equipo, como se finaliza la misma y las estadísticas de los jugadores:

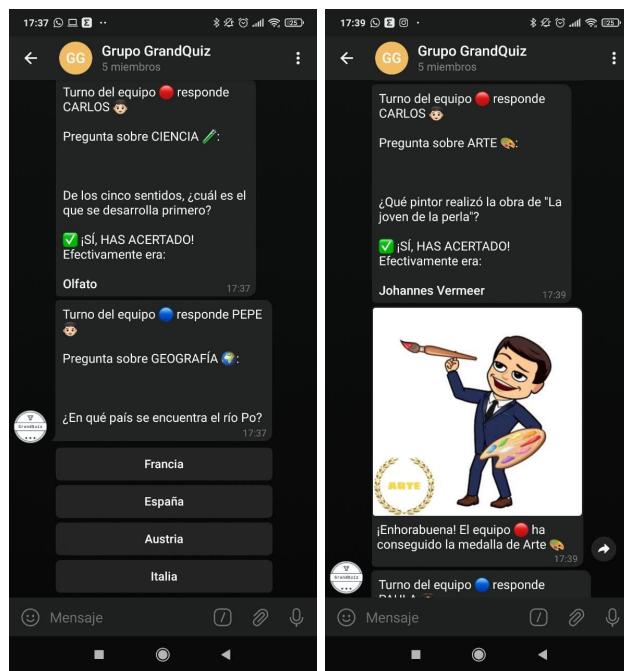


Imagen 39: Flujo partida y obtención de medallas

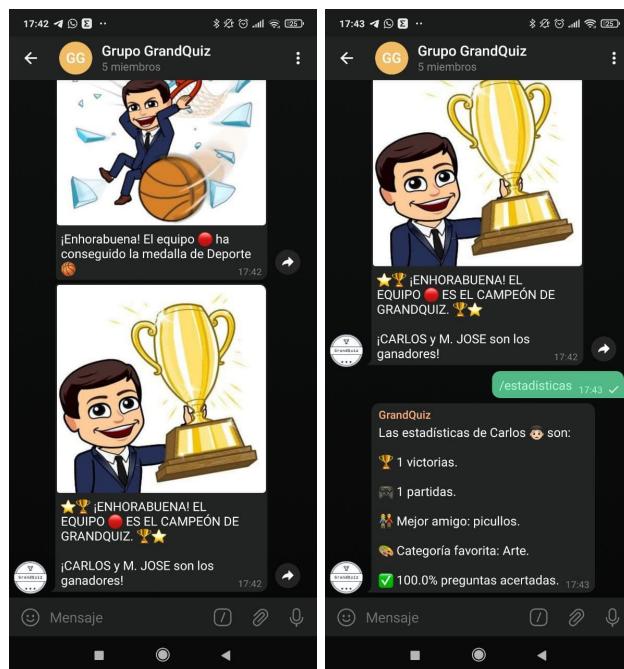


Imagen 40: Fin de partida y estadísticas

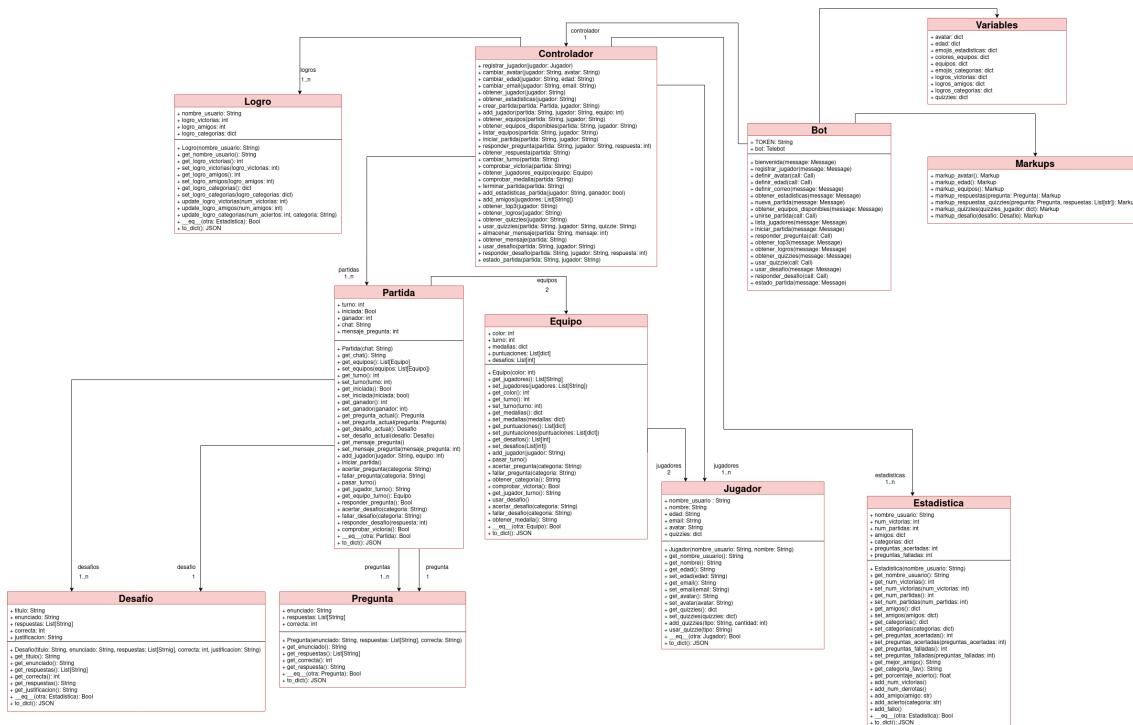
Finalmente, tras las diferentes características observadas en esta entrega, se establece una base de cara a las futuras entregas de este proyecto, realizando nuevas modificaciones y añadiendo nuevos elementos con el fin de completar el juego final.

### 7.3. Entrega 3

En esta entrega el propósito es añadir una gamificación al proyecto, donde se amplie la mecánica definida previamente en *GrandQuiz*. En esta entrega, se puede observar como se amplían las características mostradas previamente con la adición de un **Top** de jugadores, la creación de un sistema de logros que recompense a los jugadores por determinados hitos, la inclusión de un sistema de power-ups denominados *quizzes* incluidos mediante un sistema de gamificación y la ampliación mediante un sistema de desafíos especiales que amplían la mecánica del juego completo.

Con esta nueva entrega se amplía la mecánica de juego sencilla de la anterior entrega y se obtiene una versión actualizada de **GrandQuiz** que obtiene una serie de premios y valoraciones a partir del desempeño de los diferentes jugadores. A partir de este punto se considera la mecánica de juego ampliable en cuanto a la individualización de partidas con un nuevo formato que se desarrollará en la siguiente entrega, y posteriores entregas de mejora del proyecto con diversas funcionalidades como la comunicación mediante notificaciones u otros factores.

Por lo tanto, se propone en esta versión el siguiente diagrama de clases:



En este diagrama se destacan por lo tanto las siguientes entidades:

- **Estadística:** Esta entidad contiene toda la información relacionada al número de victorias, partidas y otras características relacionadas con el rendimiento o relaciones de un jugador, por lo que contiene una referencia al mismo.
  - **Logros:** Esta entidad contiene toda la información relacionada al sistema de logros del juego, recogiendo los logros por victorias, amigos y categorías de los jugadores.

- **Jugador:** Es la entidad principal de usuario que contiene los elementos que hacen referencia al mismo como su nombre o su *nick* de Telegram (*nombre\_usuario*) además de un factor importante en el juego como es la edad, además se incluyen en esta versión el correo electrónico y un avatar que añade cierta personalización.
- **Pregunta:** Representa de forma sencilla una pregunta de tipo test con su enunciado y cuatro opciones disponibles, guardando también el índice de la respuesta correcta.
- **Desafío:** Representa de forma sencilla una pregunta mediante una imagen de un desafío de diferentes tipos, como un razonamiento, reconocimiento o análisis del problema, ofreciendo cuatro opciones disponibles y la justificación de la respuesta, guardando también el índice de la respuesta correcta.
- **Equipo:** Es la entidad que engloba a los equipos del juego, los cuales contienen los jugadores que forman parte del mismo, la mecánica asociada a las puntuaciones de los jugadores de forma individual y como equipo mediante medallas comunes, además del turno actual del jugador dentro del equipo.
- **Partida:** Es la entidad sobre la que gira el juego, la cual contiene dos equipos inicialmente con sus puntuaciones asociadas, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.
- **Controlador:** Es la entidad que controla todo el sistema del proyecto, el cual se encarga de crear las diferentes entidades que componen el juego y administra los jugadores y las partidas de forma que se gestionen de forma individual.
- **Variables:** Asociadas al **controlador** se definen las diferentes variables asociadas a visualizaciones de elementos en Telegram.
- **Bot:** Es la interfaz del juego con Telegram, para ello hace uso de las funciones propias de la librería **Telebot** y utiliza el **controlador** como controlador de lógica del juego y del sistema, haciendo uso de los diferentes **markups** definidos.

En cuanto a las diferentes rutas de interacción con la API del bot, se distinguen:

- **/start:** Con este comando podemos recibir la bienvenida al juego por parte del bot y nos indica cómo proceder con el registro.
- **/registro:** Este comando inicia el proceso de registro, indicandonos a continuación que introduzcamos el grupo de edad al que pertenecemos.
- **Definir edad:** Nos ofrece tres grupos de edades para asignarnos a un grupo con los que se harán las restricciones de equipos posteriormente.
- **Definir avatar:** Nos ofrece una lista de emojis para escoger como avatar, con el fin de añadir cierta personalización.
- **Definir correo:** Nos indica que escribamos nuestro correo, al enviarlo lo detecta y lo almacena, terminando el registro.
- **/estadisticas:** Este comando solicita la obtención de las estadísticas del jugador que lo solicita para visualizarlas.

- **/nueva\_partida:** Este comando (que solo puede ser utilizado en un grupo) crea una partida e indica el procedimiento para unirse a la misma.
- **/unirme:** Este comando (que solo puede ser utilizado en un grupo) solicita la unión a la partida del grupo (si existe) y comprueba si hay huecos disponibles, en caso afirmativo muestra los equipos disponibles.
- **Elegir equipo:** Permite seleccionar mediante botones alguno de los equipos con huecos disponibles y si se cumplen las restricciones de edades de los miembros que lo integran, lo acepta dentro.
- **/lista:** Este comando (que solo puede ser utilizado en un grupo) muestra la lista de participantes en la partida del grupo.
- **/jugar:** Este comando (que solo puede ser utilizado en un grupo) inicia la partida comprobando que hay suficientes jugadores y establece los turnos, realizando la primera pregunta.
- **Responder pregunta:** Este manejador de **callbacks** obtiene la respuesta a una pregunta, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de victoria de un equipo finaliza la partida y lo indica. Además incorpora la funcionalidad de notificar cuando un equipo obtiene una medalla.
- **/top:** Este comando obtiene el top 3 de jugadores con las diferentes categorías tanto de victorias, amigos y preguntas acertadas.
- **/logros:** Este comando obtiene los logros del jugador indicado en forma de lista obteniendo los logros asociados a victorias, amigos y categorías de preguntas.
- **/quizzes:** Nos ofrece la lista de *quizzes* disponibles del juego, considerando que el usuario disponga de estos y mostrando la cantidad disponible.
- **Usar quizzie:** Permite seleccionar mediante botones los *quizzes* indicados y se encarga de la gestión de la aplicación del comodín seleccionado respecto a la pregunta existente.
- **/desafio:** Este comando indica que se quiere utilizar el desafío de la partida, por lo que se elimina la anterior pregunta y se muestra el desafío con su imagen asociada.
- **Responder desafío:** Este manejador de **callbacks** obtiene la respuesta a un desafío, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de acierto se roba una medalla aleatoria del equipo contrario, pero en caso de fallo se pierde una medalla aleatoria.
- **/estado:** Este comando se utiliza en mitad de una partida para obtener las medallas de cada equipo y observar el estado de la partida en ese punto.

A continuación, con el objetivo de comprender la funcionalidad propuesta, se muestran una serie de capturas donde se puede observar el comportamiento del juego, para poder comprender la realización del mismo.

Inicialmente se observa la top clasificatoria del bot:



Imagen 41: Top 3

Posteriormente, observamos la solicitud y visualización de los logros del jugador (u otros indicando el nick) como un listado:

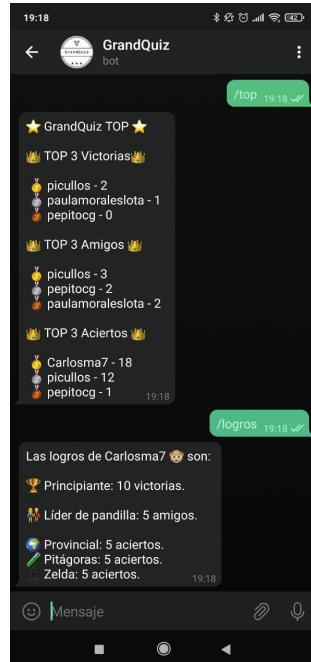


Imagen 42: Logros

A continuación, se observa como se utilizan los comodines o *quizzies* en la partida,

primero solicitándolos (se muestran únicamente los que se poseen indicando el número restante de los mismos del jugador), y posteriormente usando uno de cada tipo:

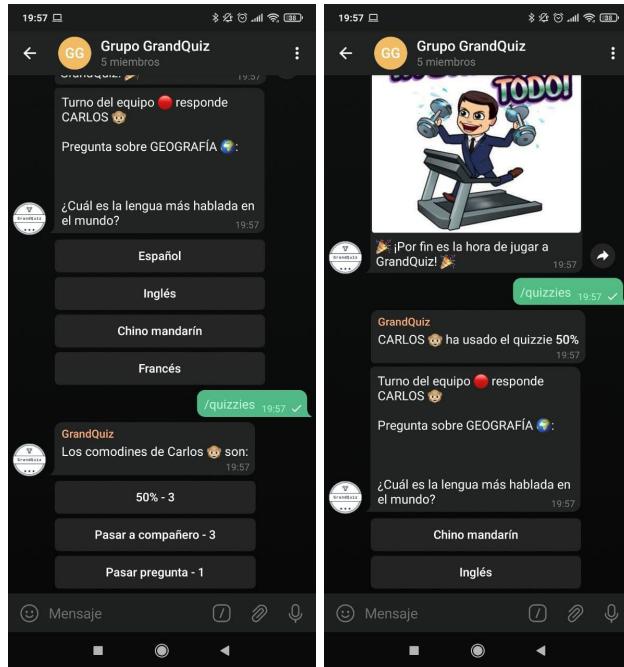


Imagen 43: Solicitud y uso de *quizzie* del 50 %

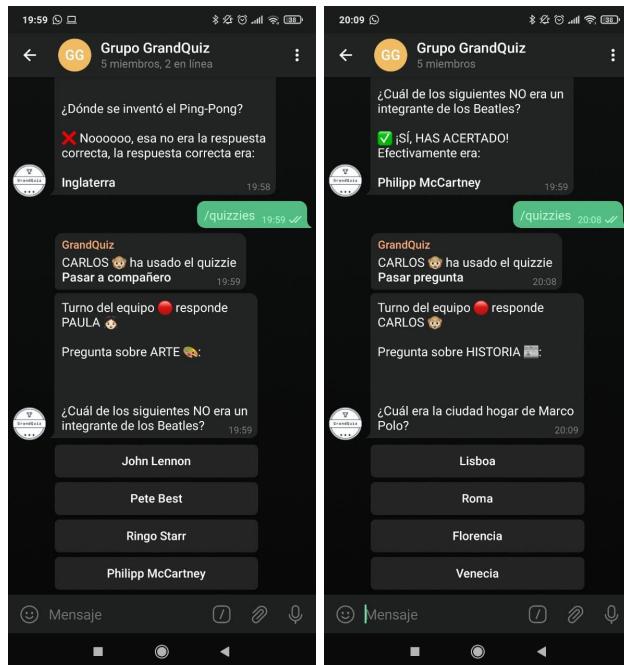


Imagen 44: Uso de *quizzes* de pasar a compañero y pasar pregunta

Finalmente, podemos ver como funciona la solicitud y uso de un desafío en la partida, para ello en el turno de respuesta, el jugador solicita el uso del desafío y se muestra la

imagen y la pregunta del mismo, ofreciendo las respuestas y al realizar la pregunta robar o perder una medalla aleatoria, siempre que se posean:

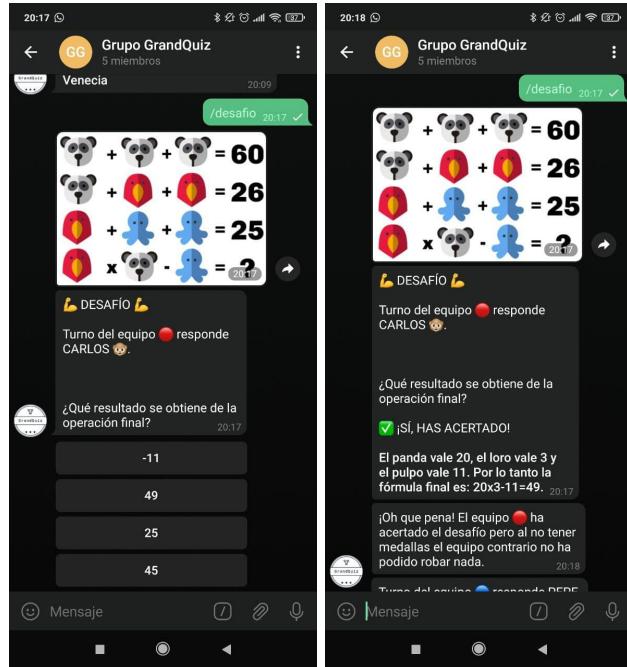


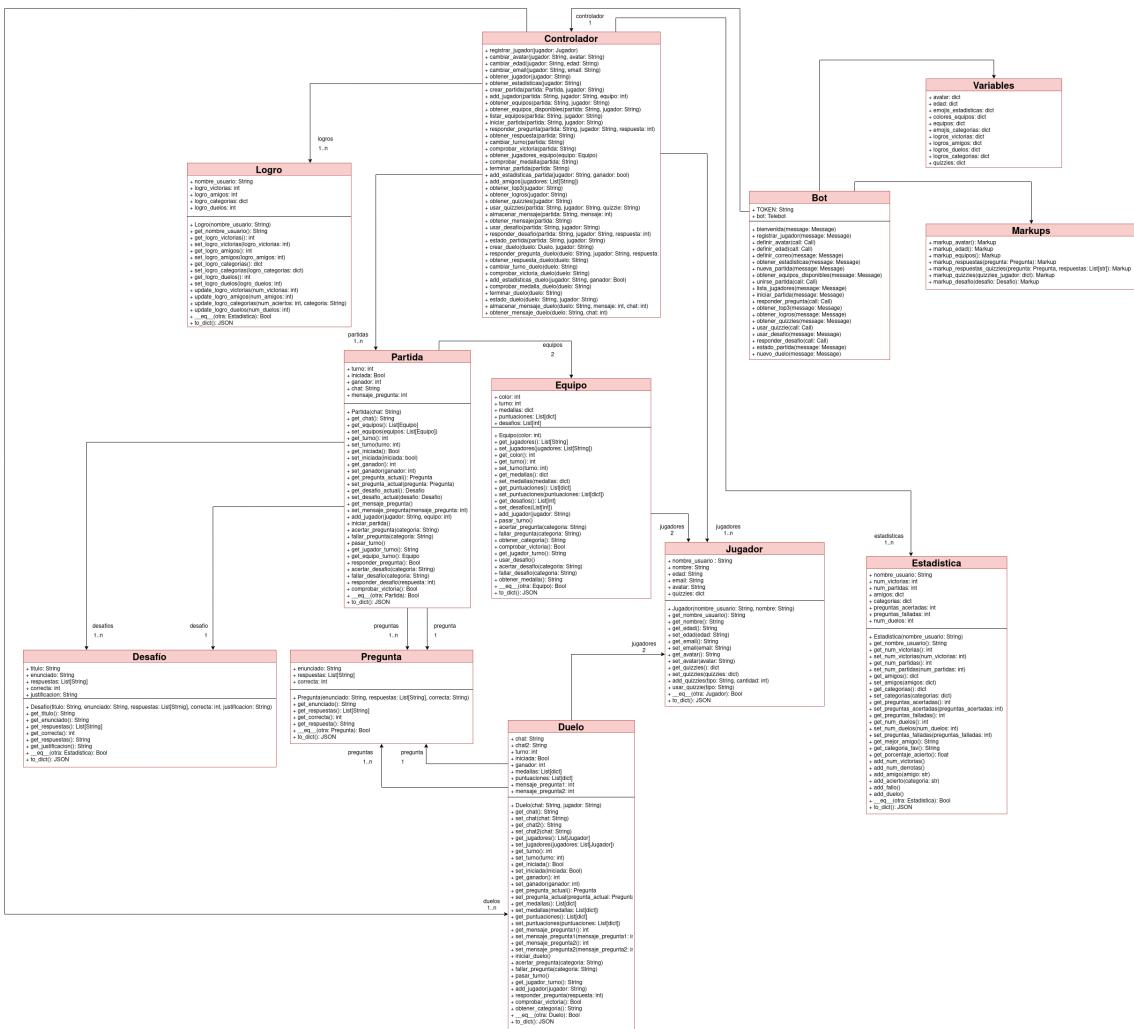
Imagen 45: Uso de desafío

#### 7.4. Entrega 4

En esta entrega el objetivo es el de obtener una versión individual del juego, ampliando las posibilidades de juego del mismo, mediante un sistema de **duelos** donde el jugador puede jugar con otros jugadores de forma aleatoria e individual. En esta entrega este sistema contempla la misma mecánica de juego salvo que la consecución de medallas por parte de un jugador son al obtener dos puntos de cada categoría de forma consecutiva. En esta versión simplificada de las partidas no se contemplan ni los *quizzes* ni los desafíos, ya que se pretende obtener una versión más dinámica y reducida del juego.

Con esta nueva entrega, a partir de las anteriores se obtiene una versión de juego más sencilla y complementaria de la anterior entrega y se obtiene una versión más amplia de **GrandQuiz**, siendo premiados con la consecución de victorias en duelos como una nueva sección de estadísticas y logros, y contemplando las mismas categorías de juego de las partidas previas.

Por lo tanto, se propone en esta versión el siguiente diagrama de clases:



En este diagrama se destacan por lo tanto las siguientes entidades:

- **Estadística:** Esta entidad contiene toda la información relacionada al número de victorias, partidas, duelos y otras características relacionadas con el rendimiento o relaciones de un jugador, por lo que contiene una referencia al mismo.
  - **Logros:** Esta entidad contiene toda la información relacionada al sistema de logros del juego, recogiendo los logros por victorias, amigos, duelos ganados y categorías de los jugadores.
  - **Jugador:** Es la entidad principal de usuario que contiene los elementos que hacen referencia al mismo como su nombre o su *nick* de Telegram (*nombre\_usuario*) además de un factor importante en el juego como es la edad, además se incluyen en esta versión el correo electrónico y un avatar que añade cierta personalización.
  - **Pregunta:** Representa de forma sencilla una pregunta de tipo test con su enunciado y cuatro opciones disponibles, guardando también el índice de la respuesta correcta.
  - **Desafío:** Representa de forma sencilla una pregunta mediante una imagen de un desafío de diferentes tipos, como un razonamiento, reconocimiento o análisis del

problema, ofreciendo cuatro opciones disponibles y la justificación de la respuesta, guardando también el índice de la respuesta correcta.

- **Equipo:** Es la entidad que engloba a los equipos del juego, los cuales contienen los jugadores que forman parte del mismo, la mecánica asociada a las puntuaciones de los jugadores de forma individual y como equipo mediante medallas comunes, además del turno actual del jugador dentro del equipo.
- **Partida:** Es la entidad sobre la que gira el juego, la cual contiene dos equipos inicialmente con sus puntuaciones asociadas, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.
- **Duelo:** Es una entidad reducida de la versión individual de la partida, la cual contiene dos jugadores con diferentes chats, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.
- **Controlador:** Es la entidad que controla todo el sistema del proyecto, el cual se encarga de crear las diferentes entidades que componen el juego y administra los jugadores y las partidas de forma que se gestionen de forma individual.
- **Variables:** Asociadas al **controlador** se definen las diferentes variables asociadas a visualizaciones de elementos en Telegram.
- **Bot:** Es la interfaz del juego con Telegram, para ello hace uso de las funciones propias de la librería **Telebot** y utiliza el **controlador** como controlador de lógica del juego y del sistema, haciendo uso de los diferentes **markups** definidos.

En cuanto a las diferentes rutas de interacción con la API del bot, se distinguen:

- **/start:** Con este comando podemos recibir la bienvenida al juego por parte del bot y nos indica como proceder con el registro.
- **/registro:** Este comando inicia el proceso de registro, indicandonos a continuación que introduzcamos el grupo de edad al que pertenecemos.
- **Definir edad:** Nos ofrece tres grupos de edades para asignarnos a un grupo con los que se harán las restricciones de equipos posteriormente.
- **Definir avatar:** Nos ofrece una lista de emojis para escoger como avatar, con el fin de añadir cierta personalización.
- **Definir correo:** Nos indica que escribamos nuestro correo, al enviarlo lo detecta y lo almacena, terminando el registro.
- **/estadisticas:** Este comando solicita la obtención de las estadísticas del jugador que lo solicita para visualizarlas.
- **/nueva\_partida:** Este comando (que solo puede ser utilizado en un grupo) crea una partida e indica el procedimiento para unirse a la misma.
- **/unirme:** Este comando (que solo puede ser utilizado en un grupo) solicita la unión a la partida del grupo (si existe) y comprueba si hay huecos disponibles, en caso afirmativo muestra los equipos disponibles.

- **Elegir equipo:** Permite seleccionar mediante botones alguno de los equipos con huecos disponibles y si se cumplen las restricciones de edades de los miembros que lo integran, lo acepta dentro.
- **/lista:** Este comando (que solo puede ser utilizado en un grupo) muestra la lista de participantes en la partida del grupo.
- **/jugar:** Este comando (que solo puede ser utilizado en un grupo) inicia la partida comprobando que hay suficientes jugadores y establece los turnos, realizando la primera pregunta.
- **Responder pregunta:** Este manejador de **callbacks** obtiene la respuesta a una pregunta, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de victoria de un equipo finaliza la partida y lo indica. Además incorpora la funcionalidad de notificar cuando un equipo obtiene una medalla.
- **/top:** Este comando obtiene el top 3 de jugadores con las diferentes categorías tanto de victorias, amigos y preguntas acertadas.
- **/logros:** Este comando obtiene los logros del jugador indicado en forma de lista obteniendo los logros asociados a victorias, amigos y categorías de preguntas.
- **/quizzes:** Nos ofrece la lista de *quizzes* disponibles del juego, considerando que el usuario disponga de estos y mostrando la cantidad disponible.
- **Usar quizzie:** Permite seleccionar mediante botones los *quizzes* indicados y se encarga de la gestión de la aplicación del comodín seleccionado respecto a la pregunta existente.
- **/desafío:** Este comando indica que se quiere utilizar el desafío de la partida, por lo que se elimina la anterior pregunta y se muestra el desafío con su imagen asociada.
- **Responder desafío:** Este manejador de **callbacks** obtiene la respuesta a un desafío, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de acierto se roba una medalla aleatoria del equipo contrario, pero en caso de fallo se pierde una medalla aleatoria.
- **/estado:** Este comando se utiliza en mitad de una partida para obtener las medallas de cada equipo y observar el estado de la partida en ese punto.
- **/nuevo\_duelo:** Este comando (que solo puede ser utilizado en un chat privado) crea un duelo o te une a uno existente e inicia la partida devolviendo la primera pregunta.

A continuación, con el objetivo de comprender la funcionalidad propuesta, se muestran una serie de capturas donde se puede observar el comportamiento del juego, para poder comprender la realización del mismo.

Inicialmente se muestra la creación del duelo y como se obtiene la primera pregunta en cuanto se une el segundo jugador:

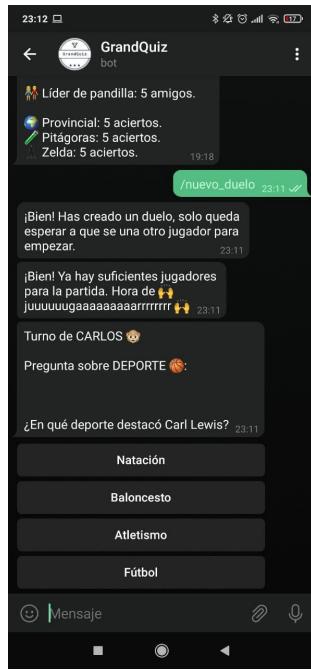


Imagen 46: Creación e inicio de un duelo nuevo

A continuación, se muestra como se contestan las preguntas y sigue el curso del duelo, pasando el turno al siguiente jugador, pero mostrándose igualmente en ambos chats para poder conocer el desempeño de la partida:

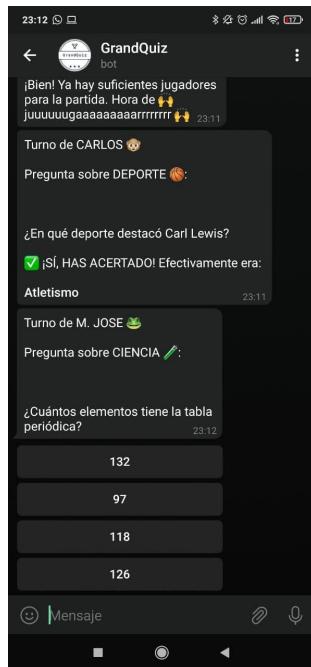


Imagen 47: Responder pregunta en el duelo y cambio de turno

Finalmente, se muestra cuando un jugador gana informando a ambos usuarios, además

se puede observar el simple mensaje que se obtiene cuando se consigue una medalla de una categoría:



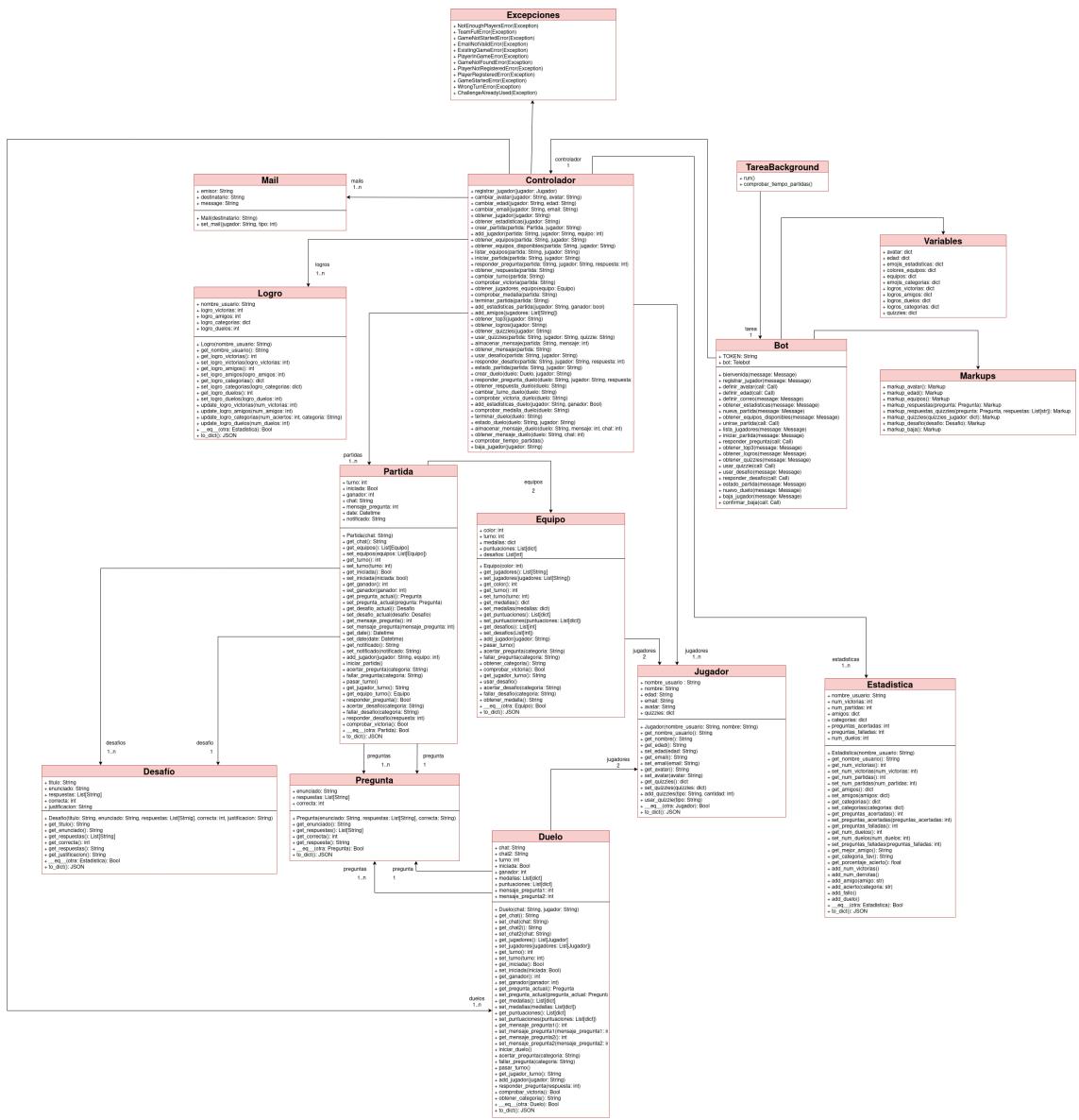
Imagen 48: Fin de partida y notificación de ganador

## 7.5. Entrega 5

En esta penúltima entrega el objetivo es definir un sistema de notificación por correo ante diferentes eventos del juego, como son el registro o la inactividad en partidas. En esta entrega se finaliza el desarrollo del sistema y se obtiene una versión completa del mismo. A partir de este punto el objetivo será el de desplegar el proyecto como un servicio cloud y obtener un sistema de alta disponibilidad tras escoger un proveedor.

En esta entrega se define también una página web de presentación del proyecto la cual contiene la información relevante del mismo de forma amistosa para los jugadores y con una ayuda de como usarlo. Todo esto se define en la Página oficial de GrandQuiz.

Por lo tanto, se propone en dicha versión el siguiente diagrama de clases:



En este diagrama se destacan por lo tanto las siguientes entidades:

- Estadística**: Esta entidad contiene toda la información relacionada al número de victorias, partidas, duelos y otras características relacionadas con el rendimiento o relaciones de un jugador, por lo que contiene una referencia al mismo.
- Logros**: Esta entidad contiene toda la información relacionada al sistema de logros del juego, recogiendo los logros por victorias, amigos, duelos ganados y categorías de los jugadores.
- Jugador**: Es la entidad principal de usuario que contiene los elementos que hacen referencia al mismo como su nombre o su *nick* de Telegram (*nombre\_usuario*) además de un factor importante en el juego como es la edad, además se incluyen en esta versión el correo electrónico y un avatar que añade cierta personalización.

- **Pregunta:** Representa de forma sencilla una pregunta de tipo test con su enunciado y cuatro opciones disponibles, guardando también el índice de la respuesta correcta.
- **Desafío:** Representa de forma sencilla una pregunta mediante una imagen de un desafío de diferentes tipos, como un razonamiento, reconocimiento o análisis del problema, ofreciendo cuatro opciones disponibles y la justificación de la respuesta, guardando también el índice de la respuesta correcta.
- **Equipo:** Es la entidad que engloba a los equipos del juego, los cuales contienen los jugadores que forman parte del mismo, la mecánica asociada a las puntuaciones de los jugadores de forma individual y como equipo mediante medallas comunes, además del turno actual del jugador dentro del equipo.
- **Partida:** Es la entidad sobre la que gira el juego, la cual contiene dos equipos inicialmente con sus puntuaciones asociadas, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.
- **Duelo:** Es una entidad reducida de la versión individual de la partida, la cual contiene dos jugadores con diferentes chats, obtiene preguntas de forma dinámica, posee toda la mecánica del juego y dirige el mismo.
- **Controlador:** Es la entidad que controla todo el sistema del proyecto, el cual se encarga de crear las diferentes entidades que componen el juego y administra los jugadores y las partidas de forma que se gestionen de forma individual.
- **Variables:** Asociadas al **controlador** se definen las diferentes variables asociadas a visualizaciones de elementos en Telegram.
- **Bot:** Es la interfaz del juego con Telegram, para ello hace uso de las funciones propias de la librería **Telebot** y utiliza el **controlador** como controlador de lógica del juego y del sistema, haciendo uso de los diferentes **markups** definidos.
- **Excepciones:** Asociadas al **controlador** se definen los diferentes tipos de excepciones del juego de forma específica.
- **TareaBackground:** Define una tarea periódica que comprueba cada hora las partidas que llevan más de 24 y 48 horas inactivas para notificar por mail o conceder la victoria al equipo que espera.
- **Mail:** Define el mecanismo de envíos de correo electrónicos de notificación de registro e inactividad, gestionando el acceso al correo y el uso de una plantilla con el nombre personalizado.

En cuanto a las diferentes rutas de interacción con la API del bot, se distinguen:

- **/start:** Con este comando podemos recibir la bienvenida al juego por parte del bot y nos indica como proceder con el registro.
- **/registro:** Este comando inicia el proceso de registro, indicandonos a continuación que introduzcamos el grupo de edad al que pertenecemos.
- **Definir edad:** Nos ofrece tres grupos de edades para asignarnos a un grupo con los que se harán las restricciones de equipos posteriormente.

- **Definir avatar:** Nos ofrece una lista de emojis para escoger como avatar, con el fin de añadir cierta personalización.
- **Definir correo:** Nos indica que escribamos nuestro correo, al enviarlo lo detecta y lo almacena, terminando el registro.
- **/estadísticas:** Este comando solicita la obtención de las estadísticas del jugador que lo solicita para visualizarlas.
- **/nueva\_partida:** Este comando (que solo puede ser utilizado en un grupo) crea una partida e indica el procedimiento para unirse a la misma.
- **/unirme:** Este comando (que solo puede ser utilizado en un grupo) solicita la unión a la partida del grupo (si existe) y comprueba si hay huecos disponibles, en caso afirmativo muestra los equipos disponibles.
- **Elegir equipo:** Permite seleccionar mediante botones alguno de los equipos con huecos disponibles y si se cumplen las restricciones de edades de los miembros que lo integran, lo acepta dentro.
- **/lista:** Este comando (que solo puede ser utilizado en un grupo) muestra la lista de participantes en la partida del grupo.
- **/jugar:** Este comando (que solo puede ser utilizado en un grupo) inicia la partida comprobando que hay suficientes jugadores y establece los turnos, realizando la primera pregunta.
- **Responder pregunta:** Este manejador de **callbacks** obtiene la respuesta a una pregunta, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de victoria de un equipo finaliza la partida y lo indica. Además incorpora la funcionalidad de notificar cuando un equipo obtiene una medalla.
- **/top:** Este comando obtiene el top 3 de jugadores con las diferentes categorías tanto de victorias, amigos y preguntas acertadas.
- **/logros:** Este comando obtiene los logros del jugador indicado en forma de lista obteniendo los logros asociados a victorias, amigos y categorías de preguntas.
- **/quizzes:** Nos ofrece la lista de *quizzes* disponibles del juego, considerando que el usuario disponga de estos y mostrando la cantidad disponible.
- **Usar quizzie:** Permite seleccionar mediante botones los *quizzes* indicados y se encarga de la gestión de la aplicación del comodín seleccionado respecto a la pregunta existente.
- **/desafío:** Este comando indica que se quiere utilizar el desafío de la partida, por lo que se elimina la anterior pregunta y se muestra el desafío con su imagen asociada.
- **Responder desafío:** Este manejador de **callbacks** obtiene la respuesta a un desafío, comprueba que responde el jugador del turno, y solicita que se compruebe el acierto. A continuación indica si ha acertado o no, y realiza la siguiente pregunta. En caso de acierto se roba una medalla aleatoria del equipo contrario, pero en caso de fallo se pierde una medalla aleatoria.

- **/estado:** Este comando se utiliza en mitad de una partida para obtener las medallas de cada equipo y observar el estado de la partida en ese punto.
- **/nuevo\_duelo:** Este comando (que solo puede ser utilizado en un chat privado) crea un duelo o te une a uno existente e inicia la partida devolviendo la primera pregunta.
- **/baja:** Este comando (que solo puede ser utilizado en un chat privado) solicita darse de baja al sistema, y este ofrece un diálogo de confirmación con dos botones.
- **Confirmar baja:** Este manejador de callbacks comprueba si la respuesta a la baja del usuario es afirmativa, eliminándolo del sistema y despidiéndose de él.

A continuación, con el objetivo de comprender la funcionalidad propuesta, se muestran una serie de capturas donde se puede observar el comportamiento del juego, para poder comprender la realización del mismo.

Para empezar se muestran los correos que se reciben, tanto para el registro en el juego como para la notificación por inactividad tras 24 horas sin jugar en una partida existente:

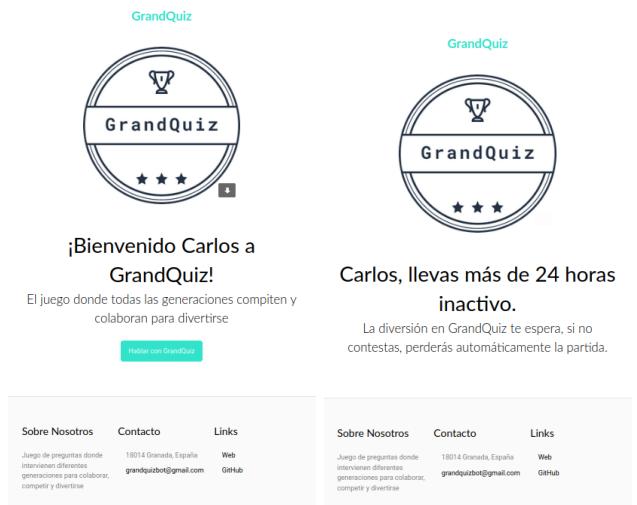


Imagen 49: Modelo de correos de notificación de registro e inactividad

Por otro lado, se observa la Página Oficial de GrandQuiz, la cual se puede previsualizar:

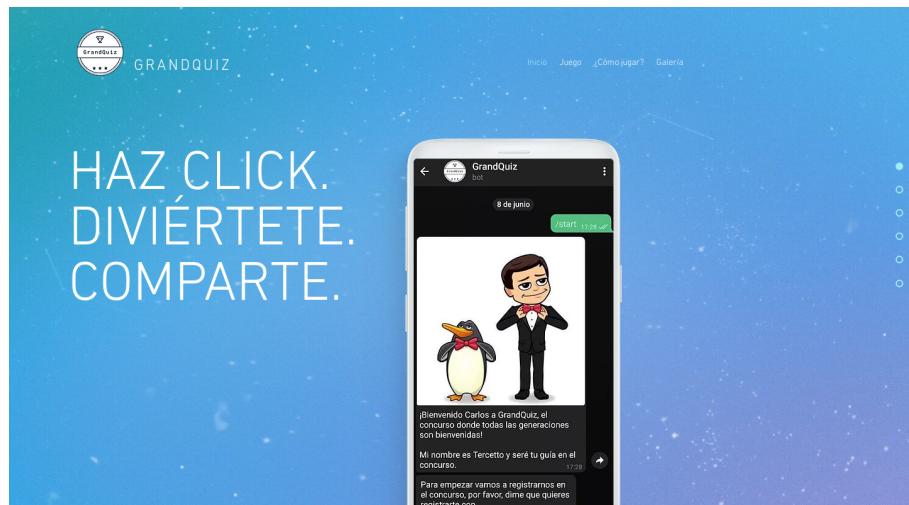


Imagen 50: Página Oficial de GrandQuiz

A continuación, se muestra dentro del juego como se realiza el procedimiento de baja de un usuario:

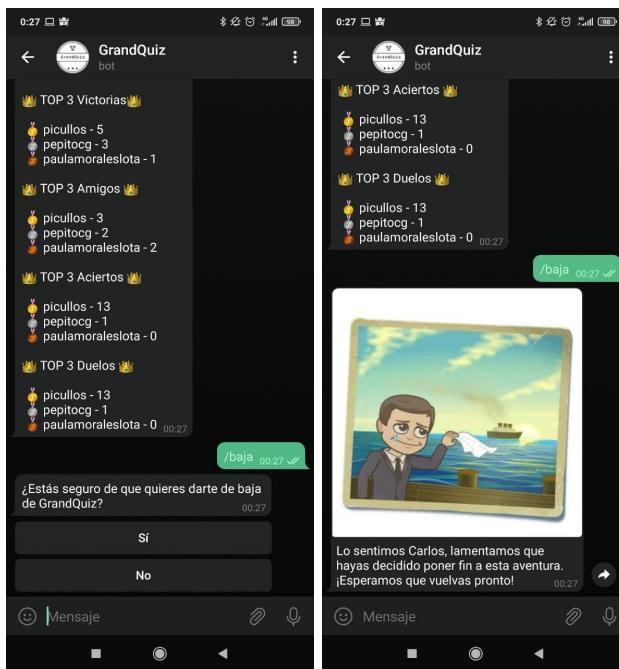


Imagen 51: Procedimiento de baja y confirmación de un usuario

## 7.6. Entrega 6

En esta entrega final del proyecto se trata el despliegue del mismo en un servicio cloud, haciendo uso de un servicio IaaS para ofrecer un servicio de alta disponibilidad. Para contenerizar el proyecto se ha utilizado el servicio de host de repositorios **Docker Hub** y a su vez, se ha automatizado la construcción de contenedores al realizar actualizaciones en el repositorio de **GitHub**.

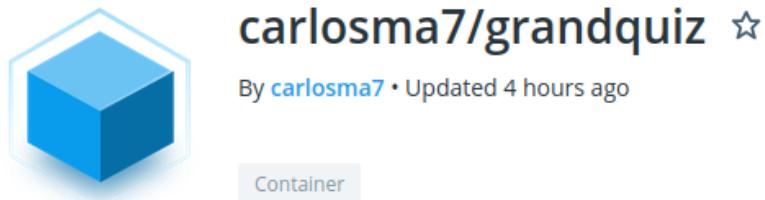


Imagen 52: Contenedor del proyecto en **Docker Hub**

Tras la contenerización del proyecto, el objetivo ha sido el de desplegar el proyecto en un máquina virtual con alta disponibilidad. Para ello se ha escogido una infraestructura provista por la compañía **Digital Ocean**, siendo esta máquina virtual un **Droplet**.



Imagen 53: Despliegue del proyecto en **Digital Ocean**

Por otro lado además se incluyen las encuestas de usabilidad a diferentes usuarios con el fin de valorar el proyecto.

## 8. Conclusiones y Trabajo futuro.

## 9. Bibliografía

- [1] Greg L. West, Benjamin Rich Zendel, Kyoko Konishi, Jessica BenadyChorney, Veronique D. Bohbot, Isabelle Peretz, and Sylvie Belleville. Playing super mario 64 increases hippocampal grey matter in older adults.
- [2] Kyoko Akimoto Susan Krauss Whitbourne, Stacy Ellenberg. Reasons for playing casual video games and perceived benefits among adults 18 to 80 years old, 2013.
- [3] Jeffrey Goldstein, Lara Cajko, Mark Oosterbroek, Moniek Michielsen, Oscar Van Houten, and Femke Salverda. Video games and the elderly. Social Behavior and Personality: an international journal, Volume 25, Number 4, 1997.
- [4] Aikaterini Papagianni Fotini Paraskeva, Sofia Mysirlaki. Multiplayer online games as educational tools: Facing new challenges in learning. Computers & Education, Volume 54, Issue 2, 2010.
- [5] Kurt Squire. Video games in education. Comparative Media Studies Department, 14N-205 Massachusetts Institute of Technology.
- [6] Residencia Divina Gracia. Juegos divertidos para personas mayores. [Link to article](#).
- [7] El Diario. Cinco juegos para estimular la memoria de los adultos mayores. [Link to article](#).
- [8] África María Cámara Estrella. El juego en las personas mayores: Una vía de desarrollo personal. Revista Portuguesa de pedagogía, 2012.
- [9] Sergio Sayago Josep Blat, Josep Lluis Arcos. Worthplay: juegos digitales para un envejecimiento activo y saludable. I+D En Envejecimiento: Proyectos Cero FGCSIC.
- [10] M. Elena Fabregat Cabrera, María Costa Ferrer, M. Teresa Romero Berenguer, and Rakel Poveda Puente. Juego como promoción de un envejecimiento saludable: definición del usuario y pautas para el diseño de producto accesible. Departamento de Pedagogía-Producto. Instituto Tecnológico del Juguete (AIJU). Alicante, España.
- [11] Petter Bae Brandtzaeg Asbjørn Følstad. Sig: Chatbots for social good. Extended Abstracts of the 2018 CHI Conference, 2018.
- [12] Farookh Khadeer Hussain Ebtesam Almansor. Survey on intelligent chatbots: State-of-the-art and future research directions. Complex, Intelligent, and Software Intensive, 2020.
- [13] Planeta Chatbot. Chatbots: análisis comparativo del diseño de un bot en telegram y en facebook messenger. [Link to article](#).
- [14] ChatCompose. Chatbots para whatsapp. [Link to article](#).
- [15] Planeta Chatbot. Desarrollando en alexa — parte 1. [Link to article](#).
- [16] Wikipedia. Wikipedia: Asistente virtual. [Link to article](#).
- [17] Claudé Pessoa Reab. ¿cómo los asistentes virtuales pueden ayudar a la tercera edad? [Link to article](#).

- [18] Luis F. López. Chatbots: Guía para empezar desde cero. [Link to article](#).
- [19] Sono Philipp. 7 top powerful platform tools to build the best chatbots. [Link to article](#).
- [20] 99signals. 8 best chatbot platform tools to build chatbots for your business. [Link to article](#).
- [21] Aaron Brooks. 10 best chatbot builders in 2021. [Link to article](#).
- [22] Botpress. Chatbots in the gaming industry. [Link to article](#).
- [23] Crhis Knight. Ai and chatbots heading into gaming territory. [Link to article](#).
- [24] Shopee Blog. 10 fun telegram games & bots that will revive your group chats. [Link to article](#).