UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"Aplicación web para compraventa de claves de videojuegos"

TRABAJO FIN DE GRADO

Septiembre - 2021

AUTOR: Jonathan García Ruíz
DIRECTOR/ES: Jesús Javier Rodríguez Sala
Miriam Esteve Campello

RESUMEN

Este proyecto está realizado con la intención de escalarlo para convertirlo en un modelo de negocio virtual, en la actualidad el uso de internet e informática no deja de aumentar y con ello también el consumo de videojuegos.

El proyecto busca adentrarse en ese nicho y ofrecer un servicio de compraventa de claves de videojuegos, lo cual vendría a ser el equivalente a comprar un videojuego en físico pero de forma digital; de esta manera, un usuario puede comprar claves en nuestra aplicación a un precio más asequible que el oficial o puede vender las claves que tenga sin utilizar.

El principio por el cual funciona es que hay distintas maneras de comprar keys o claves a un precio menor, por ejemplo, un juego no vale lo mismo en todos los países, el precio se ajusta en base a la economía del mismo. Otra manera es comprar bundles o paquetes de claves al por mayor, recibiendo su correspondiente descuento. La aplicación actúa como intermediario ofreciendo una plataforma a los usuarios para comprar y vender, donde se aplican pequeñas comisiones o tasas por este servicio.

La aplicación ha sido desarrollada principalmente mediante el lenguaje de programación Python y los frameworks Django y Material Design Bootstrap 4, además de otras tecnologías como HTML o CSS.

AGRADECIMIENTOS

Quiero decir que ha sido un placer compartir estos años de carrera con mis compañeros de clase, de mi promoción y de otros años así como los profesores, en especial a Jesús Javier Rodríguez Sala por permitirme hacer este trabajo con él como tutor y toda la ayuda y apoyo que me ha brindado con la realización del mismo.

A mi familia, sobre todo a mis padres, muchas gracias por criarme y hacer que sea lo que soy ahora y por ser comprensivos y apoyarme en todo lo que he hecho.

Tengo muchos recuerdos no tan buenos pero sobretodo muchos recuerdos buenos que son los que a lo largo de esta etapa más recuerdo y más nostalgia me da, sobretodo echo de menos el primer año de carrera, ese año tuvo algo mágico; hasta algunos recuerdos no tan buenos los recuerdo ahora junto con mis amigos de carrera y nos hace gracia. Me llevo buenos amigos, Carmelo, Adam, José y Francisco, os daría las gracias a vosotros pero vamos a seguir viéndonos así que os quedáis con las ganas.

Fuera bromas, muchas gracias a todos por haber estado ahí, de corazón, gracias.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1
Introducción

1.1.- IDEA DE APLICACIÓN

La idea de la aplicación es atacar un nicho de mercado que cada vez crece más, el cual es la compra-venta de videojuegos, en este caso, en formato digital. Será una aplicación no para una empresa u organización sino que se trata de un proyecto personal, pero con perspectivas de, a futuro, seguir trabajando en ello y poder convertir esta idea en un proyecto comercial.

Al ser un proyecto de compraventa online de videojuegos no se necesitaran muchos recursos para poder llevar a cabo el desarrollo del mismo, no se requieren almacenes físicos donde almacenar los productos, ni empleados a los que contratar, ni una tienda o un local físico donde exponer los productos para su comercialización; sólo hace falta una aplicación web y su correspondiente deploy o llevarla a producción. Para la realización de dicha aplicación web, al menos en una primera versión de prueba de concepto, tampoco es

necesario un equipo de desarrolladores que se dividan el trabajo, sino que un único desarrollador puede por sí solo desarrollar por completo una versión funcional de esta web de compraventa.

1.2.- JUSTIFICACIÓN DEL PROYECTO

La aplicación a desarrollar busca ofrecer un servicio para entrar en el mercado de compraventa de juegos en formato electrónico, el cual es cada vez más usado, sustituyendo al formato físico clásico.

1.2.1.- Crecimiento de Internet

Desde la llegada de la World Wide Web basada en la arquitectura Cliente-Servidor, (en el cual se basa el presente proyecto como comercio electrónico), Internet no ha hecho más que crecer.

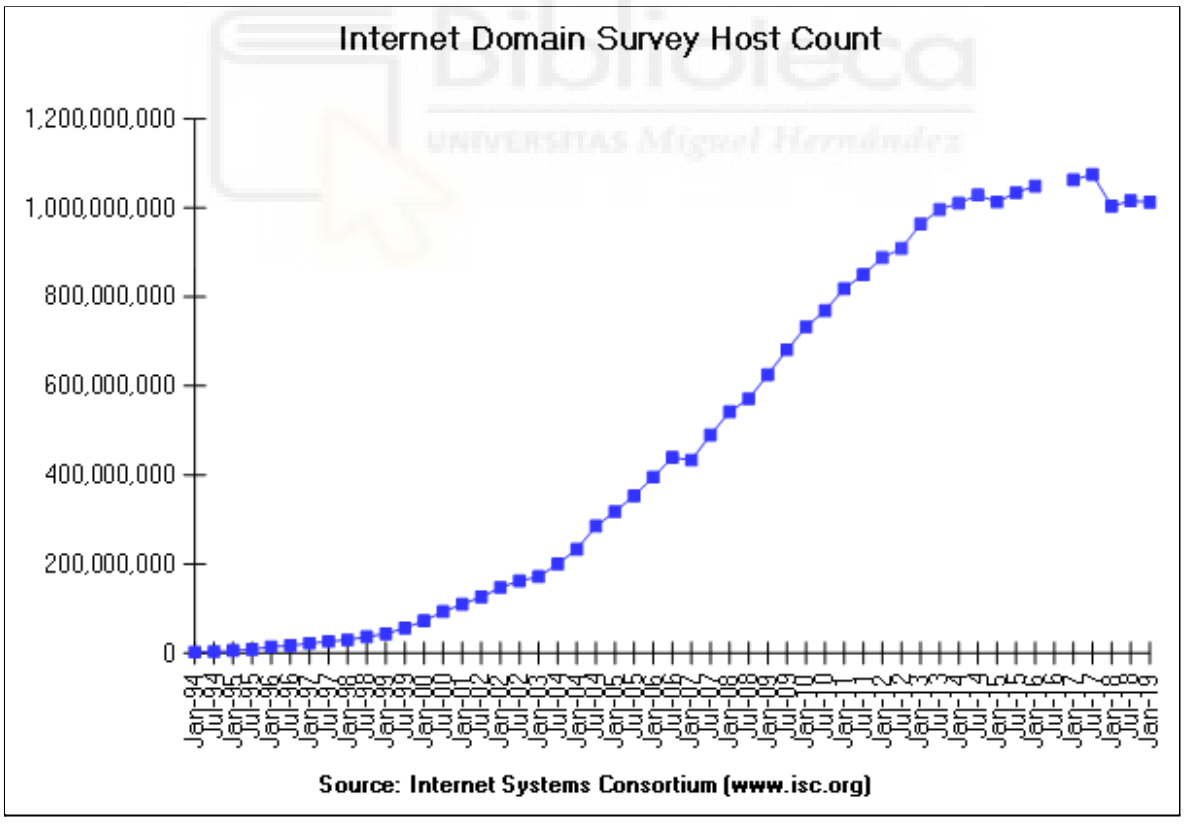

Figura 1.1 - Número de hosts a lo largo de los años desde que se funda W3C [1]

El teletrabajo en España ha aumentado con el COVID-19 de un 5% a un 34%. Por otra parte, encuestas realizadas a diferentes empresas reportaron que al menos el 68% de estas

empresas mantendrán este modelo de trabajo. Por último, no sólo el teletrabajo sino obviamente las compras online, las cuales incrementaron al 66% (un aumento del 16% respecto a su etapa pre covid) y que, tras el confinamiento, ha seguido aumentando [4].

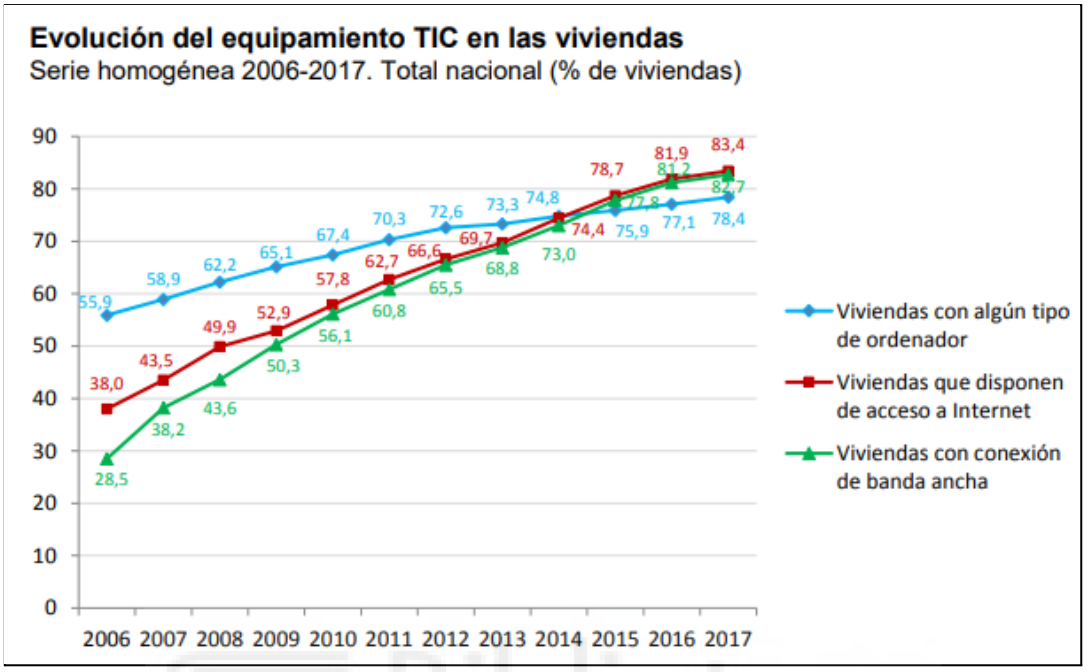

Figura 1.2 - Crecimiento de las tecnologías en las viviendas de España [2]

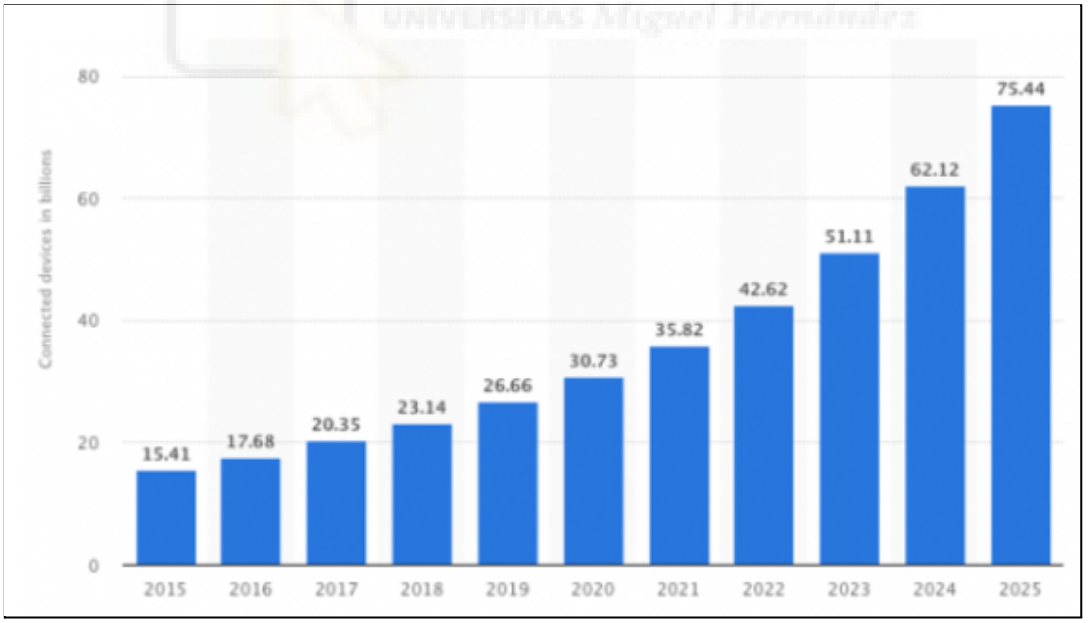

Figura 1.3 - Crecimiento habido y esperado del IoT [3]

Teniendo en cuenta estas figuras, se puede ver que Internet es un "*monstruo*" imparable que no hace más que crecer, y eso es un gran factor a tener en cuenta ya que hace que inevitablemente tengamos más tráfico en la red y en las aplicaciones.

1.2.2.- Transición del formato físico al digital

Teniendo en cuenta del apartado anterior, que hay cada vez más usuarios en Internet, y que el consumo vía streaming en plataformas como Youtube, Twitch, etc. aumenta constantemente, se entiende que aumente la venta de videojuegos, sobre todo en formato digital. Una analogía para entenderlo mejor, podría ser Netflix, cada vez hay más gente que contrata el servicio de streaming de Netflix en vez de comprar la película en formato físico/digital [5][6][7].

En el ámbito del los videojuegos, esta tendencia viene desde antes del auge del streaming, gracias a la plataforma Steam, la cual permite introducir Keys (claves) de videojuegos, las cuales permiten activar dichos videojuegos, y por tanto equivalen a tenerlos en propiedad. Lo que inició Steam lo están imitando las grandes empresas de videojuegos como Blizzard, Epic Games, entre otras. Por otra parte, cada vez son menos frecuentes las unidades ópticas en los PCs, por este mismo motivo. En consolas tenemos el mismo resultado, tanto es así que SONY tiene una versión de su última consola sin unidad óptica, ya que la empresa aboga cada vez más por su servicio de streaming PlayStation Now, el cual cuenta con un catálogo de más de 600 videojuegos; también pasa esto con su gran competidor, Microsoft con Xbox Game Pass con una biblioteca de más de 100 juegos para consola y PC. Amazon, Apple con Arcade, Google con Stadia, también están trabajando en su servicio de streaming [5][6][7].

Figura 1.4 - El negocio digital vs el físico del videojuego [7]

Como síntesis, la justificación del proyecto es la viabilidad económica al tratar con un mercado, el de las claves de videojuegos, que no hace más que crecer.

1.2.3.- Adicción a los juegos mediante la dopamina

Nuestro cuerpo, tanto mental como físicamente, apenas ha evolucionado desde la prehistoria. Es importante dejar este estancamiento claro y sobre todo servirá para explicar con ejemplos el problema que supone este estancamiento en nuestros tiempos.

La dopamina es un neurotransmisor que libera nuestro cuerpo ante las cosas que cree que son importantes para nuestra supervivencia. A diferencia de lo que opina mucha gente, este neurotransmisor no es liberado tras hacer una tarea y percibirla como recompensa, sino que se libera para hacer la tarea. Una de las cosas que hace es controlar tu forma de pensar; si tenemos un mal hábito, pero que nos resulta "cómodo" o "agradable", nuestro cerebro mediante la dopamina buscará 100 excusas para justificar y mantener ese mal hábito.

Hay estudios [8] que demuestran cómo ratas sin dopamina, simplemente pierden la motivación de actuar y mueren de hambre, ya que, comemos porque nuestro cuerpo libera dopamina para motivarnos a comer.

Antiguamente, se liberaba dopamina para reproducirse, socializar, comer, cazar, recolectar y construir. Mientras que nuestro cuerpo sigue trabajando bajo este esquema siendo por tanto, un sistema que no ha evolucionado, mientras que la tecnología sí lo ha hecho y a un ritmo exponencial. La psicóloga Deirdre Barret afirma que la tecnología produce en los seres humanos lo que ella llama "super estímulos", los cuales son estímulos artificiales pero más intensos que los naturales [9], y que al consumir estos "super estímulos", perdemos la necesidad de consumir el estímulo natural.

Según esta autora, la pornografia es un sustituto de la reproducción, ya que no tiene procreación con fines de supervivencia; la comida basura es un sustituto de la comida sin procesar o natural, la cual se puede conseguir sin esfuerzo de recolección y preparación alguno; las redes sociales son un sustituto de socializar con tu entorno más cercano, que vendría a ser el equivalente a la tribu; y por último, los juegos son un sustituto de caza, recolección, construcción, socialización y mejora como individuo, dependiendo del género o géneros del juego.

Entrando más en detalle, en el campo de los juegos hay muchos géneros y cada uno viene de nuestros deseos ocultos como especie. Algunos ejemplos:

- Los juegos de horror y monstruos son una simulación de los depredadores de antaño.

- Los First Person Shooter como Counter Strike, Far Cry, son una simulación de caza y la lucha contra tribus o bandos enemigos.

- Los juegos deportivos como FIFA o NBA2K (entre otros muchos) también son una simulación de escenas de caza y batalla.

- Los RPG como World of Warcraft son una simulación de progreso y evolución.

Tenemos estos deseos ocultos porque son los que nos brindaban una mejor posición en la tribu, acceso a más recursos, supervivencia y perpetuación de la especie.

Hay cuatro motivos principales por los que nuestra mente prefiere los super estímulos a los estímulos normales y que provoca que nuestra dopamina se centre exclusivamente en estos super estímulos haciendo que perdamos el interés en otros estímulos como socializar en la calle, desarrollo personal, hacer deporte o ir al gimnasio, comer saludable, etc.:

Motivo 1) Se trata de "super" estímulos, por tanto liberan más dopamina que los estímulos normales.

Motivo 2) Es muy fácil acceder a ellos (un par de clicks con el dedo y ya entramos a Instagram, donde solo tenemos que deslizar el dedo hacia arriba o hacia abajo para "socializar") o pasamos de la idea de estudiar a la idea de querer jugar porque no queremos seguir y hacerlo realidad en cuestión de segundos, con la PS5 basta con coger el mando, pulsar el botón 1s y el inicio de sesión es tan rápido que sueltas el dedo y ya lo tienes todo listo para abrir el juego.

Motivo 3) No necesitamos salir de nuestra zona de comodidad, no nos supone ningún riesgo o peligro ya que estamos agusto en nuestra casa sin exponernos al mundo real (Cazar un tigre en la vida real vs Cazar el tigre albino más grande de la manada con dientes el doble de grandes que los de uno normal).

Motivo 4) El cerebro no diferencia entre recompensa a largo plazo como lo es el gimnasio o recompensa a corto plazo como lo son los juegos MMORPG en los cuales tu personaje progresa y mejora su equipamiento mucho más rápidamente. Percibe neurológicamente ambas opciones como recompensas y estamos mentalmente programados para elegir el camino con menor resistencia.

Entonces, si tenemos en cuenta lo dicho, las personas, al consumir estos super estímulos, perdemos la motivación por las experiencias reales, y unos nos llevan a otros, nos rodean por todas partes pero sobretodo, cada vez son más famosos y demandados, y nos mantienen en un estado de procrastinación casi continuo.

Los juegos son cada vez más realistas por lo que la liberación de dopamina aumenta y con ella la adicción, sin mencionar que se estudian maneras de mantener atrapado al jugador el mayor tiempo posible, evitando, por ejemplo, pantallas de “GAME OVER”, muy clásicas en los comienzos de esta industria.

El psicólogo B. F. Skinner descubrió en una de sus investigaciones “El condicionamiento operante”, es un proceso en el que se ejerce control sobre la conducta de un organismo controlando las variables y el ambiente en el que se encuentra. Esta investigación consistió en encerrar una paloma en una caja con un disco dentro, esta caja y lo que es ahora la conclusión del experimento se le conoce como “La caja de Skinner”. Descubrió que podía asociar en cuestión de minutos que para la paloma, picotear el disco acababa en recompensa, la de comer. Pero no se quedó ahí y refinó su estudio, haciendo que lo asociara con, no picar el disco por picar, sino cada 10 veces, o sólo echar comida 1 vez por minuto. Al final, el psicólogo extrapoló los resultados a la conducta humana y, en especial, con la adicción al juego [10].

En su tiempo, usó este estudio para explicar por qué los casinos generan adicción, al parecer, los jugadores patológicos asociaban el tirar de una palanca a ganar dinero tarde o temprano, esto es debido a que ven que tirando de la palanca, no siempre cae recompensa, pero cae, y el hecho de que caiga es el motivo de que se cree este condicionamiento operante, pero va más allá, el hecho de que sea aleatoria, incrementa aún más la dopamina que se libera [10].

Todo este mecanismo se ha refinado en los videojuegos actuales, los cuales ofrecen recompensas por jugar, por hacer ciertos logros, incluso por sólo conectarte al juego. También cada vez más juegos incluyen, igual que en los casinos, cofres que cuestan dinero virtual muy difícil de conseguir a no ser que los compres con dinero real, casi obligándome a ello, en los cuales ofrecen grandes recompensas aleatorias. El ex investigador de Ubisoft Nick Yee, condujo estudios demostrando que entre más horas pasamos jugando videojuegos, mayor autodesarrollo comenzamos a derivar de los videojuegos en lugar de la vida real. Al aprovechar estos sistemas de recompensas casi infinitas, los videojuegos se empiezan a parecer más a las Cajas de Skinner que a fuentes de entretenimiento inofensivo.

En 2019 la OMS añadió un nuevo desorden a su lista, que se prevé entre en vigor en 2022; el trastorno por videojuegos, el cual lo definen como: “un patrón de comportamiento de juego persistente o recurrente, que puede ser en línea o fuera de línea, manifestado por un control deficiente sobre el juego, aumentando la prioridad otorgada al juego sobre otros intereses de la vida y actividades diarias, y la continuación o escalada del juego a pesar de la ocurrencia de consecuencias negativas” [11].

Por otra parte, si tenemos en cuenta el crecimiento de Internet, el refinamiento de las técnicas de marketing digital, incluyendo la publicidad de juegos desde otros super estímulos como las redes sociales (Twitter, Facebook, Instagram, Discord) ya sea de parte de una empresa o de los propios jugadores haciendo inconscientemente publicidad gratuita al compartir sus experiencias jugando cierto juego; juntando todo esto vemos que es una industria que va a seguir creciendo indudablemente, por motivos puramente biológicos.

1.3.- OBJETIVOS

El objetivo principal del presente trabajo es el de desarrollar una aplicación completa y funcional que me permita aprender ciertas herramientas y tecnologías que no se estudian en la carrera como Python, Django, Heroku, PostgreSQL entre otras; para ampliar los conocimientos que poseo y ser más versátil como desarrollador. Por otra parte, tengo especial interés en aprender Python y ampliar mis conocimientos de JavaScript porque me gusta la forma en la que están planteados, el innumerable número de usos que se le puede dar en diferentes tipos de ámbitos y su creciente uso en estos últimos años.

Como objetivo secundario me planteo presentar una idea de aplicación que pueda ser explotada comercialmente, quiero ir cogiendo soltura con la creación de aplicaciones web ya que me parece una manera rápida y barata de montar un negocio a partir de una idea, que es algo que quiero intentar en un futuro para montar negocios hasta que uno sea exitoso. En el presente proyecto, quiero desarrollar una aplicación web para compraventa de claves (o keys) de activación de videojuegos online. En el capítulo 2 se hace un estudio de otras webs que ya comercializan este tipo de productos.

1.4.- LÍMITES DEL PROYECTO

Como se ha dicho anteriormente, el presente proyecto es meramente académico, no se trata de desarrollar una aplicación web para un cliente real con unos requerimientos concretos que satisfacer. Al ser un PMV no se van a desarrollar plenamente las funcionalidades completas que debería tener un comercio electrónico, aunque sí se van a implementar una parte de la funcionalidad.

Capítulo 2 Antecedentes y estado de la cuestión

2.1.- PRODUCTO MÍNIMO VIABLE

El objetivo de este trabajo es presentar un "Producto Mínimo Viable" (PMV), de tal manera que su evaluación sirva para obtener feedback de un posible cliente y/o buscar posibles inversores. Pero, ¿qué es un producto mínimo viable? Un producto mínimo viable es descubrir las necesidades del cliente y si está dispuesto a utilizar tu producto para resolverlas, con el menor coste posible. Para esto, se desarrolla una versión simplificada del producto final, lo suficientemente funcional para que se vea visualmente lo que se quiere conseguir. De esta manera, con un coste mínimo, se puede averiguar si nuestro modelo de negocio tiene sentido o no a la vez que aprendemos del mercado y de la psicología del consumidor. El objetivo es, realmente, evitar construir un modelo de negocio completo para luego ver que nadie lo va a usar, cuando se podría haber evitado o mejorado con el correspondiente prototipo. De esta manera, confirmamos rápidamente si habría demanda o no. [15]

Esto se basa en la metodología Lean Startup desarrollada por Eric Ries con su libro “The Lean Startup” [16]. Se debe buscar la mayor cantidad de "Early Adopters" para probar nuestro producto, en este caso el jurado tendrá que hacer ese rol. Un early adopter es un perfil de cliente que es fácil de identificar; es el tipo de persona que tiene un problema o necesidad claramente identificado, no tiene aversión al riesgo y está dispuesto a probar/comprar tu producto porque le gusta probar novedades y realmente tiene una necesidad por solventar su problema. Son los típicos que compran todo lo novedoso, sobre todo si es tecnológico, como el último móvil, la última consola, el último periférico, etc. [17]

Una vez el producto mínimo viable haya sido analizado, lo más importante es aprender del feedback obtenido, si podemos continuar adelante, cambiar el enfoque, etc. [18]. El feedback o información que tengamos, se debe intentar parametrizar, para ello, se usa el concepto de métrica. Las métricas recopiladas no pueden ser vanidosas, como por ejemplo, cantidad de vendedores registrados o número de visitas en la página web. Las métricas que realmente aportan valor son aquellas que te permiten tomar decisiones una vez obtenido el dato en cuestión. Por ejemplo, la relación vendedor-visitas a la página. Gracias a estas métricas, es más exacto y seguro tomar decisiones como corregir un problema y volver a comprobar el PMV con los cambios realizados. Cuando nos aferramos demasiado a una idea inicial y no hacemos caso a los resultados de los experimentos, entonces no merece la pena llevar a cabo la experimentación a través de un Producto Mínimo Viable [15]. De esta manera, mediante este ensayo-error se va refinando y depurando el producto de manera eficaz, rápida y con costes mínimos.

Visto lo que es un PMV, el presente proyecto se centra en ofrecer un servicio en el sector de los videojuegos virtuales (Keys), para que los usuarios puedan comprar y vender estos productos. Por otra parte, este proyecto no incluye la búsqueda de un inversor, ya que es un proceso que consume más tiempo que recursos para su puesta en marcha. Es decir, se busca presentar un PMV para ser evaluado, continuado y poder convertirse en un negocio.

La plantilla de trabajadores estaría formada por mí exclusivamente, con una posible futura ampliación de personal, para mantener los costes lo más bajo posibles. Para sus inicios, bastará con emplear servicios gratuitos o muy baratos como los que ofrece Heroku para hacer de host de la aplicación y de la base de datos. Cuando sea necesario su ampliación si será necesario invertir más dinero, sin embargo, si esto sucede es que el producto está generando ingresos también. El mayor coste inicial estaría en contratar los servicios de un abogado especializado en conocimientos informáticos, para asegurar la viabilidad legal del proyecto, así como alguna campaña de marketing para darlo a conocer.

El principal problema del proyecto, es que en su inicio va a ser inferior a la competencia (la cual analizaremos más adelante) y que necesita mucha credibilidad para que los

usuarios se sientan seguros al comprar o vender. Al ser un producto inferior, se busca apostar por su sección de blogs y chat, el objetivo es crear un entorno amigable el cual funciona mejor si la afluencia de usuarios es menor. Cuanto más crezca el proyecto menos eficaz será y volveremos a no tener nada con lo que competir con las otras páginas, sin embargo, conforme eso vaya pasando seguiremos apostando por ese sentimiento de "Comunidad", mejorando la sección de blogs y chat orientándose con un enfoque de foro, donde se pueda debatir sobre temas de todo tipo.

2.2.- HERRAMIENTAS DISPONIBLES EN EL MERCADO

Este proyecto no cubre un nicho de mercado no explotado, sino que viene a competir con otras opciones las cuales vamos a analizar en los siguientes apartados. A continuación analizaremos las más relevantes en la actualidad, en qué destacan, analizar cómo funcionan y por qué funcionan, ver sus pros y contras, a la vez que dar una valoración de las mismas.

2.2.1.- G2A.COM

G2A.COM [12] es la primera herramienta que se va a ver y también es de las más famosas a la vez que antigua. Como se muestra, no sólo permite que los usuarios vendan y compren keys, sino todo tipo de hardware, periféricos y complementos, por lo que no se limita a trabajar con productos virtuales siendo más parecido a páginas como PCComponentes [13] o Amazon [14] donde se ofrece un delivery de todo tipo de productos.

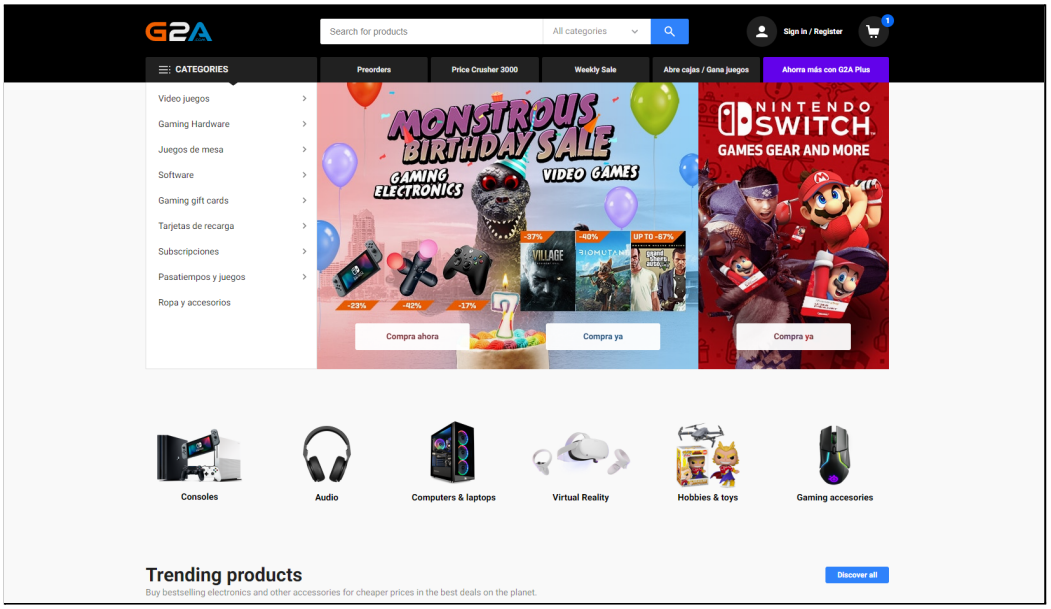

Figura 2.1 - Home Page o Index de G2A.COM

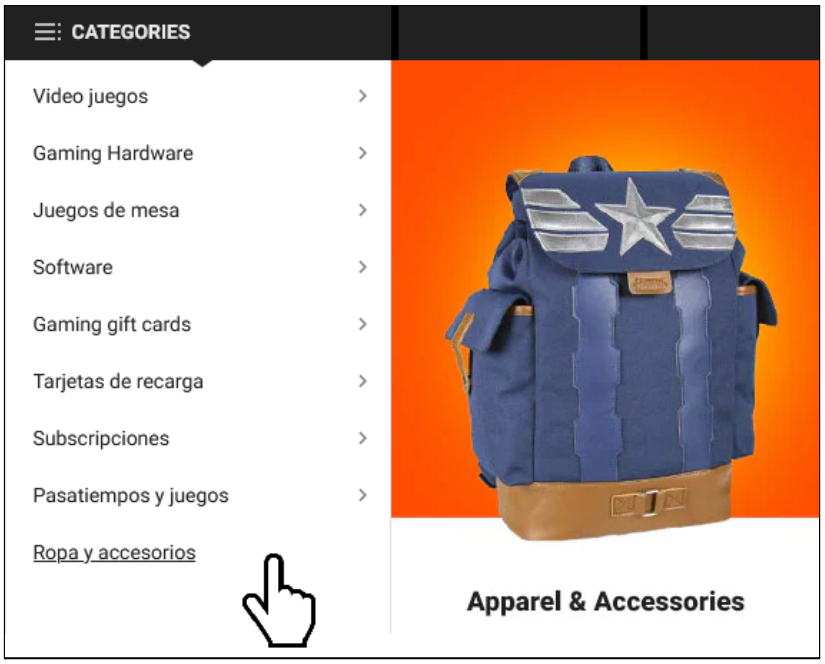

Figura 2.2 - Menú de categorías de G2A

Como puede observarse en el menú de categorías (figura 2.2) existe una gran variedad de productos disponibles. Además de los videojuegos se pueden comprar otros productos digitales como Windows 10 o Microsoft Office, o Gift cards, las cuales son ideales para regalar dinero en cierta plataforma como XBOX Live, PlayStation o Steam. Parecido a las gift cards, también tenemos las tarjetas de suscripción, las cuales permiten comprar cierto servicio por cierto tiempo, como el XBOX Live GOLD, Spotify o como hablaremos más adelante, G2A PLUS.

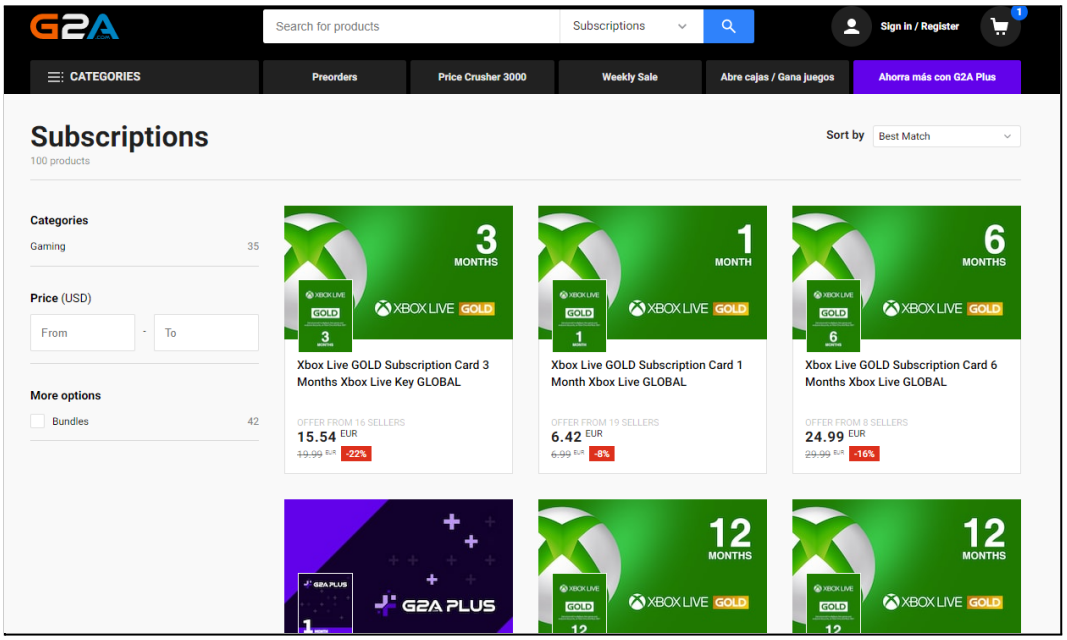

Figura 2.3 - Subscriptions

Por último, también tendríamos Merchandising como Funko pops, ropa, carteras, mochilas, juguetes como helicópteros, drones, cámaras y todo lo relacionado con el contenido audiovisual/informático.

Con esto, se ha cubierto los tipos de productos que venden, pero no cómo los venden. Se pueden comprar juegos que no han salido todavía en Pre orders o entrar en Weekly sale donde el objetivo es vender los juegos que están de oferta (con mayor oferta de lo normal). Como vemos en la Figura 2.4, este es el menú de compra de cierto juego, es interesante observar si se puede activar en el país en el que se reside, la plataforma del juego (PC, PS4, etc.), el envío (el cual puede ser inmediato o tardar unos días) y de que vendedor queremos comprar, pero esto es lo de menos ya que automáticamente se pondrá la key más barata de un vendedor con buena reputación. La compra no es inmediata sino que se agrega al carrito, el cual veremos más adelante.

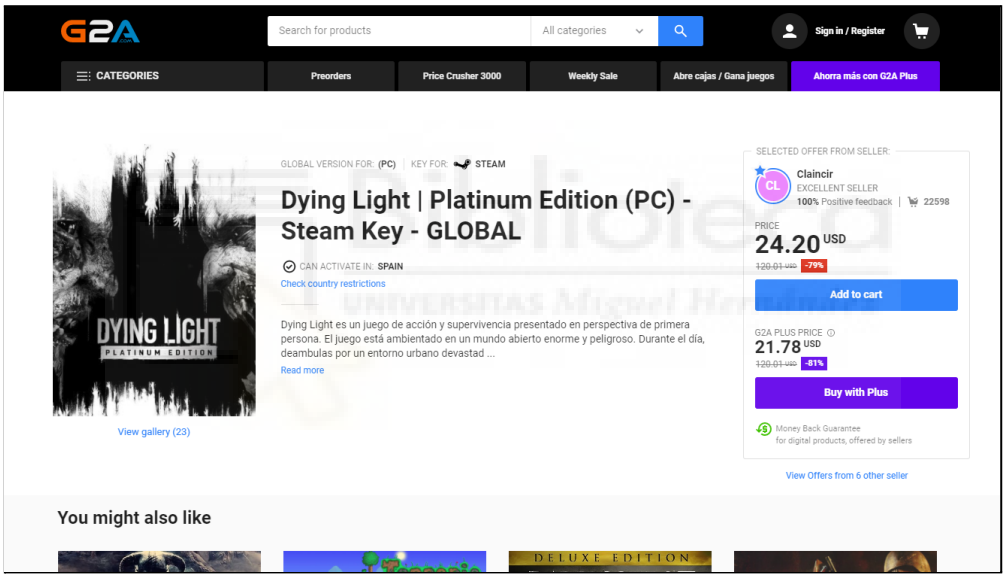

Figura 2.4 - Menú compra de juego

En cuanto al envío, este puede ser inmediato, el cual es la opción más común, donde el vendedor ya dispone de dicha key y la opción menos común que es envío (sin que sea inmediato), en esta opción el vendedor no dispone de la key en ese momento pero si tiene una facilidad de conseguir más keys y es el propio sistema de G2A que se encarga de ir revisando cuando el vendedor meta keys a la página para repartirlas a los de envío no inmediato. En caso de que el vendedor no introduzca keys, el cliente debe reclamar abriendo un ticket a soporte, G2A se pondrá en contacto con el vendedor y en caso de no llegar a un acuerdo es posible un reembolso.

También se observa que hay 2 tipos de compra, una normal y la compra con G2A PLUS, que consiste en una suscripción la cual puede ser pagada mensualmente o anualmente. Con ella se tiene acceso a juegos de manera gratuita cada mes, sin embargo, ellos eligen los

juegos que dan de manera gratuita a sus usuarios PLUS dentro de un loot box, no pueden ser elegidos por el usuario. Hay descuentos más amplios en todos los productos (10% como máximo) y la suscripción se puede cancelar en cualquier momento.

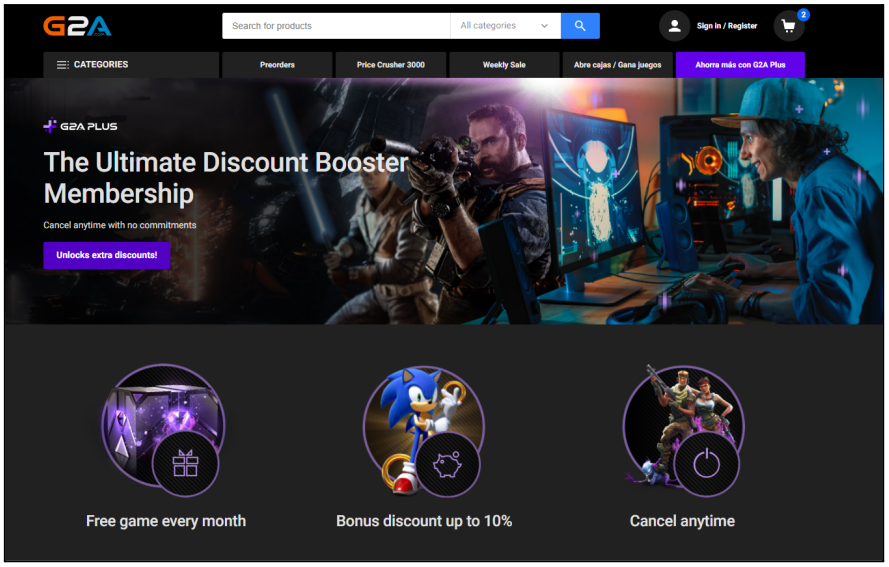

Figura 2.5 - G2A PLUS

Como vemos en la figura 2.6, el carrito de la compra puede incluir cualquier tipo de producto de los vistos anteriormente, el tipo de pago (normal o plus) e incluso variar la cantidad, automáticamente (cogerá la siguiente key más barata disponible). Lo interesante es que se puede comprar sin registrarse, con tal sólo poner un correo electrónico, se recibirá un email con la información de los productos comprados.

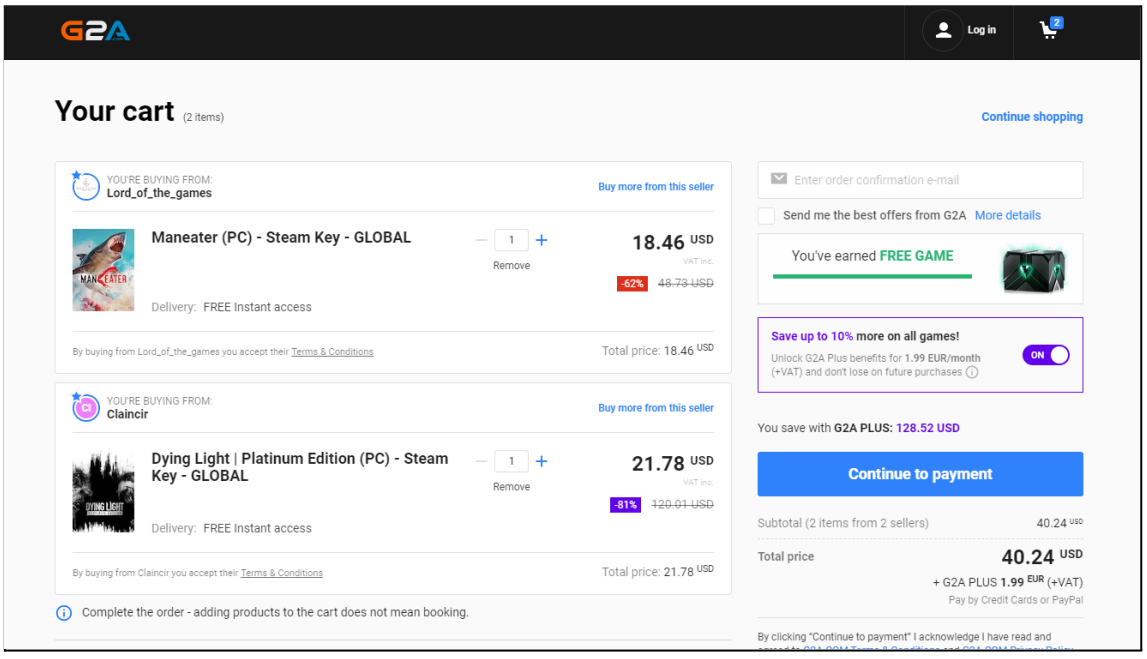

Figura 2.6 - Carrito de la compra

Se ha mencionado antes el término loot box. Esto tiene mucho que ver con lo expuesto en la introducción, las apuestas de azar (máquinas tragaperras, loot box) generan mayor adicción y esto genera más dinero. Básicamente en una de estas "cajas" hay un cierto número de productos "aleatorios", cada uno con su respectiva probabilidad de que toque, cuanto más caro sea el producto menos probabilidades hay de que salga ese producto, se podría decir que es una especie de "*tómbola*".

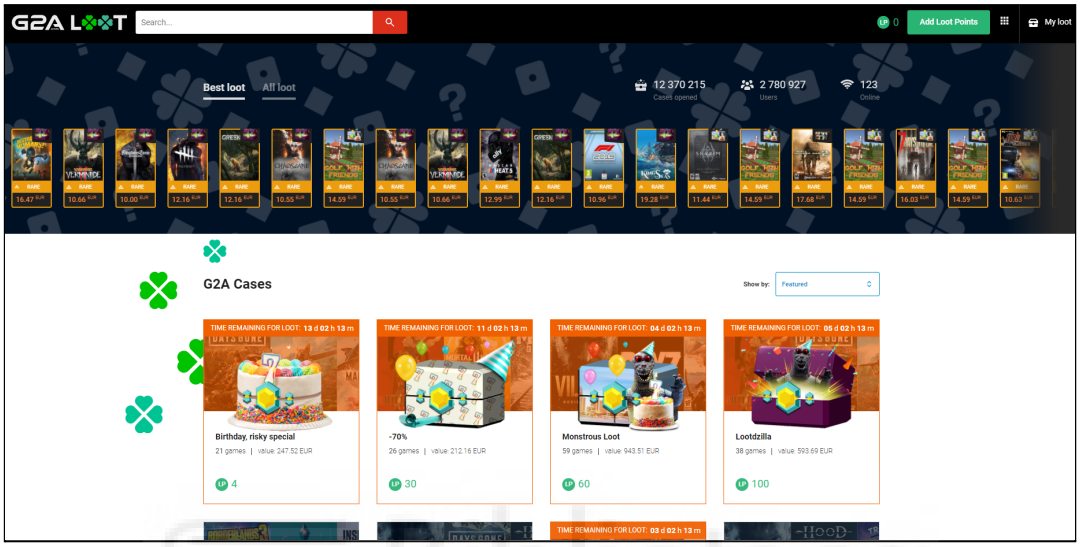

Figura 2.7 - G2A.COM LOOTBOXES

Este sistema, el cual es muy rentable, es relativamente nuevo, y está inspirado en los pioneros de los loot boxes, Valve, con sus cajas de armas en el Counter Strike: Global Offensive (CSGO). Al ver su rentabilidad, todo tipo de páginas está incorporando este sistema y G2A no se ha quedado atrás.

Por último, mencionar su herramienta G2A PAY, la cual se basa en la empresa ZEN.COM (ver figura 2.8), para ofrecer a cualquier e-commerce una pasarela de pago.

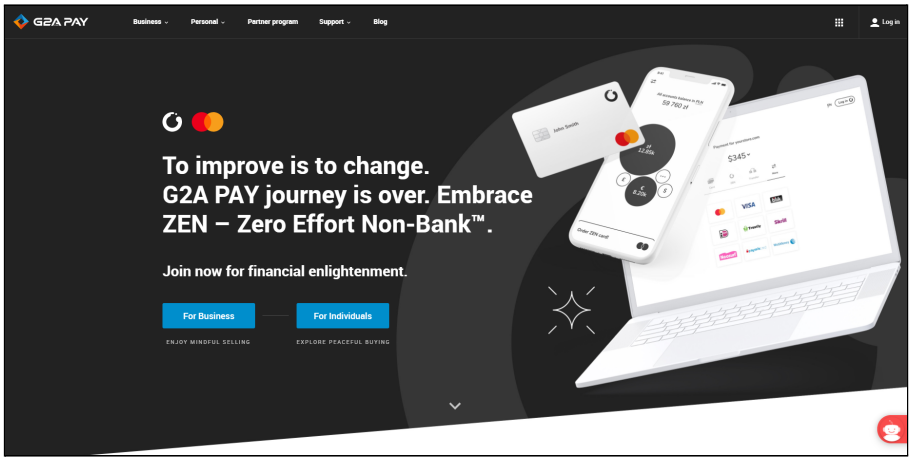

Figura 2.8 - G2A PAY

2.2.2.- Instant Gaming

Instant Gaming [19] es otra página de las más antiguas y que no se ha hundido sino todo lo contrario, es de las más grandes y conocidas junto con G2A.COM. Como vemos en la figura 2.9 se ve una página más "minimalista" y más orientada a sólo videojuegos a diferencia de su competidora G2A.

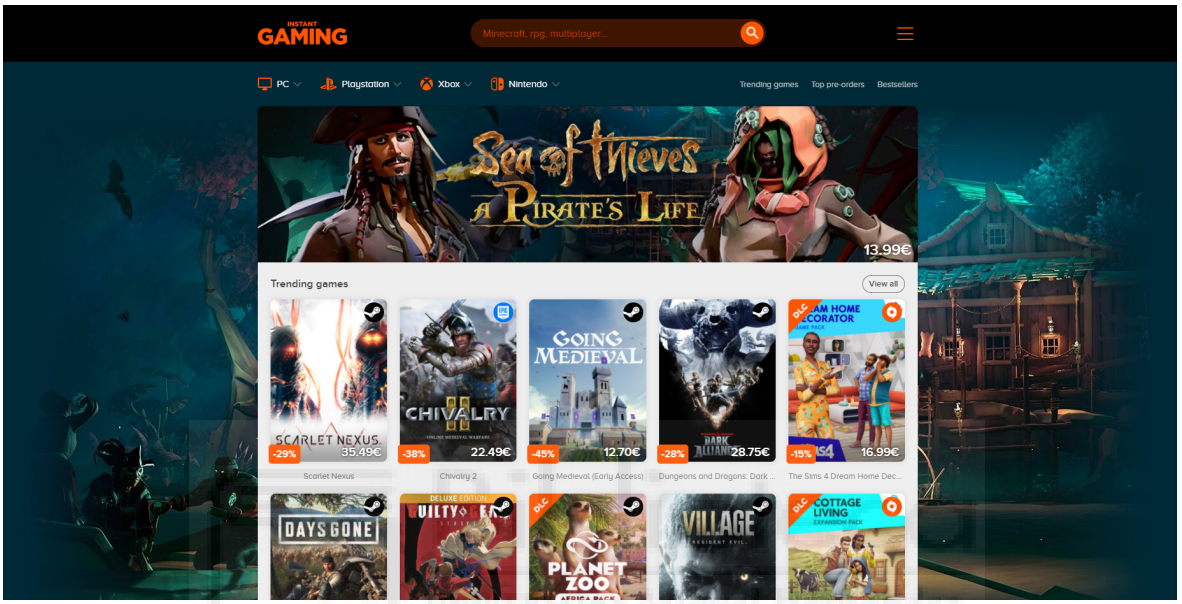

Figura 2.9 - Home Page de Instant Gaming

Como se centra más en juegos que en otro tipo de productos, hace una distribución acertada en cuanto a la plataforma del cliente, como se aprecia mejor en la Figura 2.10

Figura 2.10 - Distribución de plataformas en el Navbar

Esto me parece acertado porque, entre las compañías (PlayStation, Xbox, Nintendo) hay una constante batalla por "fidelizar" a sus clientes, de tal forma que usuario de Xbox, no use PS o Nintendo, llegando a ser extremo el fanatismo o el odio a la competencia en algunos casos. En PC no pasa apenas eso. Por otra parte, para un usuario de a pie, con los conocimientos justos de informática pero que le guste el mundo de los videojuegos, esta división puede resultar más apropiada para el mismo.

Como se muestra en la Figura 2.11 al pulsar sobre una de las plataformas, ordena los juegos en base al programa que se vaya a usar para activar la Key, ya que comprar una Key de Steam sólo funcionará en Steam y no en las otras. No obstante, no sólo hay juegos exclusivamente, sino que también hay tarjetas prepago como se muestra en la Figura 2.12.

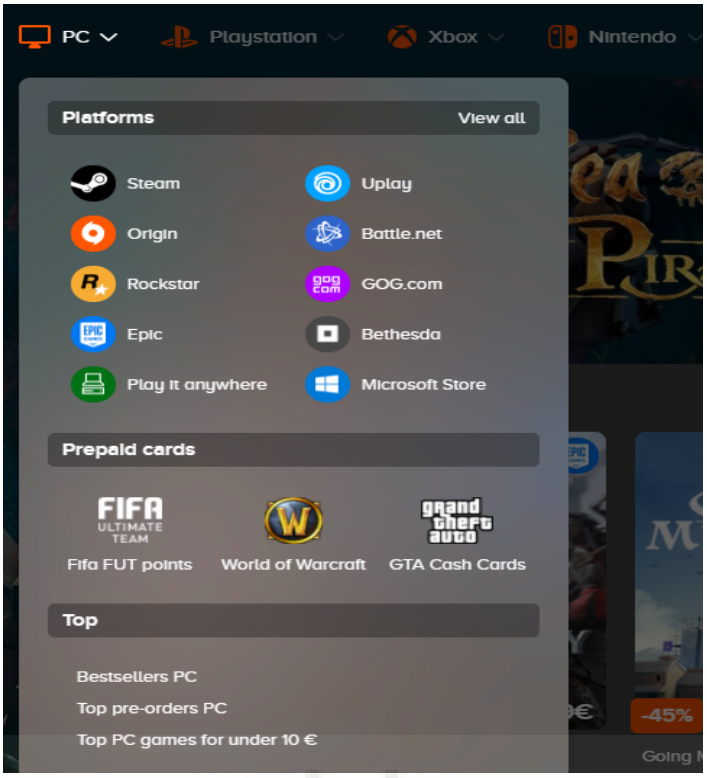

Figura 2.11 - Desplegable PC

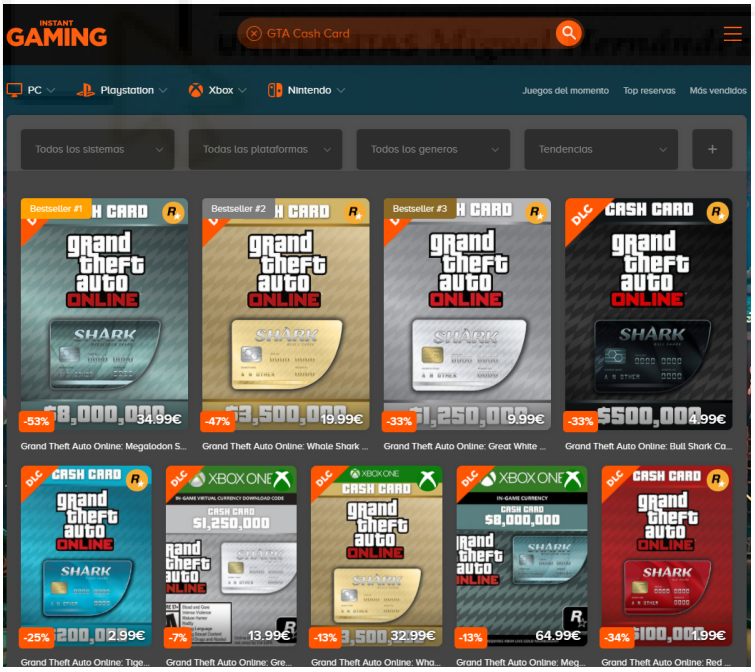

Figura 2.12 - Tarjetas prepago de GTA Online

Desde el Home, tenemos las siguientes opciones: Juegos del momento, Top reservas, Opinión de los usuarios, Más vendidos, Partners, Reviews de jugadores, Extensión, Ofertas semanales. Cada sección muestra unos ejemplos y un botón para ver todos.

En juegos del momento, top reservas, más vendidos y ofertas semanales simplemente enseñan juegos en base a la agrupación correspondiente de cada sección, lo interesante son las secciones Opinión de los usuarios, Reviews de jugadores y la Extensión. La zona de la extensión es un enlace a la extensión que han creado para Google Chrome, ésta añade un enlace en la página de Steam si el juego que se está viendo en Steam está en Instant Gaming.

Como vemos en la Figura 2.13 se muestran los últimos comentarios y la puntuación que han establecido, clicando sobre el comentario te llevan al juego que han comentado, no obstante, es confuso que salga el nombre del juego donde según los estudios en accesibilidad web el usuario espera encontrar el nombre del que comenta, no el nombre del juego, de hecho es lo que me ha pasado a mí. Como vemos el último comentario sería "nice" para el juego DbD (Dead by Daylight).

Figura 2.13 - Opinión de los usuarios

Clicando en Ver todo, podremos ver todos los comentarios habidos y por haber como muestra la Figura 2.14, aquí se muestra al contrario, ya no muestran el juego sino el nombre del usuario, para saber de qué juego hablan hay que pasar el ratón por encima y ver a dónde te llevaría el enlace en la parte inferior izquierda.

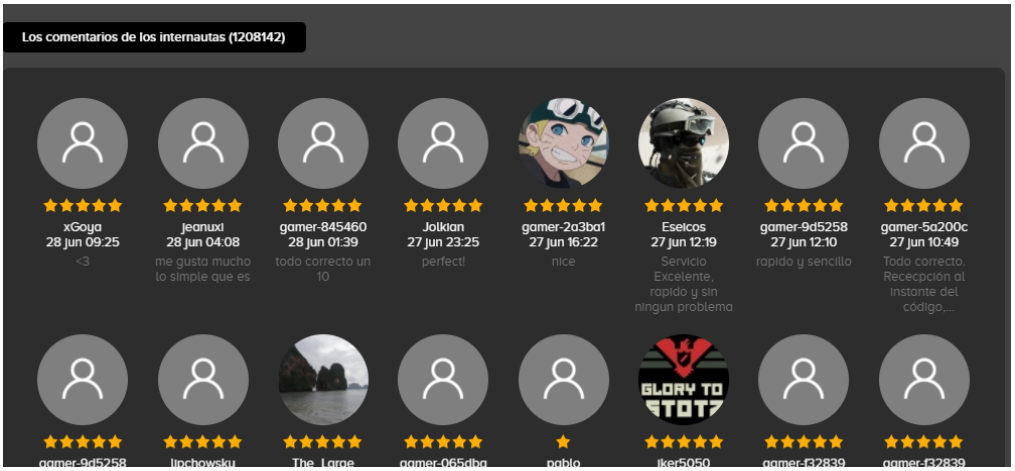

Figura 2.14 - Todos los comentarios de Instant Gaming

En la Figura 2.15 vemos la sección de Partners, un Partner promociona tu página más o menos en base al contrato que hagas con dicha persona y la relevancia de la misma. Este sistema es muy eficiente, ya que el que patrocina lo hace desde plataformas donde está nuestro público objetivo, Twitch, Youtube, etc. Además, genera confianza el ver que tu streamer favorito trabaja con X empresa, hasta puede que compren sólo por contentar al streamer, algo equivalente a una donación pero ambos ganan y por tanto, Instant Gaming también gana.

No obstante, al pulsar en Conviértete en Partner, te lleva a una sección muy parecida a lo que sería un FAQ, pero rápidamente te ubicas. Una afiliación desde Twitch, sería algo así (Figura 2.16).

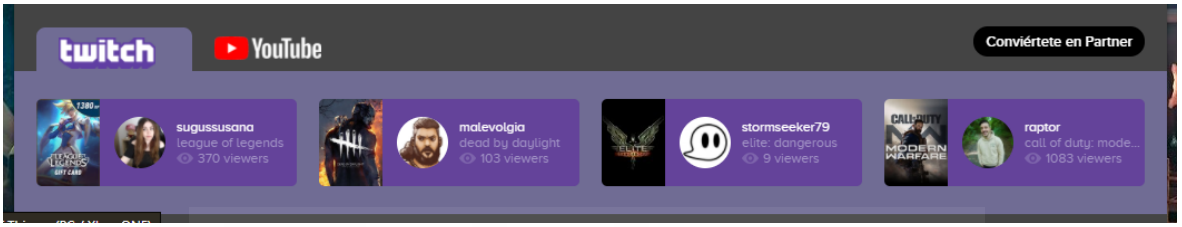

Figura 2.15 - Partners

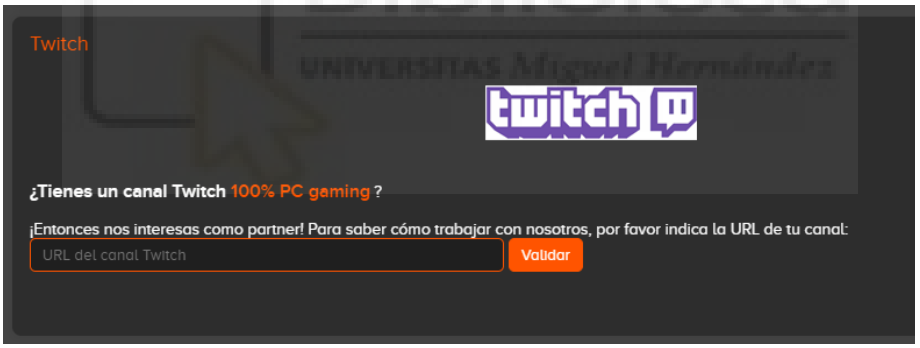

Figura 2.16 - Afiliación con tu canal de Twitch

La última sección es reviews, aquí podemos ver las últimas reseñas sobre los juegos, ya que permite dar una opinión o una review del mismo. En este caso, puedes hacer una review sin necesidad de haber comprado el juego (Figura 2.17), sin embargo, una vez realizada debe ser validada, cuando es validada se le informa al usuario por correo.

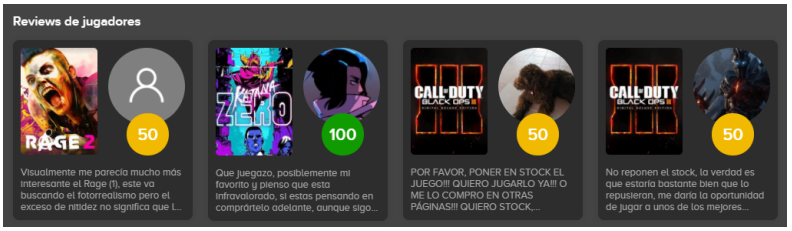

Figura 2.17 - Reviews de jugadores

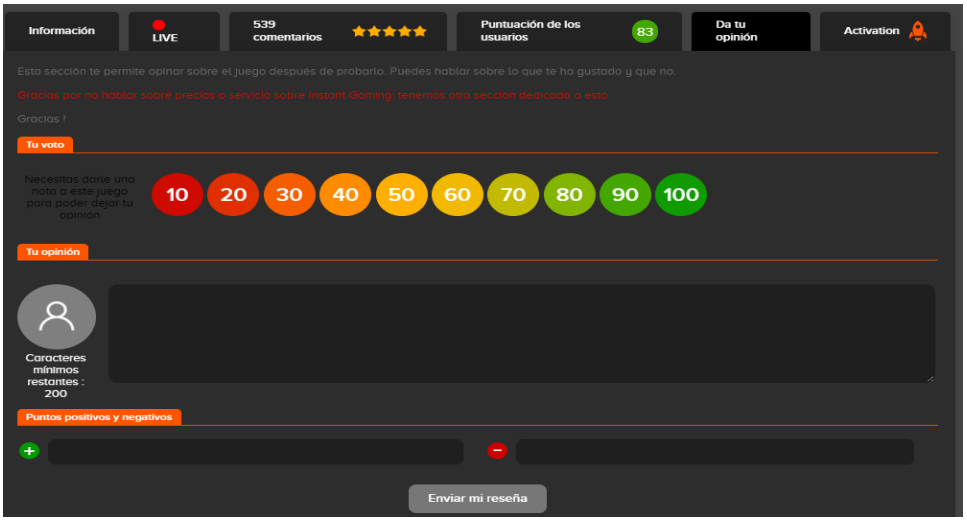

Figura 2.18 - Formulario para hacer una reseña.

Curiosamente, desde la Figura 2.17 sí permite acceder al perfil del usuario que ha creado dicho mensaje (no como en la Figura 2.13). En la Figura 2.19 se muestra el perfil de un usuario, en el cual podemos ver sus logros, juegos comprados, opiniones publicadas, reviews realizadas, solicitar amistad y mucho más.

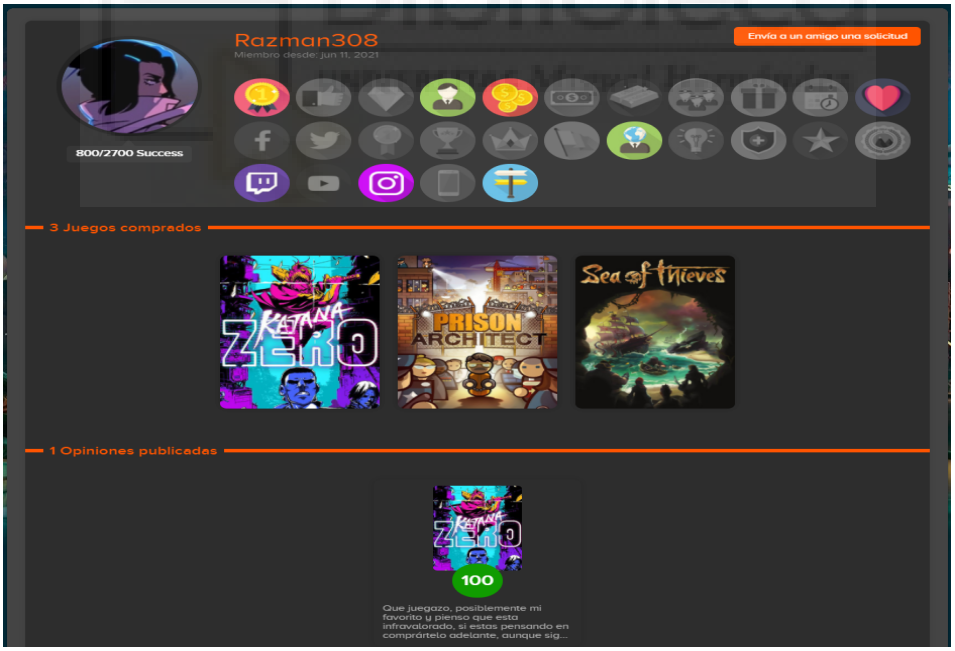

Figura 2.19 - Perfil de un usuario

En la Figura 2.20 tenemos el login de Instant Gaming, como vemos, se centra en que el usuario se logee mediante su cuenta creada o si no tiene, que logee con una de otro servicio; evitan en la medida de lo posible un registro manual, como se muestra en la Figura 2.21.

Figura 2.20 - Login Instant Gaming

Figura 2.21 - Registro manual en Instant Gaming

A modo de curiosidad, como se aprecia en la Figura 2.22 Instant Gaming, dado un correo electrónico, distingue si ese correo está asociado mediante una cuenta de correo electrónico normal o mediante otra plataforma como Facebook o Google.

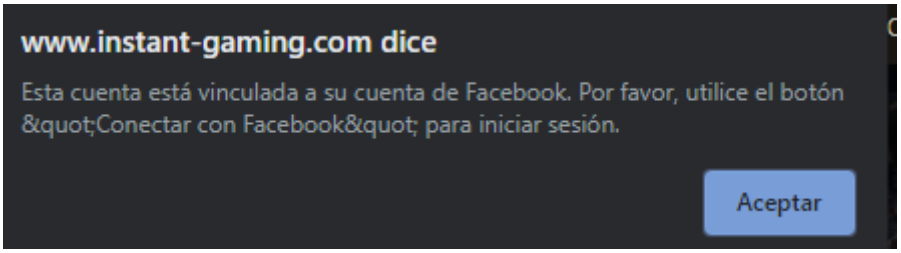

Figura 2.22 - Alert sobre login con otro servicio.

Por último, el juego en detalle (Figura 2.23), vemos que desde aquí podemos elegir la versión del juego (Standard, Stranger Things, Silent Hill) y los DLCS que tenga. La plataforma (Steam o Nintendo Switch), los idiomas, el género del juego, wishlist y lo más interesante, el número de personas viendo ese juego. Eso, instintivamente crea un refuerzo positivo de compra, ya que harías algo que haría la mayoría. Aquí no podemos elegir el comprador, ya que no hay un comprador usuario como tal. Es Instant Gaming quién compra las Keys en cantidades al por mayor para conseguirlas más baratas.

Es interesante que aparte de poder ver la información del producto (requerimientos y de qué trata el juego) o de los comentarios y reseñas, se pueda ver quién de los partners está stremeando dicho juego en ese momento.

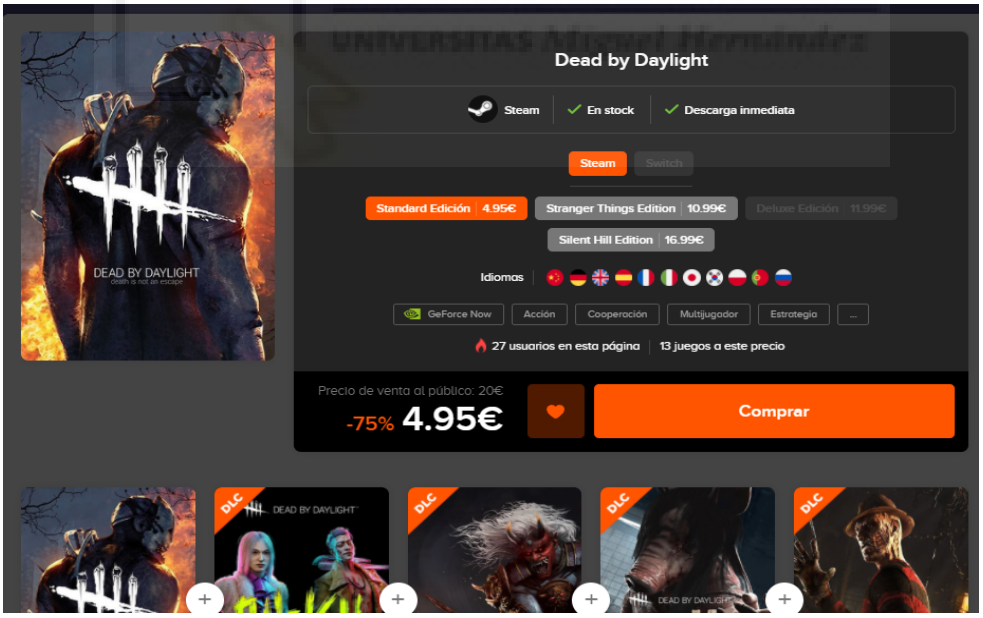

Figura 2.23 - Juego en detalle

Para comprar el juego, debemos ofrecer una dirección de facturación y elegir una forma de pago como lo muestra la Figura 2.24, también cabe mencionar que Instant Gaming no tiene carrito, sólo puedes comprar un producto (Con o Sin DLCs) a la vez y debes estar registrado en la página para recibir el producto y poder verlo desde tu perfil.

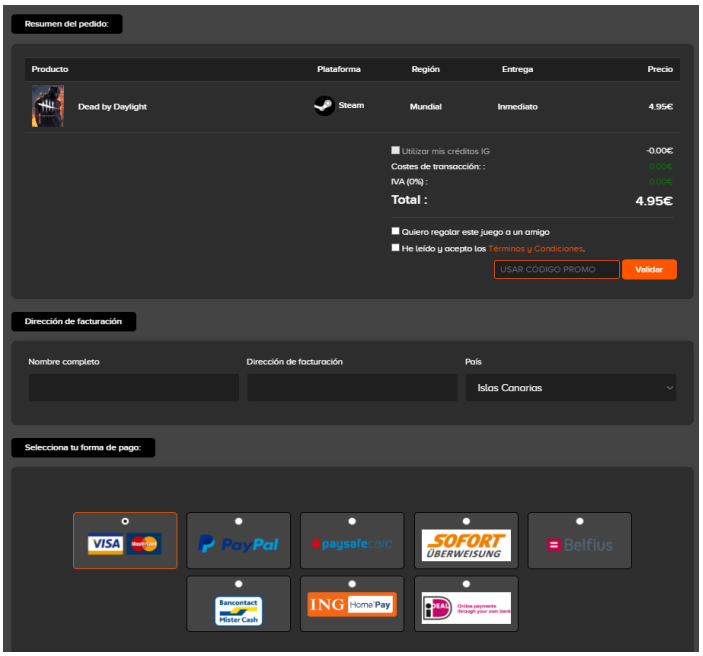

Figura 2.24 - Resumen del pedido

2.2.3.- Kinguin

Kinguin [20] es otra página pionera, creada en 2013, sigue activa a día de hoy. Esta página, vende tanto Keys de juegos como Keys de software. En la Figura 2.25 se muestra su Home.

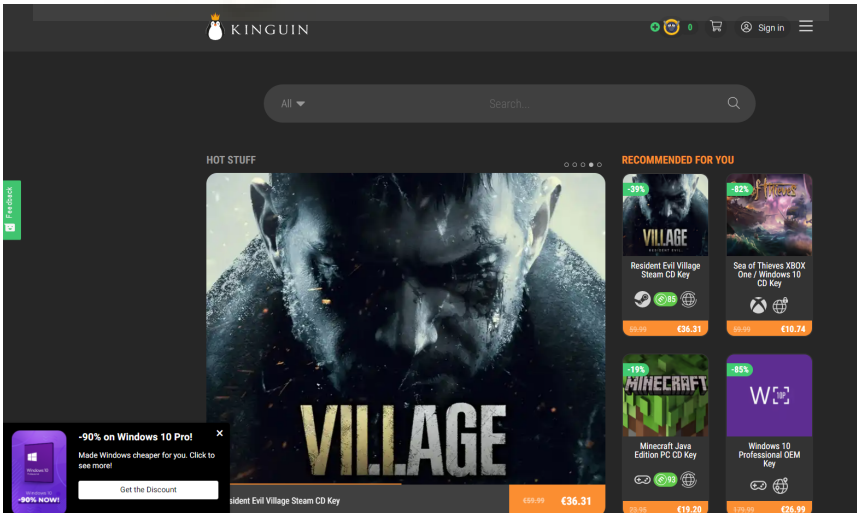

Figura 2.25 - Home Page de Kinguin

La Home Page se divide en 4 secciones, la primera donde se enseñan las Keys de lo más novedoso y recomendado. La segunda muestra los productos más vendidos. La tercera (Figura 2.26) muestra lo nuevo en la página, los juegos a los que se puede hacer pre-order, mostrar los juegos menores a 20 €, los menores a 10€ y un apartado a CSGO, donde

comercia con Gift cards de CSGO. La cuarta muestra los distintos géneros que hay y las distintas plataformas de los productos. En todas las secciones hay una flecha al final de la misma para cargar más contenido en la sección correspondiente.

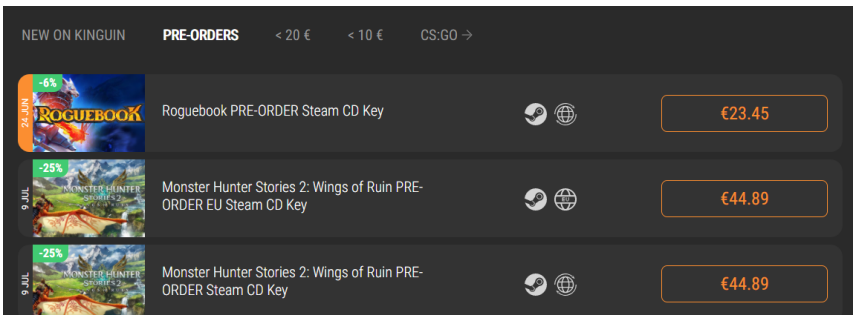

Figura 2.26 - Sección 3 del Home Page

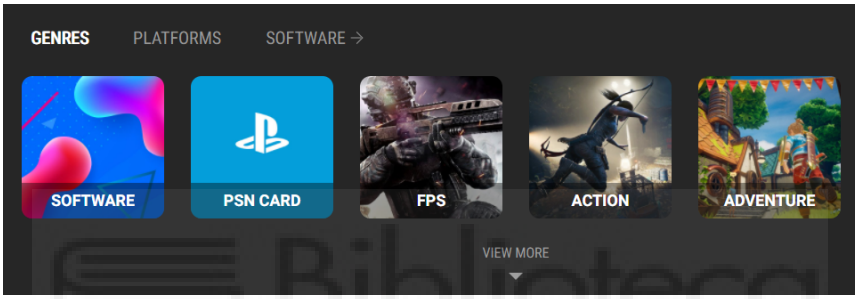

Figura 2.27 - Sección 4 del Home Page

Como se ve en la Figura 2.28 Kinguin cuenta con un Chatbot desde el que se puede contactar con la página para reportar algún problema o incidencia.

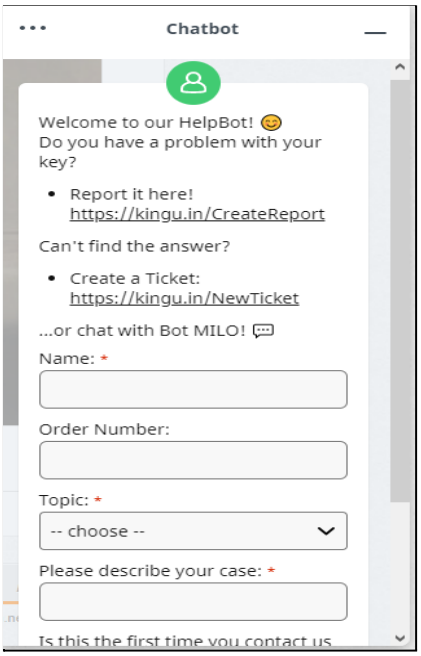

Figura 2.28 - Chatbot

Para comprar en Kinguin, es necesario registrarse primero, la vista es la de la Figura 2.29, algo tradicional.

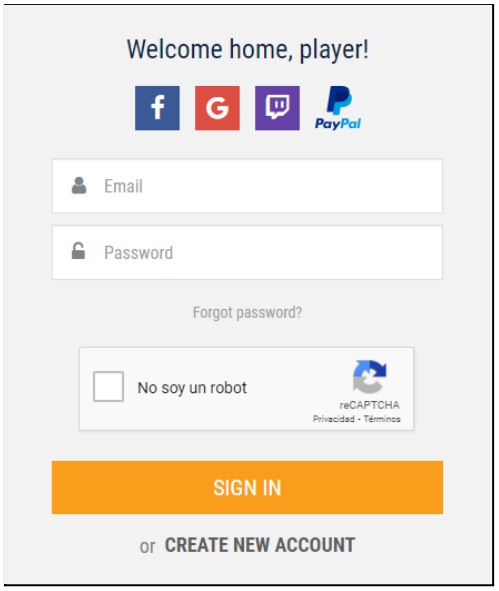

Figura 2.29 - Login

Una vez se ha iniciado sesión ya se puede comprar, es muy similar a G2A, cuenta con un carrito y se puede elegir la Key del vendedor que se quiera (Figura 2.30). No obstante, no es del todo similar. No muestra arriba la Key más barata, esto es debido al sistema de negocios de Kinguin. Mediante este sistema [21] se pueden hacer cosas como mejorar tu rating de negativo a positivo, o de neutral a positivo, mostrar un aviso al usuario para que no se olvide de puntuar, destacar tu Key sobre las demás como se ve aquí, destacar tu producto sobre otros en el Home Page y muchas otras opciones, todo esto tendrá un coste mayor o menor dependiendo de lo invasivo para la página o los beneficios que genere.

Figura 2.30 - Detalle del producto

Por otro lado, cuenta con un sistema de afiliación peculiar [22], proporciona con link único, todo el que compre a través de este enlace hará que te lleves una comisión, dicho esto, si esa persona se registra a través de tu enlace y se afilia también, te llevarás otra comisión de las comisiones que se lleve él. Es una escala piramidal, el de arriba se lleva una comisión de todos los que hay por debajo de él.

Para vender una Key, necesitas crear una cuenta de vendedor, para ello se puede hacer desde [23], se llega a través de las opciones de tu cuenta. Una vez ahí, pulsando en START SELLING NOW se ve en la Figura 2.31

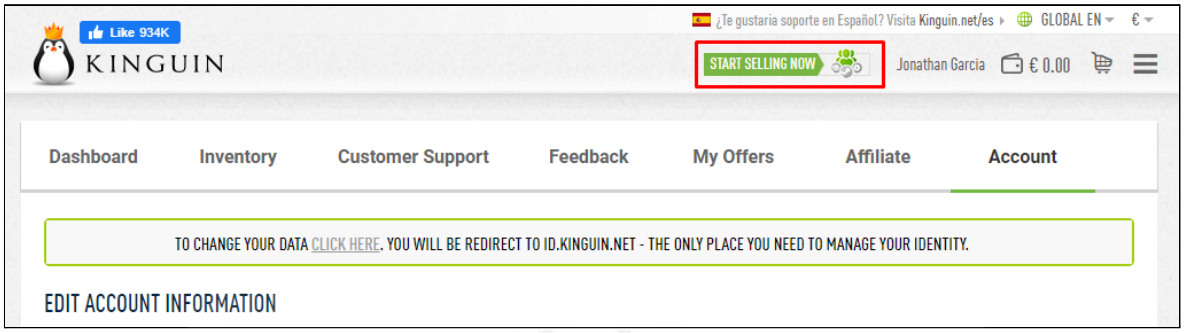

Figura 2.31 - Opciones de Cuenta, Empieza a vender.

2.3.- RESUMEN

Como resumen, vamos a comparar las páginas analizadas, realizando una breve reseña de cada una y establecer qué necesidades cubren, cuáles no y posibles inconvenientes.

G2A. Es una página muy al estilo Amazon, ya que vende casi cualquier producto que esté relacionado con el gaming, como merchandising, mandos de consola, juegos de mesa, software, drones, juegos, etc. Es la página que más ambiente profesional genera y más intenta expandirse y unificar distintos usuarios objetivo bajo un mismo prisma, el gaming.

Lo mejor de G2A, a mi modo de ver, es ese look profesional que tiene gracias a su interfaz, su G2A PAY con ZEN y la variedad de productos. Esto genera una confianza abismal al usuario para poder usarla sin miedo a que sea estafado, por otra parte, su diseño es elegante y simple, es decir, tiene mucho contenido para mostrar, pero la forma y sencillez y lo simple que lo hace para el usuario es remarcable.

Por último, su sistema de loot boxes, un gran acierto que, si bien no lo he visto en la competencia, sí lo he visto en otras páginas más centradas en un juego concreto como CSGO.

Lo peor de G2A, realmente no hay una carencia, no falta nada que sea “necesario”, sólo podría decir que añadiese más funcionalidades extra como las que tiene la competencia para ser aún mejor.

Instant Gaming. Es una página más centrada en el juego en sí y no tan versátil como G2A, sin embargo, me gusta más porque tiene un toque de cercanía que la hace única; como he dicho, no es tan versátil, pero como conjunto ataca mejor el propósito (venta de keys) mejor que ninguna otra, ya que toda la página es una armonía en pos de este objetivo. Esta cercanía se puede ver con un vistazo rápido, sólo tienes que ver los avatares que se ponen en una página u otra; en G2A se suele mantener el avatar default mientras que en Instant Gaming se suele cambiar.

Todo está colocado de tal forma que se siente el propósito de la misma, además, si bien he dicho que G2A es muy elegante, sencilla y simple para el usuario, Instant Gaming lo hace de otra manera, que en mi opinión, al limitarse sólo a las keys, tiene más sentido, me refiero a la diferenciación entre plataformas (PC, Consolas), tiene más sentido como dije en el análisis de Instant Gaming, ese sentimiento de pertenencia a una consola (Sony vs XBOX) y que además tiene más sentido lógicamente esa distribución, pero esto no significa que G2A deba hacerlo así, ya que está más diversificada, lo que digo es que para este caso han usado la mejor opción.

Tiene otros aspectos muy positivos como la sección de comentarios, lo cual genera confianza, pero además se pueden ver todos los comentarios; la sección de partners en Twitch u otras plataformas que además brilla el doble cuando, a la hora de elegir un juego, cabe la posibilidad de que haya alguien stremeando y puedas verlo para ver si te convence.

Por último, la extensión que han creado para que, aunque no estés en la página, sepas que ese juego está (o no) en Instant Gaming y compres el juego, no desde lo has visto, que en este caso sería Steam, sino de Instant Gaming.

Lo peor de Instant Gaming es que el usuario no puede vender sus keys, es la propia página la que compra bundles o paquetes de keys al por mayor y las venden, son ellos los que regulan el mercado y precios.

Por otro lado, no me gusta que en la sección de opiniones de usuarios, en un comentario salga el avatar y nombre del juego, me parece antinatural, deberían aparecer ambas cosas. En la sección de reviews es al revés, sale quién hace el comentario pero no a qué juego se refiere.

Kinguin es sin dudas la peor página de las tres analizadas, en cuanto a feeling y diseño me transmiten muy poca confianza, como si no tuvieran presupuesto para hacer algo de calidad. No obstante, tiene ideas muy buenas como el link de afiliado o el de pagos extra

para tener una mejor posición/reputación/etc en la página y su "*floating icon*" con el chatbot.

Lo peor de Kinguin como he dicho es el feeling y diseño, no me transmite nada de confianza, además usándola da algunos errores y tiene muchas páginas distintas dentro del propio sitio, cambiando el diseño completamente, hasta de la barra de navegación, violando principios de la accesibilidad básicos. La distribución de géneros y plataformas es algo pequeño e importante, pero está colocado al final de la página, por otra parte la forma de mostrarlo es cuestionable.

Capítulo 3
Hipótesis de trabajo

3.1.- HERRAMIENTAS BÁSICAS

En este apartado se hablará del entorno de desarrollo con el que se ha trabajado para el desarrollo del proyecto.

3.1.1.- Visual Studio Code

VSCode ha sido el principal y único editor de código, es uno de los editores más utilizados actualmente, en parte por su diseño, en parte por su flexibilidad. La clave de esta flexibilidad se debe a que es una aplicación escrita con HTML, CSS y Javascript, transformada en app de escritorio con Electron, que es una tecnología que permite construir aplicaciones nativas, tanto para Windows como Mac o Linux entre otros, a partir de los lenguajes mencionados [24] [25].

La flexibilidad de VSCode no es sólo en cuanto a diseño, permite incluir plugins, los cuales pueden ser creados por la comunidad de usuarios o por empresas, de esta manera, se puede trabajar con cualquier tipo de fichero, CSS, C/C++, JSON, Objective-C, Python, R, C#, SQL, Swift, PHP, Ruby, Go, Java, Dockerfile, XML, entre otros muchos. Además, tiene un IntelliSense muy completo, lo cual te permite elegir entre los diferentes métodos y propiedades de un objeto, definiciones de función y módulos importados. También cuenta con una herramienta de Debug para debugear desde el editor, permitiendo el uso de break points, call stacks y una consola interactiva. Por otra parte, mediante los plugins se puede obtener otras herramientas de Debug más personalizadas. [24]

Por último, otro de los puntos fuertes de VSCode es sin lugar a dudas su integración con Git, de tal manera que es posible hacer *git commands* desde diferentes branches a diferentes repositorios incluso sin saber los comandos, ya que cuenta con una interfaz gráfica para facilitar esta tarea (Figura 3.1). [24]

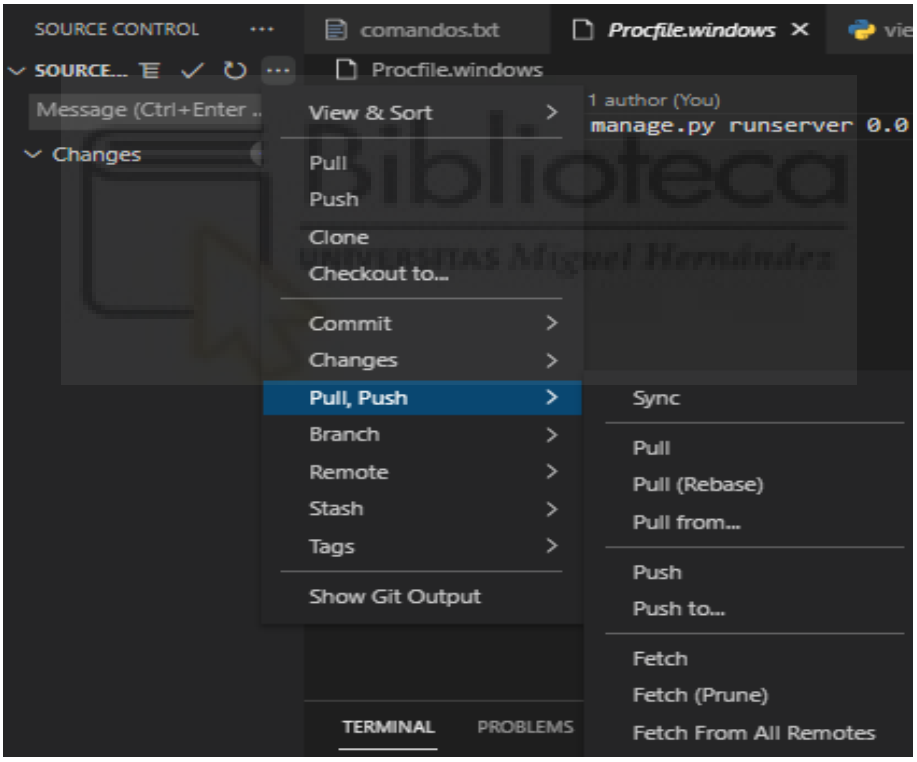

Figura 3.1 - Visual Studio Code, integración Git

3.1.2.- Git y Github

Git es una herramienta de control de versiones [26] que permite mantener un tracking o seguimiento de las diferentes versiones por las que va evolucionando el proyecto, así como trabajo colaborativo mediante sincronización entre los participantes del proyecto. Para ello,

se tiene una copia del proyecto en el servidor al cual los participantes tienen acceso. Cuando dos personas hacen cambios independientes Git junta ambas versiones y los aplica en el proyecto (es recomendable revisar si ha surgido un conflicto al unir).

La mejor característica de Git, es que por cada commit se guarda una copia de la versión, de esta manera, se puede navegar entre las diferentes versiones que ha tenido el proyecto, revertir cambios, buscar errores, etc.

Otra característica muy potente, sobre todo en entornos colaborativos, es el concepto de branches o ramas. Una branch, explicado de forma abstracta es equivalente a un camino. Esto, aplicado a la práctica, podría ser incluir una nueva parte al proyecto, mientras que otra persona trabaja en otra nueva parte del proyecto para luego unir esas dos ramas a la rama principal, de esta manera, la rama principal tendría las partes nuevas de esas 2 ramas en ella.

Un ejemplo en la Figura 3.2, supongamos que trabajamos hasta tener una versión estable, a partir de aquí queremos añadir una nueva función, pues le creamos una rama. Dado el momento descubrimos un bug de la versión estable; pues partimos desde ese estado, solucionamos el error y hacemos un merge entre la Branch master y la otra.

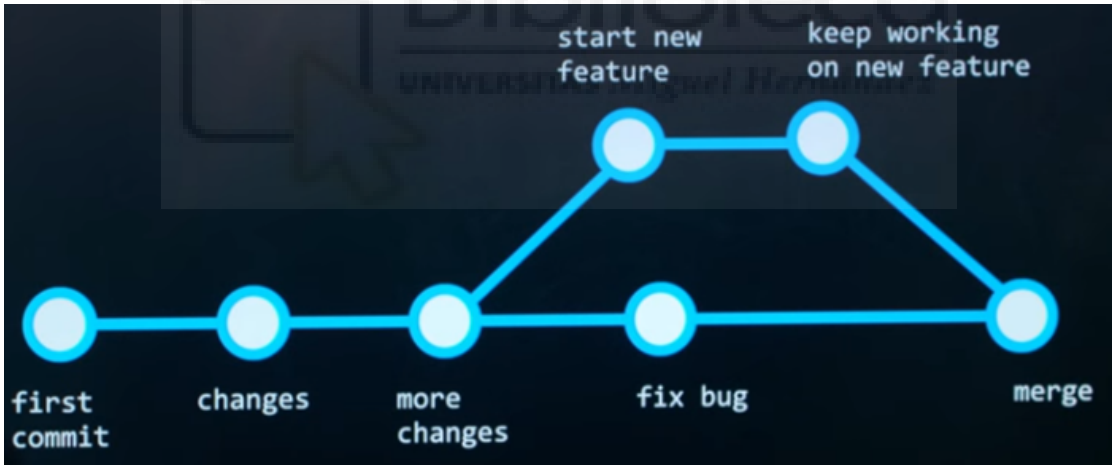

Figura 3.2 - Trabajando con branches o ramas

Para hacer una documentación del repositorio, se escribe en un fichero llamado README.MD, para que Git no busque ciertos directorios o ficheros, se deben añadir en un fichero llamado .gitignore. Por ejemplo, un fichero .env con keys de APIs.

Otra buena característica es su capacidad de colaboración con otras personas, ya sean de la empresa o cualquiera que quiera colaborar, esto se consigue mediante los forks y pull requests, un fork es copia del repositorio en tu pc, conseguido mediante el comando clone. Una vez se disponga del fork, puede ser modificado para luego solicitar un pull request, de esta manera el creador verá los cambios y considerará si incorporarlos o no.

Por último, el concepto de CI/CD permite detectar muchos errores. CI, o Continuous Integration, consiste en frecuentes integraciones y uniones de código modificado entre diferentes contribuidores del proyecto hacia una rama principal con testeos automatizados para verificar esas integraciones. CD, o Continuous Delivery, consiste en hacer actualizaciones frecuentes e incrementales a una aplicación web cuando esas actualizaciones terminen. De esta manera, la master branch se mantiene actualizada y libre de errores (casi todos) gracias a los testeos automatizados. Hacer estos testeos puede variar dependiendo de la herramienta que se utilice, como Travis o CircleCI.

Por último, está la plataforma web Github, la cual almacena proyectos que pueden verse de manera gráfica, tanto los propios como los de otros desarrolladores en el caso de que sean públicos. Hasta es posible instalar algunos de estos proyectos para nuestros trabajos.

3.1.3.- Sistema Operativo

Para el desarrollo del proyecto, no es necesario un SO en concreto, tampoco recomendable. Basta con usar el que te permita trabajar más cómodo, en mi caso Windows 10. De SO a SO no hay prácticamente mucho que cambiar, ya que dispones de la mayoría de herramientas utilizadas u equivalencias, por ende, la diferencia es más en cómo instalar los programas a utilizar; no obstante, si hay que tener en cuenta cómo hacer ciertas cosas a la hora de trabajar con uno u otro.

3.1.4.- Terminal

El terminal utilizado el 90% del tiempo es la CMD clásica de Windows, trabajando en su gran mayoría con comandos de Python, Django, Heroku, Git. El 10% restante se ha utilizado la integración WSL de Windows [27][28], el cual permite tener el shell de bash o zsh; se ha utilizado una extensión de zsh llamado Oh my ZSH (Figura 3.3).

Figura 3.3 - Terminal con Oh my ZSH

3.1.5.- DIA

Dia es una aplicación informática de propósito general para la creación de diagramas desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades. [32]

Está diseñado de tal manera que sea un sustituto perfecto, al ser gratuito, a su equivalente versión comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas como diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. También se pueden agregar nuevas formas dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML.

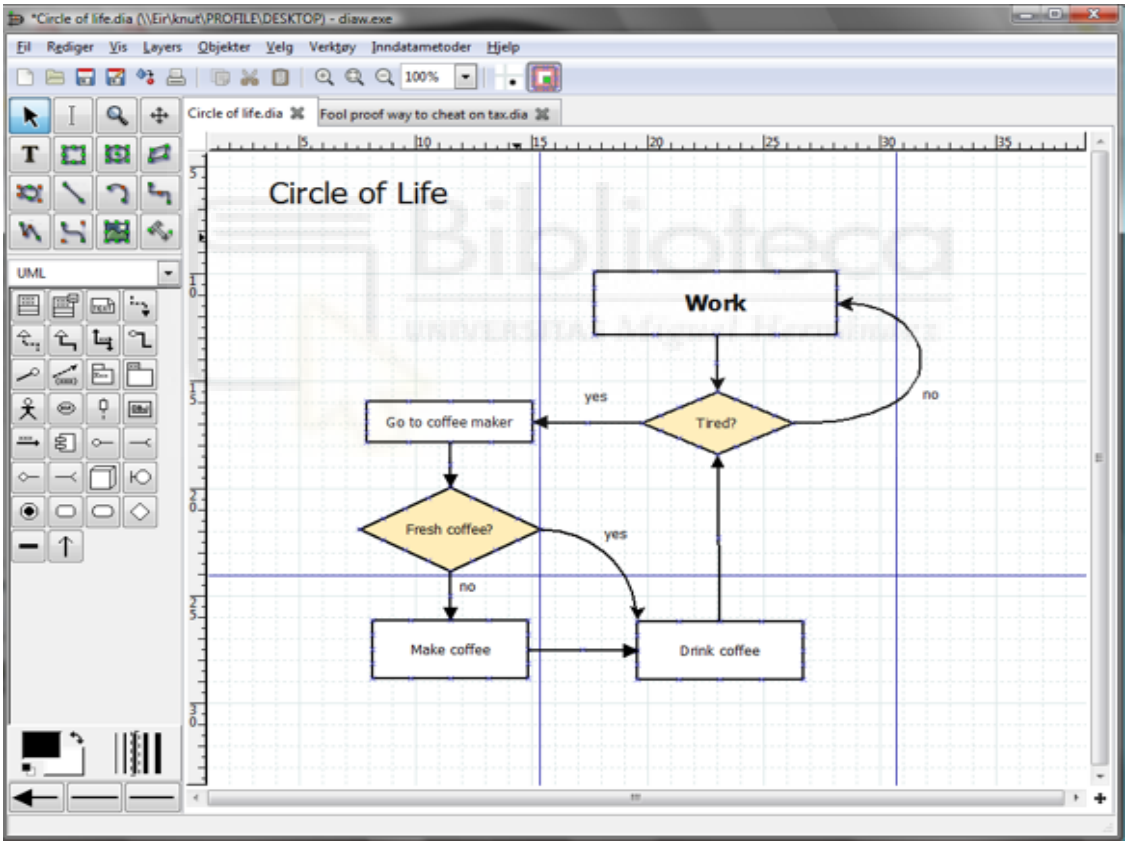

Figura 3.4 - Interfaz de Dia

3.2.- FRONTEND

En este apartado se describirán las tecnologías para el desarrollo de la parte frontend de la aplicación.

3.2.1.- HTML5

Es un lenguaje de marcado, lo cual quiere decir que está compuesto por etiquetas. Es uno de los 3 grandes pilares de las aplicaciones web. HTML5 es la última versión de HTML, que surgió de la necesidad de aunar HTML y XHTML. Especifica el formato de los documentos estándar en WWW, es un lenguaje regulado por el W3C, por tanto, se deben seguir los estándares que marque dicho organismo [33][51].

Esta versión cuenta con una API renovada, agregando Canvas, WebGL, Audio, Video, Web Storage, Websockets, Drag & Drop, Web components y más.

3.2.2.- CSS3

CSS3 es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. El lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. En el presente proyecto, se utiliza para estilizar nuestro HTML5. [34][51]

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles. CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes.

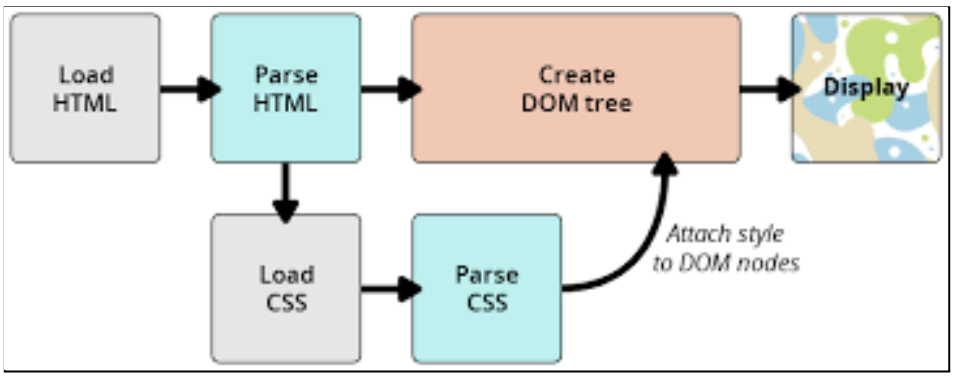

Figura 3.5 - CSS3 Loading

Cuando se solicita documento, el navegador carga el HTML recibido de la red, convierte el HTML en un DOM (Modelo de objetos del documento), busca los recursos vinculados al HTML como las imagenes o videos incrustados y también el CSS vinculado, (luego vendría código Javascript), el navegador analiza el CSS y ordena en diferentes cubos las diferentes reglas según el tipo de selector (de clase, de elemento, de id, etc). Para cada tipo

de selector se calcula qué reglas deben aplicarse y a qué nodos en el DOM se les aplica el estilo según corresponda y se muestra [35].

El encargado de formular las especificaciones de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores es también el W3C.

3.2.3.- JavaScript

Javascript [36] es un lenguaje de programación interpretado, puede ser utilizado tanto en el lado del cliente ejecutándose en el navegador o puede ser ejecutado en el lado del servidor con Node.js. No obstante también puede ser utilizado para apps de escritorio gracias a Electron.js

Lo que ha hecho que Javascript perdure en el tiempo fue su utilidad de reducir la carga de código en el servidor, además de velocidad ya que el código se ejecuta al momento (no tras un HTTP Request). Por otra parte, el lenguaje está constantemente evolucionando, con cada actualización se añaden mejoras muy útiles como lo son la directiva let (similar a declarar una variable en cualquier lenguaje), for of loop (similar a python), las funciones arrow, las promesas, la inclusión de clases y herencia con extends. [51]

3.2.4.- Framework Material Design BootStrap 4

MDB 4 [37] es un framework que se basa en Bootstrap 4 pero añadiendo un estilo 'Material Design' que puso de moda Google. Tiene diferentes variantes para Angular, React, Vue y la versión normal con jQuery.

MDB 4 se basa en BootStrap 4 [42], BS4 es también un framework CSS creado por Twitter para maquetar/estilizar el HTML mediante el uso de librerías CSS. Estas librerías son clases que se pueden incorporar en el atributo class de tu elemento HTML.

Estas librerías se han ido actualizando con el tiempo e incluyen todo tipo de estilos, para botones, menús, tipografías, imágenes, display y fixing y un largo etcétera. Por otra parte, estos estilos son responsive, es decir, se adaptan al tamaño de la pantalla para que siempre se vea algo distinto pero coherente.

MDB 4, al usar BootStrap tiene sus mismas clases y algunas más, también mantiene el sistema de añadir el estilo dentro del atributo class. Sin embargo, no sólo permite añadir CSS, sino también muestra ejemplos o "Componentes", proporcionando el código HTML, CSS y JS si lo hubiera.

En la Figura 3.6 se tiene un ejemplo de cómo estiliza un botón con las clases para que se vea como un botón con MD y un dropdown.

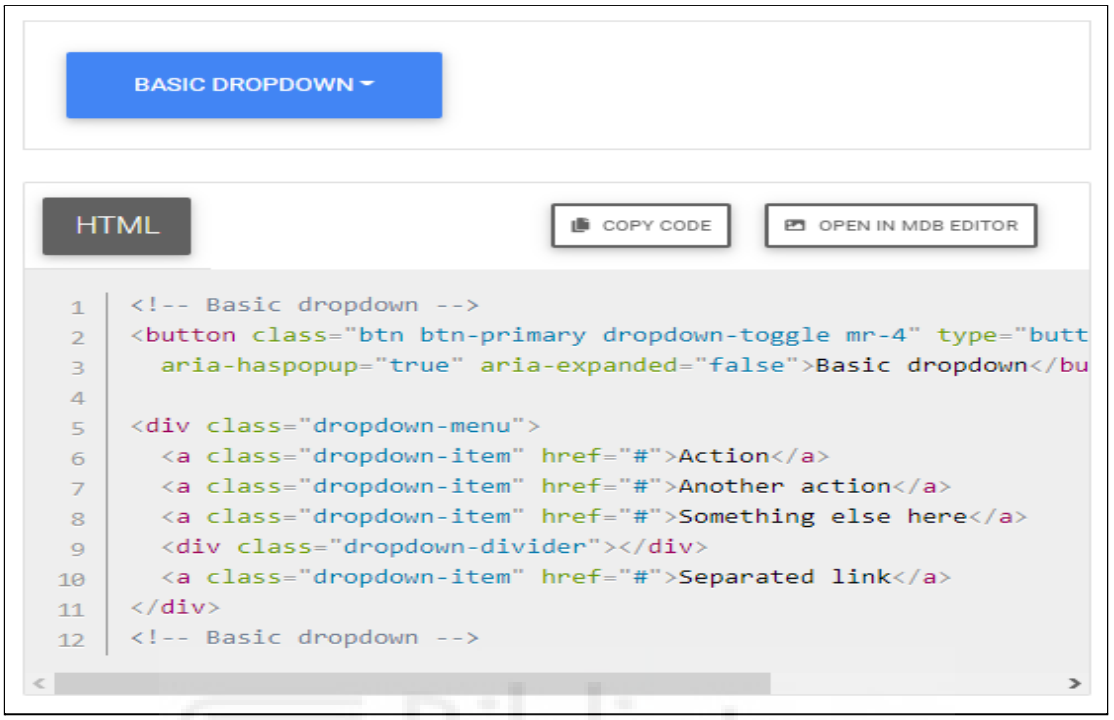

Figura 3.6 - Ejemplo de uso MDB 4

3.3.- BACKEND

En el backend se tienen las tecnologías usadas en el lado del servidor para dar lógica y dinamicidad a la aplicación web.

3.3.1.- Python

Python es un lenguaje de programación de tipado dinámico y fuerte, es decir, sabe distinguir el tipo de variable en base a su contenido, pero no permite concatenar tipos de datos distintos. Es un lenguaje interpretado, cuenta con su propio intérprete el cual debe ser instalado, éste lee el código Python.py línea a línea y las va ejecutando.

Es un lenguaje muy versátil ya que se puede utilizar tanto en aplicaciones de escritorio como servidor como web, funcionando en todos los Sistemas Operativos (siempre y cuando se instale el intérprete). Es un lenguaje orientado a objetos con herencia múltiple, donde todo son referencias. Está pensado para ser intuitivo a la hora de leer, es por esto que usa operadores como “in”, “is”, “not in” “is not”, etc. [51]

3.3.2.- Framework Django

Django [49] es un framework open source basado en Python para desarrollo web en Python. Cuenta con una documentación muy pulida [50] y resulta muy fácil encontrar recursos y ayuda para su aprendizaje. Un proyecto en Django es una carpeta con el nombre del propio proyecto, que adicionalmente estará compuesto de apps (que se podrían entender como módulos del proyecto) ubicadas en sus propias carpetas [51] (en el anexo I se amplían los detalles sobre este framework).

3.3.2.1 - Apps de terceros

Como muestra la Figura 3.7, para poder usar AWS S3 se ha instalado el paquete/app '*storages*' [78], para el login con Google se ha instalado la app '*allauth*' [79], para estilizar los forms la app '*crispy_forms*' [80], para disponer de un listado de con todos los países (para usarlo en un select desplegable en la aplicación) la app '*django_countries*' [81], y para redimensionar imágenes desde el template la app '*daguerre*' [82].

```
#AWS S3
'storages',

#Third-party apps
'crispy_forms',
'widget_tweaks',
'django_countries',
'daguerre',

#Third party app Allauth
'django.contrib.sites',
'allauth',
'allauth.account',
'allauth.socialaccount',
'allauth.socialaccount.providers.google'
```

Figura 3.7 - Apps de terceros

3.3.3.- Base de Datos

En este proyecto se han usado dos bases de datos, SQLite para trabajar en local, que trae Django por defecto, y PostGreSQL para producción. En ninguno de los dos casos ha sido

necesario usar el lenguaje SQL gracias al ORM (Object-relational mapping) de Django. SQLite [43], al contrario de otros gestores de base de datos, no es una base de datos cliente-servidor. Se podría describir como una base de datos embebida dentro del programa-servidor.

PostGreSQL [39] es un gestor de bases de datos orientadas a objetos, cumple los estándares SQL92 y SQL99, y soporta funcionalidades avanzadas. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Es de código libre, publicado bajo la licencia BSD. Permite el uso de datos espaciales, es decir, admite datos geográficos los cuales se pueden ver en un mapa. Dispone de diversos plugins, entre ellos PostGIS, que permite trabajar con dichos datos espaciales. PostGreSQL cuenta con una webapp que permite manejar la BD con interfaz gráfica, llamada PgAdmin4 [40] (Figura 3.8).

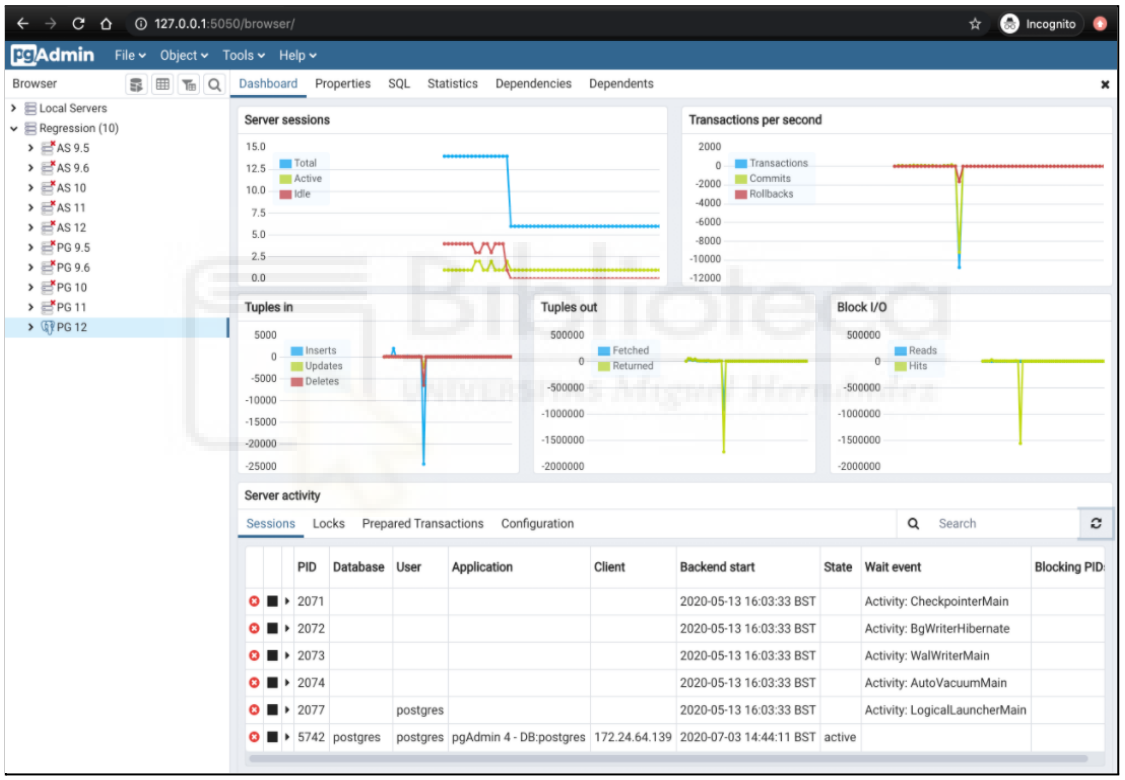

Figura 3.8 - Interfaz PgAdmin4

3.3.4.- DBeaver

Dbeaver [41] es una herramienta de Base de Datos gratuita multiplataforma para desarrolladores, administradores de Base de datos, analistas y todo tipo de usuarios que quieran trabajar con Bases de datos. Soporta todo tipo de bases de datos populares, como MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc.

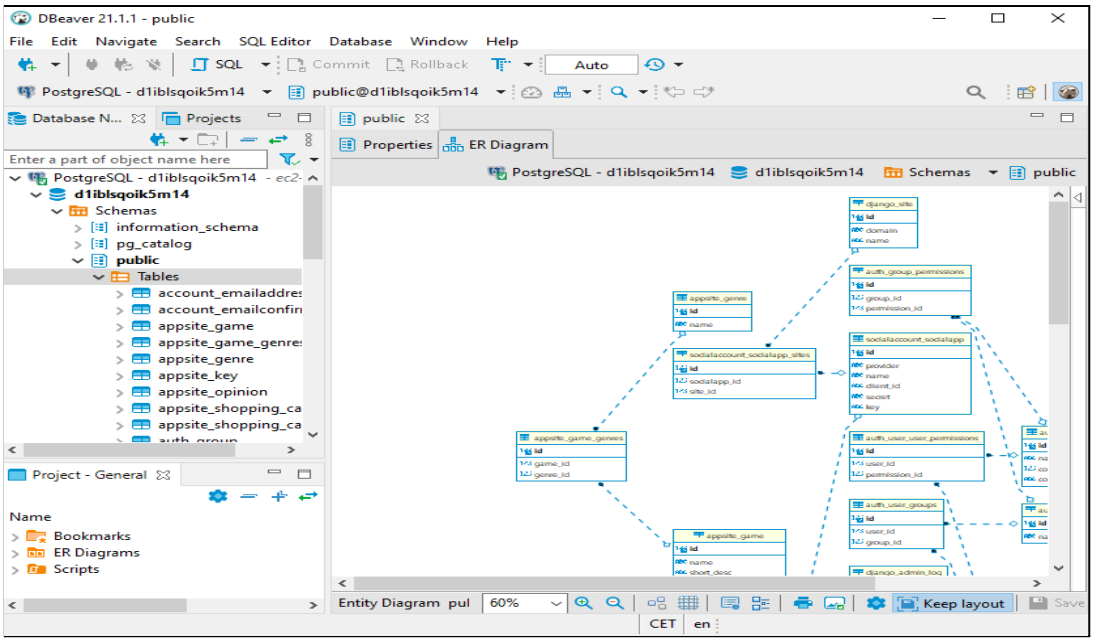

Figura 3.9 - Interfaz Dbeaver

3.4.- DESPLIEGUE EN PRODUCCIÓN

Para hacer el despliegue de la aplicación en se ha utilizado la plataforma Heroku y el CDN AWS S3 (Content Delivery Network de Amazon Web Services) para las imágenes, además de ciertas modificaciones en el proyecto desde el framework Django. Se han elegido estas herramientas porque combinan bien entre ellas, es decir, su implementación resulta fácil, por lo que son muy usadas para este tipo de despliegues. Ambas herramientas, junto con Django y PostgreSQL, se pueden ver como un '*stack*', análogo a WAMP/LAMP (Windows/Linux + Apache + MySQL + PHP), que conforman un conjunto de tecnologías que, usadas en conjunto, constituyen una base sobre la que desarrollar aplicaciones.

3.4.1.- Heroku

Heroku [44] es una de las primeras plataformas cloud [45] que viene desarrollándose desde 2007, proporciona un servicio de tipo PaaS (*Platform as a Service*), y soporta diversos lenguajes de programación como Ruby, Java, Node.js, Scala, Clojure, Python, PHP y GO.

El cloud hosting es un tipo de host en el que se accede a un clúster, y éste, en vez de estar vinculado a un servidor principal, utiliza un conjunto de servidores virtuales que pueden o no estar en el mismo entorno, y se van asignando dinámicamente según la necesidad de uso. Esta característica proporciona una serie de beneficios, como escalabilidad y mayor potencia de procesamiento [45].

Para que Heroku funcione hay que conocer sus tres aspectos principales:

- Los “dynos”.- Es el nombre con el que se denomina a los clusters en Heroku. Por defecto, una app gratuita utiliza un dyno gratuito. Un dyno es como una contenedor ligero que ejecuta los comandos ejecutados en un fichero llamado Procfile. Un dyno gratuito dejará de ejecutarse tras media hora de inactividad, o lo que es lo mismo, tras media hora de no recibir tráfico.

- Las variables de entorno.- Son variables almacenadas en zonas no visibles del programa o repositorio, de tal manera que puedan utilizarse pero sin que se pueda saber su valor. Esto es útil, por ejemplo, para almacenar contraseñas de APIs de terceros. Heroku buscará estas variables en el fichero .env si trabajamos con Heroku en local, o en config:var si está desplegado.

- Los ficheros clave.- Definen formas de operar, por ejemplo, detectan si una app está en Python, o PHP, que procesos se deben ejecutar, de qué tipo son, etc. Los dos ficheros más importantes son:

    - El fichero requirements.txt.- donde estarán las dependencias del programa, para que Heroku las instale en los dynos.

    - El fichero Procfile.- ejecuta los procesos correspondientes, en este caso un proceso de tipo web para ejecutar el servidor con gunicorn.

En el anexo II se amplía la información sobre esta plataforma.

3.4.2.- CDN - Amazon Web Services S3

Amazon Web Services [52] cuenta con muchos servicios que se ejecutan a través del cloud computing para que una empresa, organización o alguien independiente pueda tener sus necesidades cubiertas. En concreto se ha utilizado el servicio S3, el cual permite disponer de una base de datos con las imágenes que serán renderizadas en la aplicación web.

Primero que todo se tiene que crear un bucket o base de datos donde guardar las imágenes, el nombre de éste será único en todo el mundo, hay que elegir la región del mismo y una serie de configuraciones. Cuando se termine la configuración se tendrá algo similar a la Figura 3.10.

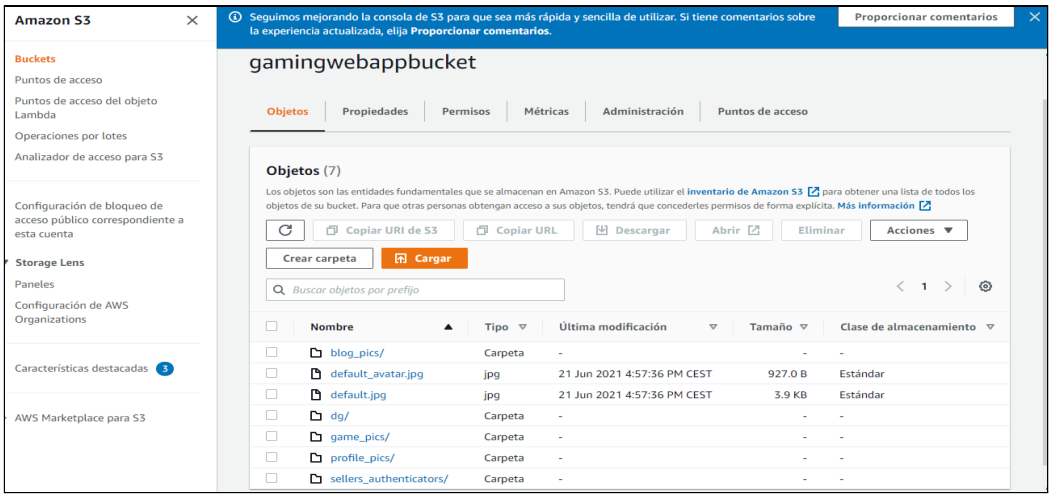

Figura 3.10 - BD con las imágenes en AWS S3

Capítulo 4
Metodología y resultados

Todo proyecto de desarrollo de software está basado en algún ciclo de vida del software, en este capítulo se va a describir el desarrollo de este proyecto, analizando el ciclo de vida utilizado, un diagrama de gantt con los tiempos de cada etapa y otros detalles acerca de los requisitos, diseño y desarrollo de la aplicación.

4.1.- PLANIFICACIÓN DEL PROYECTO

4.1.1.- Ciclo de vida

Un ciclo de vida en cascada puede ser implementado de muchas maneras distintas, aquí se ha usado la que muestra la Figura 4.1, a continuación se procederá a explicar de forma resumida en qué consiste cada etapa [83][84][85].

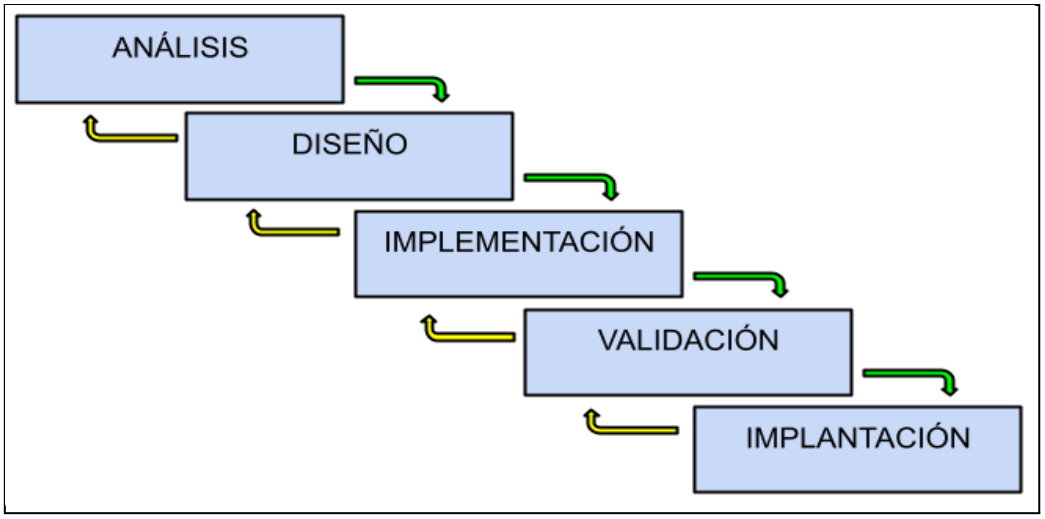

Figura 4.1 - Ciclo de vida en cascada.

- Análisis: En esta etapa se extraen los requerimientos del software; se busca encontrar requerimientos incompatibles y/o ambiguos. Se describe el comportamiento que se espera que tenga el software una vez sea desarrollado, identificando las necesidades del negocio y la interacción. Por ello, esto se verá reflejado en los casos de uso. Esta etapa es muy importante porque una buena etapa de análisis puede ahorrar mucho tiempo y trabajo en las futuras etapas.

- Diseño: Describe de manera general la construcción de una aplicación mediante documentación, diagramas de clases, base de datos, secuencia, entre otros.

- Implementación: Consiste en traducir los casos de uso, requerimientos y otros diagramas en código, es decir, en la aplicación en sí, incluyendo la implementación de la Base de datos también.

- Validación: Se prosigue a la validación e implementación del mismo mediante testeos, para buscar errores, ya sea revisando el código a ser posible por una persona distinta al que lo realizó, implementar testeos desde el propio Framework Django e incluir validaciones CI/CD mediante herramientas como Travis o Selenium y se evalúa que la documentación entregada sea de calidad.

- Implantación: Se recoge toda la documentación general en el resto de etapas del proceso: especificaciones, diseño, manuales e informe de errores con sus soluciones y se realiza el correspondiente despliegue de la aplicación.

  Al ser un PMV, el feedback es muy importante, éste será tomado en cuenta para realizar una retroalimentación y volver a etapas anteriores de este ciclo de vida para corregir y/o actualizar los errores o posibles sugerencias.

4.1.2.- Diagrama de Gantt

El proyecto se ha desarrollado en el segundo cuatrimestre del año 2020 y del año 2021, en el 2020 se hizo el primer PMV y una breve documentación del mismo. En 2021 se ha realizado otro PMV esta vez con la documentación correspondiente para que sea un TFG.

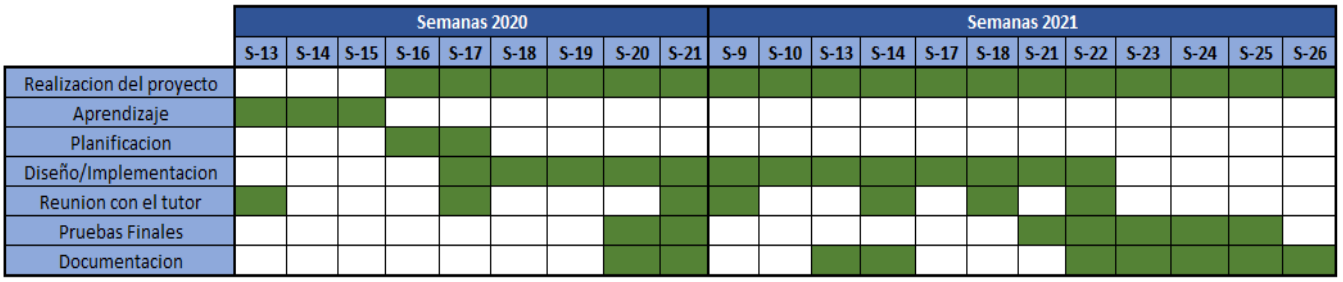

Figura 4.2 - Diagrama de Gantt

En la etapa de aprendizaje todavía no se había consumado la idea general del proyecto, sólo se buscaba aprender tecnologías nuevas, en este período se investigó en su mayoría la bibliografía [51] entre muchas otras. A partir de aquí, se deseó poner en práctica lo aprendido junto con el objetivo de crear un negocio, esto se gestó en la etapa de planificación. A continuación vino la etapa más importante y larga, el diseño e implementación del presente proyecto.

4.2.- CAPTURA DE REQUISITOS

4.2.1.- Actores

Tabla 4.1.- Descripción del rol "Invitado"

Actor	**Invitado**
Descripción	Actor no logeado en la página, por tanto no tiene permisos para hacer la mayoría de casos de uso.

Tabla 4.2.- Descripción del rol "Usuario registrado"

Actor	**Usuario registrado**
Descripción	Este actor está registrado en la BD de la aplicación y ha iniciado sesión. Puede hacer la mayoría de casos de uso.

Tabla 4.3.- Descripción del rol “Vendedor”

Actor	**Vendedor**
Descripción	Es un usuario que se crea a partir de un usuario registrado, debe ser validado por un administrador para que pueda ejercer sus funciones.

Tabla 4.4.- Descripción del rol “Administrador”

Actor	**Administrador**
Descripción	Es un usuario el cual un superusuario le ha dado permisos para ser una cuenta de tipo admin. Además, necesitará que le se asignen más permisos para poder trabajar desde la URI /admin y no solo poder logearse en la misma.

Tabla 4.5.- Descripción del rol “Superusuario”

Actor	**Superusuario**
Descripción	Es un usuario el cual tiene todos los permisos y por tanto puede hacer todos los casos de uso, menos el de loguearse ya que ya está logueado.

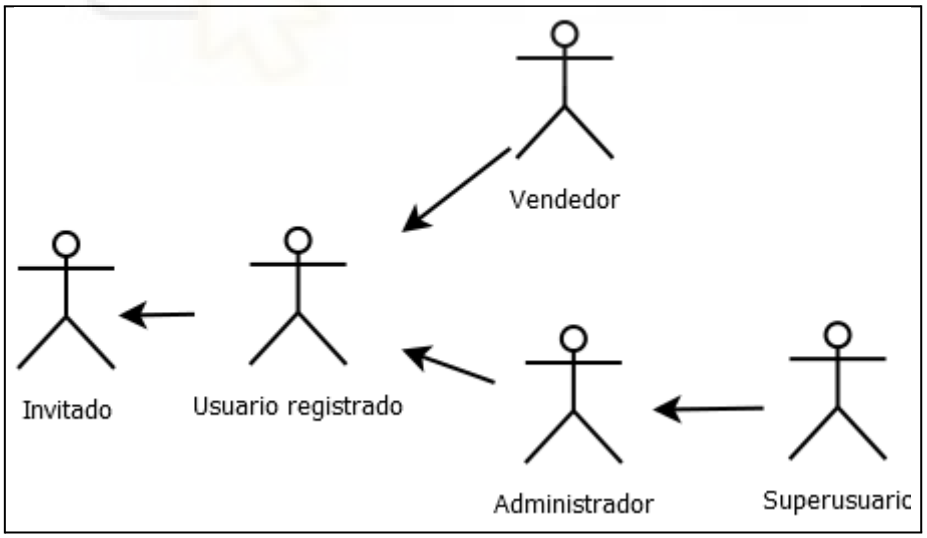

Figura 4.3 - Diagrama UML con la herencia de los actores

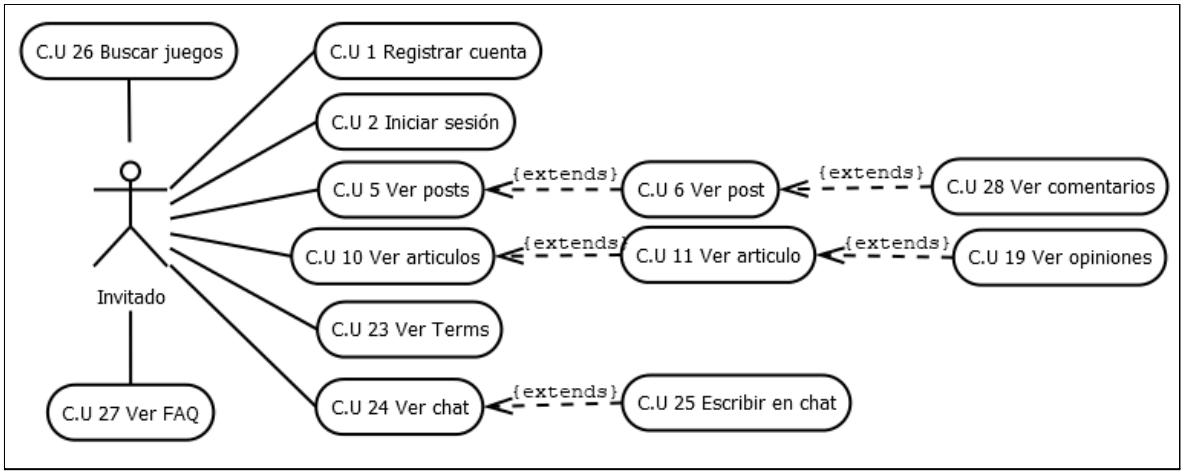

Figura 4.4 - Diagrama UML con los casos de uso de Invitado.

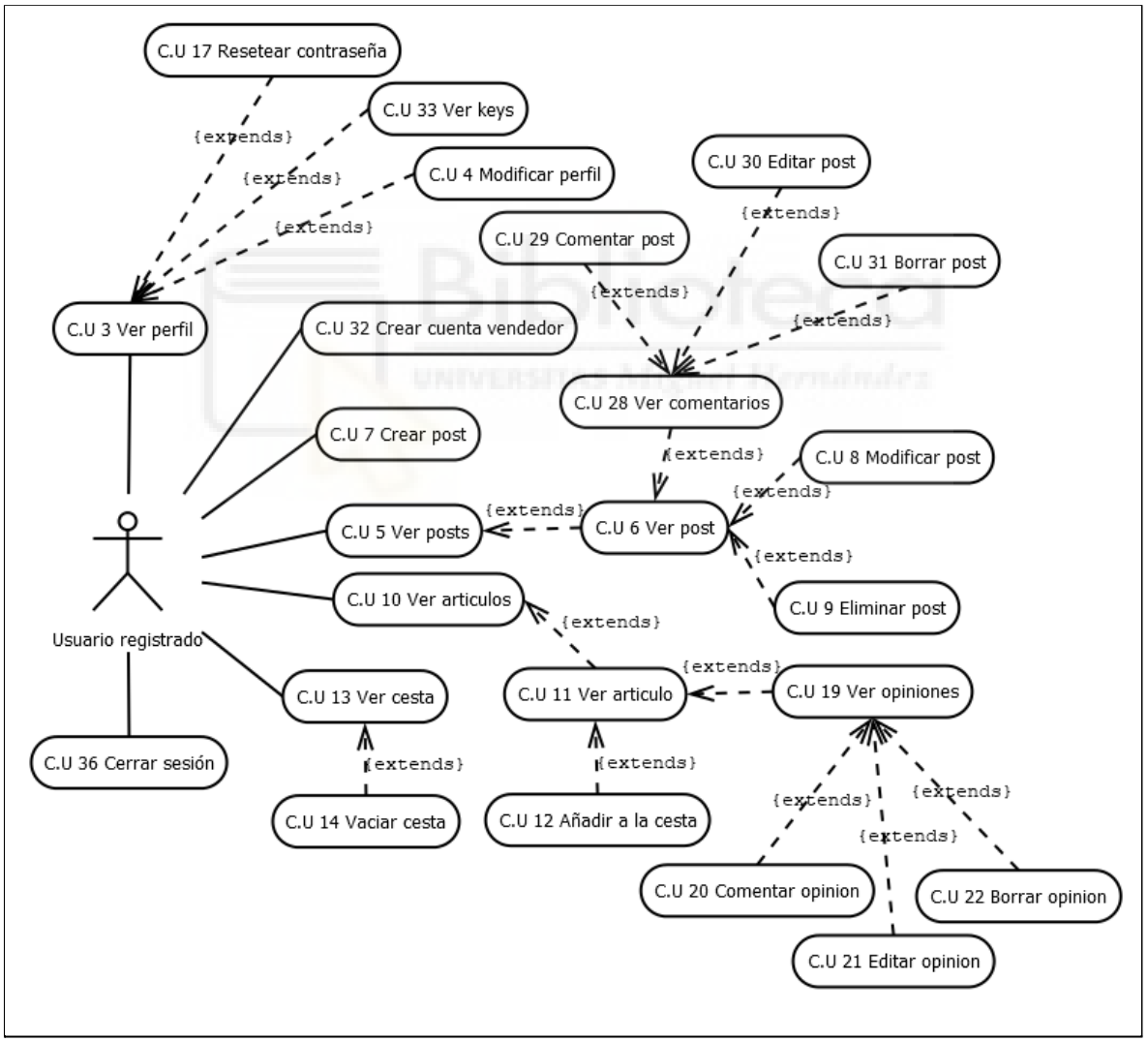

Figura 4.5 - Diagrama UML con los casos de uso de Usuario registrado.

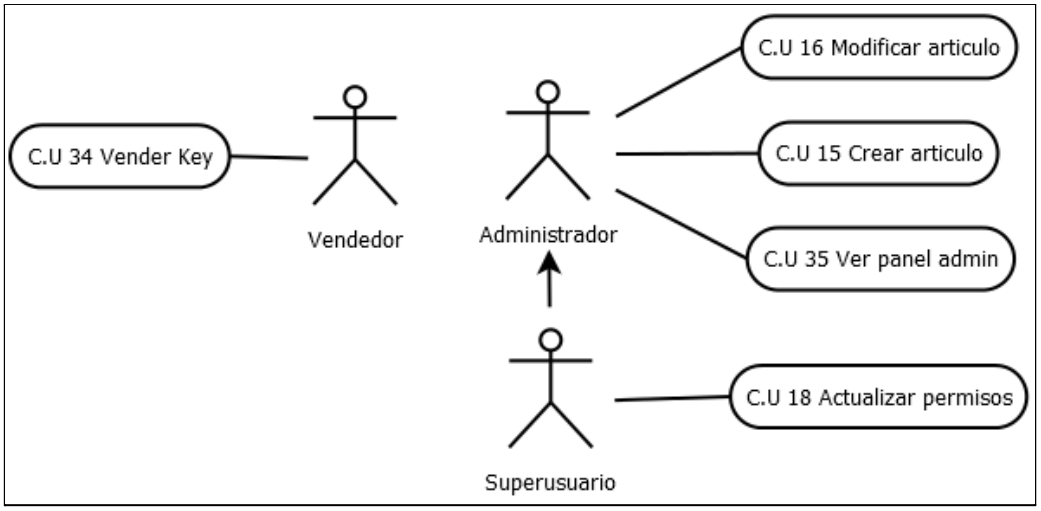

Figura 4.6 - Diagrama UML con los casos de uso de Vendedor, Administrador y Superusuario.

4.2.2.- Casos de uso

La descripción de cada caso de uso se ha realizado usando plantillas como la que se muestra en la tabla 4.6. A modo de ejemplo, en dicha tabla, se muestra el caso de uso 1 de la aplicación. El resto de casos de uso se encuentran en el anexo III.

Tabla 4.6.- Casos de uso 1: Registrar cuenta

C.U. 1	*Registrar cuenta*
Actores	Invitado
Descripción	Un invitado o guest en la página puede registrarse en la aplicación rellenando un formulario o con una cuenta en Gmail.
Dependencias	
Precondición	
Secuencia normal	P1 - El usuario introduce sus credenciales en el formulario o pulsa en el icono G, para iniciar sesión con la cuenta de Google. P2 - Pulsar el botón “Sign Up”.
Poscondición	Si el formulario es válido, se creará una cuenta, el actor invitado pasará a ser un usuario registrado.
Excepciones	● Si el formulario no es válido, se indica con un mensaje y se vuelve al formulario de registro.
Comentarios	Un administrador o superusuario también puede registrar una cuenta desde el panel de admin.

4.3.- DISEÑO DE LA BASE DE DATOS

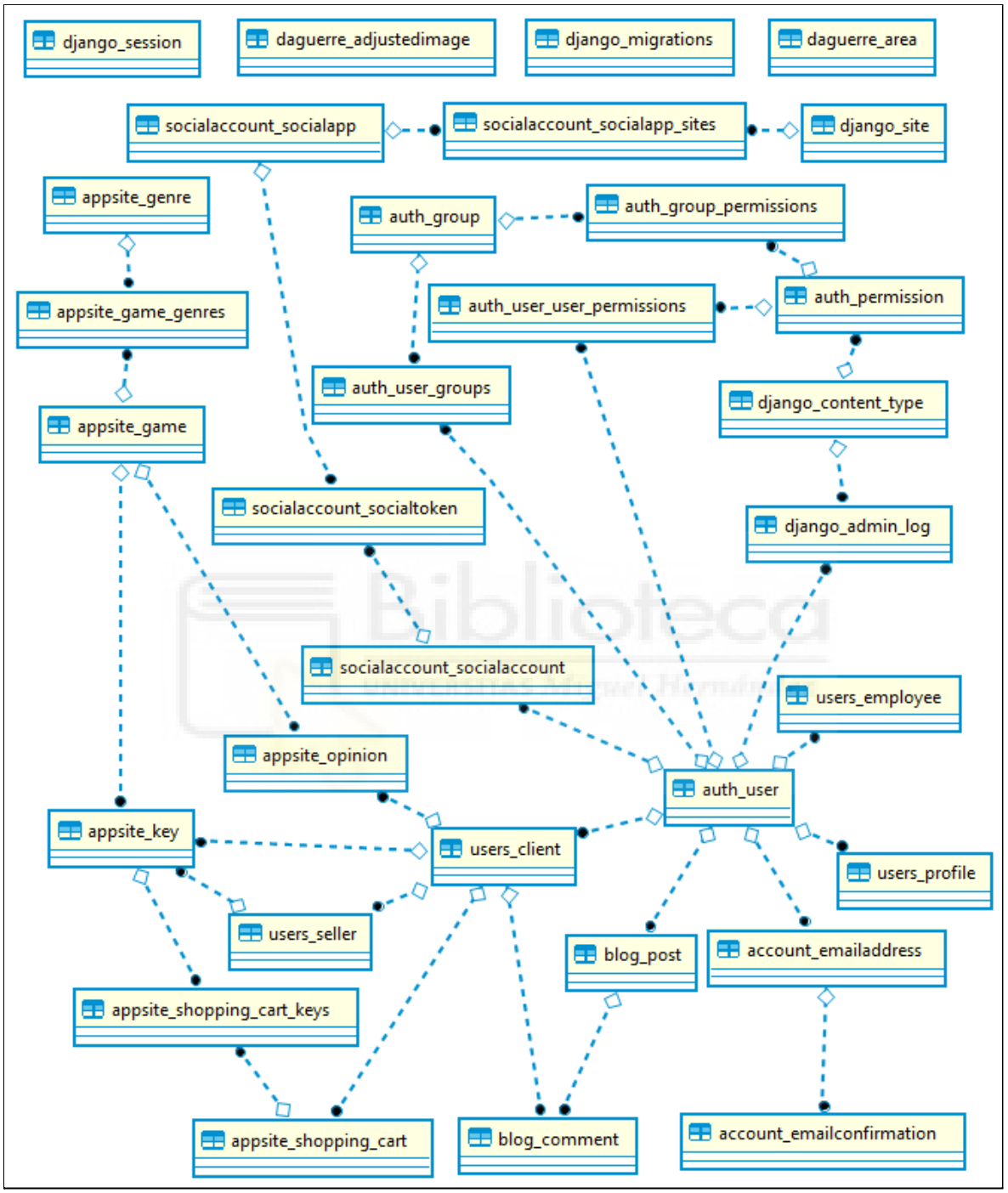

Figura 4.7 - Diagrama ER de la base de datos

La figura 4.7 muestra todas las tablas de la base de datos de la aplicación junto con sus relaciones, dichas relaciones son todas de cardinalidad ‘1-m’ (uno a muchos), donde el punto negro representa a la tabla con cardinalidad ‘m’ (muchos), y el rombo con fondo blanco representa la cardinalidad ‘1’ (uno). Como se puede observar, los nombres de las

tablas empiezan con un prefijo determinado que viene especificado desde el ORM de Django para indicar a qué aplicación pertenece cada tabla, de esta manera se evita la colisión de nombres, estos prefijos tienen el siguiente significado:

- appsite_: Son las tablas que pertenecen a la aplicación 'appsite' en Django, se utilizan para gestionar toda la información referente a los juegos como el género o tipo de juego, las opiniones, las claves y el carrito de compra.

- auth_: Se trata de las tablas que genera Django por defecto para la gestión de la autenticación y validación, contienen información de identificación del usuario, así como los grupos y los permisos tanto de los usuarios como de los grupos.

- blog_: Estas tablas contienen la información de blogs y los comentarios de los hilos de cada uno de ellos.

- daguerre_: Son las tablas pertenecientes a la aplicación 'daguerre', la cual permite la gestión de las imágenes, tal sea como el renderizado de la misma en la página, como la forma o presentación de la misma (16:9, 4:3).

- django_: También son tablas que genera el framework Django de manera automática. Sirven para la gestión del framework para que Django pueda ejercer su función de controlador.

- socialaccount_: Pertenecen a la aplicación 'socialaccount', sirven para la gestión del login desde aplicaciones de terceros, como el login mediante Google.

- users_: Son las tablas que pertenecen a la aplicación 'users', se encargan de manejar la información del usuario, como el perfil y el tipo de cuenta (empleado, cliente, vendedor).

En el anexo IV se incluye todo el detalle del diseño físico de cada una de las tablas de la base de datos.

4.4.- IMPLEMENTACIÓN

En la Figura 4.8 se tienen 2 funciones, una que devuelve una lista con los juegos y la otra función es una vista, en esta vista cargaremos el resultado de lo obtenido en la primera. La función distingue entre géneros y texto que contiene el juego, es decir, si ponemos un género listará todos los juegos que tengan ese género, si ponemos una palabra suelta listará todos los juegos que contengan esa palabra suelta; cuanto más estricto sea el nombre más

específica la búsqueda, si se introduce el texto “dy” listará todos los juegos que contienen dy, pero si se introduce “dying light” listará todos los juegos que contienen dying light, obteniendo una búsqueda más precisa.

```python
#Function to return a list with the games found in the query
def get_game_queryset(query=None):
    queryset = []
    queries = query.split(" ")
    genres = Genre.objects.all()
    aux=False
    genre = None

    for q in queries:
        for auxgenre in genres:
            if q.upper()==auxgenre.name.upper():
                aux=True
                genre = Genre.objects.filter(name=q.capitalize()).first()

    for q in queries:
        if aux==True:
            games = Game.objects.filter(stock__gte=1).filter(genres=genre).distinct()
            for game in games:
                queryset.append(game)
        else:
            games = Game.objects.filter(stock__gte=1).filter(Q(name__icontains=q)).distinct()
            for game in games:
                queryset.append(game)

    return list(set(queryset))

#View to show the games matching the query
def search_game_queryset(request):
    context = {}
    query = ""
    if request.GET:
        query = request.GET['q']
        context['query'] = str(query)

    try: #Searching for name
        games = sorted(get_game_queryset(query), key=attrgetter('name'), reverse=True)
        context['games'] = games
    except: #Searching for genre
        games = sorted(get_game_queryset(query), reverse=True)
        context['games'] = games

    return render(request, "appsite/search.html", context)
```

Figura 4.8 - Código para el buscador de juegos en el navbar

```python
class GameDetailView(FormMixin, DetailView):
    model = Game
    form_class = OpinionForm

    def get_context_data(self, **kwargs):
        context = super(GameDetailView, self).get_context_data(**kwargs)
        context['lower_key'] = Key.objects.filter(game_id=self.object.id).filter(used=False).order_by('price').first()
        context['keys'] = Key.objects.filter(game_id=self.object.id).filter(used=False).order_by('price')
        #context['opinions'] = Opinion.objects.filter(game_id=self.object.id)

        if self.request.user.is_authenticated:
            opinion = Opinion.objects.filter(client=self.request.user.client).filter(game_id=self.object.id)
            if opinion.exists():
                context['opinion_bool'] = True
            else:
                context['opinion_bool'] = False
        else:
            context['opinion_bool'] = True

        context['form'] = OpinionForm() #initial={'game': self.object, 'client': self.request.user.client}
        return context

    def post(self, request, *args, **kwargs):
        if not request.user.is_authenticated:
            return HttpResponseForbidden()
        self.object = self.get_object()
        form = self.get_form()
        form.instance.client = self.request.user.client
        form.instance.game = self.object
        if form.is_valid():
            return self.form_valid(form)
        else:
            return self.form_invalid(form)

    def form_valid(self, form):
        try:
            form.save()
        except IntegrityError:
            messages.error(self.request, f'Your have already commented on this game')
        except:
            return HttpResponseForbidden()

        return super(GameDetailView, self).form_valid(form)

    def get_success_url(self):
        return reverse('appsite-game_detail', kwargs={'pk': self.object.pk})
```


Figura 4.9 - Código para mostrar un juego

En la Figura 4.9 se tiene la vista de un juego en concreto, heredando de la clase DetailView; no obstante, como dentro de la vista del juego tenemos un form para dejar un comentario, tenemos que conseguir que la misma ruta sirva para mostrar el juego y para completar el formulario. Esto se consigue con los mixins, en concreto, FormMixin [86]. De esta manera, con la función post, form_valid y get_success_url conseguimos combinar estas 2 necesidades en una vista.

Para el chat, en vez de crear uno de cero, se ha optado por elegir un chat ya creado, por esto se ha elegido Xat. Xat permite crear un chat con moderadores, administradores, control de logs y de permisos, así como el fondo que se use, entre muchas otras opciones.

Para utilizarlo se ha creado un xat, y se ha vinculado incrustando el iframe correspondiente.

El algoritmo de la figura 4.10, en lenguaje template Django, verifica primero si la página tiene paginación, luego verifica si la página en la que estamos tiene página previa (por tanto será false si estamos en la página 1, o si sólo hay 1 y por ende estaríamos en la página 1). Luego con el for añadimos 2 páginas anterior a la que estamos, en la que estamos y las 2 próximas. Por último comprobamos si hay página siguiente, si la hay añade los botones Next y Previous.

```
{% if is_paginated %}

    {% if page_obj.has_previous %}
        <a class="btn btn-outline-info btn-sm mb4" href="?page=1">First</a>
        <a class="btn btn-outline-info btn-sm mb4" href="?page={{ page_obj.previous_page_number }}">Previous</a>
    {% endif %}

    {% for num in page_obj.paginator.page_range %}
        {% if page_obj.number == num %}
            <a class="btn btn-info btn-sm mb4" href="?page={{ num }}">{{ num }}</a>
        {% elif num > page_obj.number|add:'-3' and num < page_obj.number|add:'3' %}
            <a class="btn btn-outline-info btn-sm mb4" href="?page={{ num }}">{{ num }}</a>
        {% endif %}
    {% endfor %}

    {% if page_obj.has_next %}
        <a class="btn btn-outline-info btn-sm mb4" href="?page={{ page_obj.next_page_number }}">Next</a>
        <a class="btn btn-outline-info btn-sm mb4" href="?page={{ page_obj.paginator.num_pages }}">Previous</a>
    {% endif %}

{% endif %}
```

Figura 4.10 - Código para la paginación de objetos

4.5.- DESPLIEGUE

El primer paso será ejecutar el comando '*pip install django-heroku*', esta librería se encarga de hacer una serie de configuraciones automatizadas para evitar una carga excesiva de configuración de parámetros.

Como se ha explicado en el capítulo 3, Heroku necesita una serie de ficheros para funcionar, el contenido del mismo variará dependiendo de las tecnologías que se vayan a usar. Como se está utilizando Django, se necesitarán las siguientes configuraciones:

- Fichero requirements.txt: Dependencias del proyecto, para saber las dependencias se ejecuta el comando pip freeze.
- Fichero Procfile: web: gunicorn gamingwebapp.wsgi --log-file -
- Fichero .env: Las variables de entorno (Para trabajar en local)
- Fichero .gitignore: Añadimos la excepción .env
- Fichero runtime.txt: python-3.8.2

Una vez se tienen los ficheros, debemos crear una aplicación en Heroku, para ello es necesario logearse en el terminal con el comando '*heroku login*', seguido del comando '*git init*' en caso de no haberse utilizado previamente. A partir de aquí ya se dispone de un repositorio en git, a continuación se debe crear la aplicación en Heroku, ya sea mediante la interfaz gráfica o con el comando '*heroku create*' o '*heroku create nombredelaapp*'. La primera opción escogerá un nombre aleatorio. '*heroku create*' crea un repositorio remoto y lo añade automáticamente a nuestro git con el comando '*remote add heroku link.git*', para comprobarlo podemos escribir git remote, nos listará los repositorios con los que está vinculado el mismo.

A continuación, será necesario modificar el fichero 'settings.py' para permitir la integración de Heroku con Django:

- import django_heroku
- STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
- ALLOWED_HOSTS = ['gamingwebapp.herokuapp.com']
- django_heroku.settings(locals())

No obstante, en [58] no se dice que Django no creará el directorio especificado en STATIC_ROOT que usa collectstatic, debemos crear el directorio manualmente, también tener en cuenta que Git no permite directorios vacíos así que se tiene que poner un fichero cualquiera también y asegurar de no añadir el directorio en .gitignore.
Se continuará agregando las variables de entorno necesarias para el correcto funcionamiento de la aplicación en Heroku, mediante el comando 'heroku config:set nombre="valor"', en la Figura 4.11 se puede observar las que utiliza este proyecto.

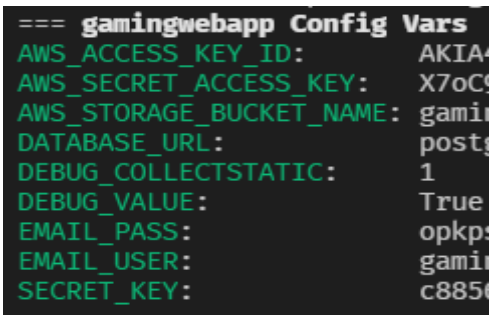

Figura 4.11 - Variables de entorno en Heroku.

Dada la situación del proyecto, se ha optado por un despliegue mediante git push, por tanto será necesario ejecutar los siguientes comandos:

- git add .
- git commit -m "..."
- git push heroku master

Con git add y luego commit disponemos los ficheros para que sean enviados con git push, este git push será al repositorio remoto, por ende git push heroku master. El uso de este despliegue en concreto tiene una serie de pros y contras:

- PRO: Soporta submódulos de git.
- PRO: Manera sencilla de añadir en cualquier workflow basado en Git
- CONTRA: Requiere acceso tanto al repositorio Git como a la app en Heroku.

Al hacer la instalación en el orden propuesto, no será necesario hacer operaciones como crear una base de datos en Heroku con el comando '*heroku addons:create heroku-postgresql:hobby-dev*' y su consecuente linkado con el comando 'heroku config:set nombre="valor"', ya que como se ha expuesto anteriormente, en el capítulo 3, al detectar ciertas librerías Heroku instala el addon automáticamente. Con el comando '*heroku pg*' se puede comprobar efectivamente la BD que se está utilizando.

Como se ha cambiado la BD, no se dispone de ningún registro en ella, será necesario volver a crear un super usuario, para ello se ejecutará el comando '*heroku run python manage.py migrate*' y '*heroku run python manage.py createsuperuser*'.

Con esto la aplicación está alojada exitosamente en Heroku, pero si se quiere conocer más sobre el despliegue como cambiar el Debug u otro parámetro, ver [31]

Por último, se tiene que configurar AWS S3 para poder ver las imágenes. Se necesita disponer de un bucket o "BD" en AWS S3 donde guardar las imágenes, el nombre de éste será único en todo el mundo, hay que elegir la región del mismo y una serie de configuraciones.

Una vez creado el bucket, debemos especificar las variables de entorno y otras variables como en la Figura 4.12, e importar la app 'storages' en 'INSTALLED_APPS'

```
AWS_ACCESS_KEY_ID=os.environ.get('AWS_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY=os.environ.get('AWS_SECRET_ACCESS_KEY')
AWS_STORAGE_BUCKET_NAME=os.environ.get('AWS_STORAGE_BUCKET_NAME')
AWS_S3_FILE_OVERWRITE = False #If 2 users upload 2 files equal named wont overwrite
AWS_DEFAULT_ACL = None

#boto3
AWS_S3_REGION_NAME = 'eu-west-3' #change to your region
AWS_S3_SIGNATURE_VERSION = 's3v4'

"""
#boto
AWS_S3_HOST = 'us-east-2' #change to your region
S3_USE_SIGV4 = True
"""

DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
```

Figura 4.12 - Código para ejecutar S3 con Django-Heroku

Las 3 primeras variables obtienen su valor del fichero .env, para poder trabajar en local. Es necesario añadir estas variables en Heroku como se muestra en la Figura 4.8, con el consecuente script:

- git add .
- git commit -m "..."
- git push heroku master

Capítulo 5
Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

Comentar las principales conclusiones derivadas de todo el proyecto desarrollado. En especial, dejar constancia de que los objetivos que se habían planteado al principio han sido alcanzados.

La principal conclusión que he aprendido es que crear una aplicación web, que sea funcional, útil y que cumpla con la ley para que se le pueda sacar beneficio económico es más difícil de lo que pensaba. También influye qué tipo de aplicación sea, pero en general, convertirlo en un negocio es bastante engorroso porque hay que recoger los datos personales de los que quieran vender, el aspecto legal para poder comercializar la aplicación es algo que tendré que investigar a fondo.

Otra conclusión es que, si tuviera que repetir este proceso para crear una aplicación web aunque sea distinta, iría mucho más rápido al no tener que perder tanto tiempo investigando cómo hacer las cosas, ya que Django tiene una curva de aprendizaje muy pronunciada.

Pero para mí la conclusión más importante es todo lo que he aprendido ajeno al proyecto, el proyecto se basa en tecnologías que no conocía, por tanto, casi todo lo que he investigado ha sido un soplo de aire fresco para mí, pero por otra parte también me ha ayudado a conocer más de lenguajes/tecnologías y de entender, o mejor dicho, tener unas bases más sólidas de ideas que ya tenía antes. He sentido como si me hubiera emergido en la cultura de otro país con respecto al desarrollo web. Antes lo tenía como piezas inconexas, pero ahora veo mejor el puzzle completo.

También tengo más cultura informática, conceptos como "programador fullstack" o palabras como "frontend" y "backend" que por raro que parezca no conocía ya que se me había explicado el concepto y en español, no con la jerga inglesa. Otros conceptos como la diferencia entre página web y sitio web, aplicación web y páginas estáticas y dinámicas; cómo podemos usar "página web" para referirse vulgarmente a un sitio web/aplicación web.

Diferentes maneras de crear una "página web", yo sabía que se podía con un CMS (Wordpress, Drupal, Magento, Ghost, Keystone, …) o sin CMS, pero vi que también hay otras opciones como un híbrido de CMS y gestión sin CMS que repito, no lo conozco pero al menos sé de su existencia; los static site generators que generan una app en HTML, CSS, JS partiendo de cierto lenguaje de programación y una estructura ya creada, algunos ejemplos: Gatsby, Jekyll, HuGo, Pelican.

Diferentes opciones para hacer una aplicación web como Multi Page Application MPA donde la aplicación web se compondría de varias rutas y varios documentos HTML, los cuales se van solicitando. Single Page Application SPA donde la aplicación web con varias rutas pero un sólo HTML que se va actualizando. Aplicaciones renderizadas del lado del servidor donde la mitad es MPA mitad SPA (Frameworks: Angular Universal, Next.js, Nuxt.js). Las SPA son aplicaciones desarrolladas con Angular, React o Vue.js y son más rápidas pero no se le puede aplicar SEO, al menos de forma sencilla.

Nuevas herramientas en el frontend, a parte de haber mejorado mis conocimientos de HTML5, CSS3, JS (Sobretodo ES6 y ES8, AJAX con callbacks, promesas, async funcs, que no he usado en este proyecto pero lo he aprendido gracias a él). Nuevos frameworks para CSS a partir del conocido BootStrap (el cual intenté utilizar una vez para un una práctica como punto opcional y como no me salía desistí), como lo son Materialize, MDB4, Bulma, Semantic, Foundation. De todos estos el que más me gustó fue Material Design BootStrap 4, que como su nombre indica se basa en BS4 pero aplicando las reglas

de estilo que ha puesto de moda Google desde su integración en el diseño Android. El problema que tuve con BootStrap 4 es que no entendí el concepto de que el atributo class de HTML pudiera ser modificado.

Por otra parte, he descubierto la existencia de preprocesadores de CSS, como Sass, Less, Stylus. En concreto he visto Sass, el cual permite crear variables, anidación y herencia; luego transforma el fichero .scss a css por consola de comandos, no me ha hecho falta utilizarlo en la práctica pero al menos he aprendido lo básico.

Frameworks de Javascript como React, Angular, Vue.js, Marko.js, Polymer, Preact, Aurelia, Ember.js. De estas sólo he investigado un poco los 3 primeros y cómo cada uno es mejor para una cosa en concreto y que React está haciéndose muy popular; sus state management como NgRx para Angular, Redux, Hooks, Context Api para React y VueX para Vue js, que no sé cómo trabajan pero al menos ahora ya sé que existen.

En la parte de Base de Datos, nunca antes había tenido una aplicación que enlazara con una base de datos remota, como tampoco sabía de la existencia del SGBD que recomendaban utilizar para el deploy, PostGreSQL.

Básicamente no he tenido que tratar con la BD a bajo nivel por línea de comandos, pero si he aprendido los diferentes tipos de datos y he probado comandos así como también me he informado de diferencias con Oracle que es con la que más he trabajado por consola.

También he visto algo de Firebase, una base de datos como servicio. Hace de API para que el cliente se pueda comunicar con la BD sin tener que pasar por el código del servidor. Al no tener server-site, la almacena el propio Google ofreciéndonos ellos la interfaz gráfica. Es bastante útil para guardar datos desde el frontend con Javascript.

Para el hosting habían varias opciones como un shared hosting donde se alquila un servidor a compartir con otros usuarios, por ejemplo Namecheap, Bluehost o Hostgator. También podría haber hecho yo mismo de servidor, pero la opción más tentadora era usar un cloud. Aunque repito, cualquier opción hubiera sido válida menos el hosting de webs estáticas como ofrecen Netlify o Github Pages.

En esta categoría tenemos opciones como Heroku, Amazon Web Services, Digital Ocean, Google Cloud, Microsoft Azure. Me he decantado por Heroku por su fácil implementación con Django y porque me permite añadir ciertos servicios, como una base de datos de manera gratuita pero escalable.

Para los frameworks de aplicaciones web vi Flask antes que Django en un curso que impartió la universidad de Harvard en la pandemia, ahí mejoré mis conocimientos de HTML5, CSS3, JS, Flask, Git, Django, Heroku entre otras tecnologías.

5.2.- POSIBLES DESARROLLOS FUTUROS

El principal trabajo futuro a realizar sería el testeo en los ficheros tests.py para poder aplicarle CI/CD con Travis.

A partir de aquí, trabajar sobre la aplicación, siendo el aspecto más importante el de enlazar una pasarela de pago con la que poder hacer transacciones. También investigar los smart contracts que ofrece la red de criptomonedas Etherium, para evitar estafas con los vendedores fraudulentos.

Añadir el número de keys vendidas en los vendedores para transmitir mayor seguridad a los clientes.

Poder variar la cantidad de keys que quieres de un juego, seleccionando automáticamente la siguiente key más barata.

Que la aplicación mande un correo cuando se cree una cuenta de vendedor, y añadir previamente a este muros de validación para reducir el spam, como por ejemplo, que la imagen del DNI sea una imagen de DNI y que sea visible.

Introducir un sistema de rating con JS para mejorar la confianza que inspiren los vendedores.

Permitir compartir los posts por las principales redes sociales.

Mejorar la sección blog para convertirla en una especie de foro.

Mejorar la personalización del perfil, dándole un toque más "*gamer*", con recompensas y logros, por ejemplo, comprador habitual cuando compres 30 keys.

Poder ver los perfiles de otros usuarios.

Permitir que los vendedores puedan comprar un servicio que les permita promocionar sus keys aun no siendo las más baratas.

Mejorar el formulario de comentario/opinión, editar y borrar.

Bibliografía

[1] Internet Domain Survey
https://www.isc.org/survey/
03/2021

[2] Hogares con acceso a Internet en España
https://www.ine.es/prensa/tich_2017.pdf
03/2021

[3] Internet de las cosas: cuando todo está conectado
https://www.lavanguardia.com/vida/junior-report/20190301/46752655177/internet-cosas-dispositivos-conectados-iot.html#foto-2
03/2021

[4] Influencia del COVID-19 en el crecimiento de internet
https://thediplomatinspain.com/2020/09/como-influyo-el-coronavirus-covid-19-al-crecimiento-del-internet/
03/2021

[5] Juegos físicos vs digitales en PC y consola
https://hardzone.es/tutoriales/compras/pc-consola-juegos-digitales-fisicos/
03/2021

[6] El 'streaming' llega a la guerra comercial por los videojuegos
https://www.laverdad.es/tecnologia/internet/streaming-llega-guerra-videojuegos-20200304121216-ntrc.html?ref=https:%2F%2Fwww.laverdad.es%2Ftecnologia%2Finternet%2Fstreaming-llega-guerra-videojuegos-20200304121216-ntrc.html
03/2021

[7] Si todos jugamos por streaming no habrá quien venda videojuegos en persona
https://www.xataka.com/empresas-y-economia/todos-jugamos-streaming-no-habra-quien-venda-videojuegos-persona-nuevo-desafio-para-game-mediamarkt-resto-tiendas-fisicas
03/2021

[8] El efecto de la dopamina en ratas
https://www.elespanol.com/ciencia/investigacion/20181018/ratas-prefirieron-placer-comida-vida/346215884_0.html
05/2021

[9] Supernormal Stimuli: How Primal Urges Overran Their Evolutionary Purpose
Deirdre Barrett
Publicado el 22 de febrero de 2010. Revisado el 05/2021

[10] Caja de Skinner
https://psicologiaymente.com/psicologia/caja-de-skinner
05/2021

[11] Trastorno por videojuegos
https://www.lavanguardia.com/cribeo/geek/20190530/47436058513/la-oms-anade-la-adiccion-a-los-videojuegos-a-su-lista-de-enfermedades.html#:~:text=La%20OMS%20dice%20que%20el,y%20actividades%20diarias%2C%20y%20la
05/2021

[12] G2A.COM
https://www.g2a.com
06/2021

[13] PCCOMPONENTES
https://www.pccomponentes.com
06/2021

[14] AMAZON
https://www.amazon.com
06/2021

[15] Producto Mínimo Viable
https://robertotouza.com/lean-startup/que-es-el-mvp-o-producto-minimo-viable-en-las-startups/
06/2021

[16] The Lean Startup
Eric Ries
Publicado en 2011. Revisado el 06/2021

[17] Qué es el Producto Mínimo Viable (MVP)
https://www.emprenderalia.com/que-es-el-mvp-producto-viable-minimo/
06/2021

[18] Mínimo Producto Viable: ¿Qué es y Para qué?
https://sg.com.mx/revista/31/minimo-producto-viable-que-es-y-para-que
06/2021

[19] Instant Gaming
https://www.instant-gaming.com/es/
06/2021

[20] Kinguin
https://www.instant-gaming.com/es/
06/2021

[21] Kinguin Business
https://business.kinguin.net
06/2021

[22] Kinguin Affiliate
https://www.kinguin.net/kinguin-mafia
06/2021

[23] Kinguin Sell
https://www.kinguin.net/sell-on-kinguin
06/2021

[24] Visual Studio Code
https://code.visualstudio.com/?wt.mc_id=DX_841432
06/2021

[25] Electron
https://www.electronjs.org
06/2021

[26] Git
https://git-scm.com
06/2021

[27] Oh My ZSH
https://ohmyz.sh
06/2021

[28] WSL
https://docs.microsoft.com/en-us/windows/wsl/install-win10
06/2021

[29] Django, campos relacionales
https://docs.djangoproject.com/en/3.0/ref/models/fields/#module-django.db.models.fields.related
06/2021

[30] Django, instancias
https://docs.djangoproject.com/en/3.2/ref/models/instances/
06/2021

[31] Django deployment checklist
https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
06/2021

[32] Dia
http://dia-installer.de
06/2021

[33] HTML5
https://es.wikipedia.org/wiki/HTML5
06/2021

[34] CSS3
https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
06/2021

[35] Como se carga el CSS
https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/How_CSS_works
06/2021

[36] JavaScript
https://www.javascript.com
06/2021

[37] MDB4
https://mdbootstrap.com/docs/b4/jquery/
06/2021

[38] Python
https://www.python.org
06/2021

[39] PostGreSQL
https://landing.aiven.io/en/aiven-for-postgresql/
06/2021

[40] PgAdmin4
https://www.pgadmin.org
06/2021

[41] DBeaver
https://dbeaver.io
06/2021

[42] BootStrap
https://getbootstrap.com
06/2021

[43] SQLite
https://www.sqlite.org/index.html
06/2021

[44] Heroku
https://www.heroku.com
06/2021

[45] Cloud Hosting
https://rockcontent.com/es/blog/cloud-hosting/
06/2021

[46] App.json Heroku Button Deploy
https://devcenter.heroku.com/articles/app-json-schema
06/2021

[47] Dockers
https://en.wikipedia.org/wiki/Docker_(software)
06/2021

[48] Estrategias para llevar a producción (deploy)
https://blog.heroku.com/six-strategies-deploy-to-heroku
06/2021

[49] Django
https://www.djangoproject.com
06/2021

[50] Documentación Django
https://docs.djangoproject.com/en/3.2/
06/2021

[51] Curso de programación web con Python, Universidad de Harvard
https://cs50.harvard.edu/web/2018/
04/2020

[52] Amazon Web Services
https://aws.amazon.com/es/what-is-aws/
06/2021

[53] Deployment checklist
https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
06/2021

[54] Signals Django
https://docs.djangoproject.com/en/3.2/ref/signals/
06/2021

[55] Lenguaje Django template
https://docs.djangoproject.com/en/3.2/ref/templates/language/
06/2021

[56] Static files Django
https://docs.djangoproject.com/en/3.2/howto/static-files/
06/2021

[57] Static files Django
https://stackoverflow.com/questions/22700097/how-to-point-correctly-to-static-image-in-django
06/2021

[58] Static files Django
https://docs.djangoproject.com/en/3.0/howto/static-files/deployment/
06/2021

[59] Django Models
https://docs.djangoproject.com/en/3.2/topics/db/models/
06/2021

[60] Django Models, Queries
https://docs.djangoproject.com/en/3.2/topics/db/queries/
06/2021

[61] Django Forms API
https://docs.djangoproject.com/en/3.2/ref/forms/api/
06/2021

[62] Django Forms Fields
https://docs.djangoproject.com/en/3.2/ref/forms/fields/
06/2021

[63] Django Forms Widgets
https://docs.djangoproject.com/en/3.2/ref/forms/widgets/
06/2021

[64] Validar un form Parte 1
https://docs.djangoproject.com/en/3.2/ref/forms/validation/
06/2021

[64] Validar un form Parte 2
https://docs.djangoproject.com/en/3.2/ref/validators/
06/2021

[65] Validar un form Parte 3
https://docs.djangoproject.com/en/3.2/ref/models/instances/#validating-objects
06/2021

[66] Estilizar un form
https://simpleisbetterthancomplex.com/article/2017/08/19/how-to-render-django-form-manually.html
06/2021

[67] Django model field types
https://docs.djangoproject.com/en/3.0/ref/models/fields/#model-field-types
06/2021

[68] Django model field types, opciones
https://docs.djangoproject.com/en/3.0/ref/models/fields/#common-model-field-options
06/2021

[69] Django heredando de user model
https://docs.djangoproject.com/en/dev/topics/auth/customizing/#specifying-a-custom-user-model
06/2021

[70] Django urls
https://docs.djangoproject.com/en/3.2/topics/http/urls/
06/2021

[71] Django Admin
https://docs.djangoproject.com/en/3.2/ref/contrib/admin/
06/2021

[72] Django Tests
https://docs.djangoproject.com/en/3.2/topics/testing/
06/2021

[73] Django views
https://docs.djangoproject.com/en/3.2/topics/http/views/
06/2021

[74] Django Class-based views
https://docs.djangoproject.com/en/3.2/topics/class-based-views/intro/
06/2021

[75] Django Class-based views Parte 2
https://docs.djangoproject.com/en/3.2/topics/class-based-views/generic-display/
06/2021

[76] Django Class-based views Parte 3
https://docs.djangoproject.com/en/3.2/topics/class-based-views/generic-editing/
06/2021

[77] Django Class-based views Parte 4
https://docs.djangoproject.com/en/3.2/topics/class-based-views/mixins/
06/2021

[78] AWS S3 App
https://django-storages.readthedocs.io/en/latest/backends/amazon-S3.html#
06/2021

[79] Django-allauth App
https://django-allauth.readthedocs.io/en/latest/installation.html
06/2021

[80] Crispy forms App
https://django-crispy-forms.readthedocs.io/en/latest/install.html
06/2021

[81] Django-countries App
https://pypi.org/project/django-countries/
06/2021

[82] Daguerre App
https://django-daguerre.readthedocs.io/en/stable/index.html
06/2021

[83] Ciclo de vida en cascada
https://es.wikipedia.org/wiki/Desarrollo_en_cascada#:~:text=En%20Ingeniería%20de%20software%20el,ordena%20rigurosamente%20las%20etapas%20del
07/2021

[84] Ciclo de vida en cascada
https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada
07/2021

[85] Ciclo de vida en cascada
https://www.crehana.com/es/blog/tech/modelo-en-cascada/
07/2021

[86] Django Mixins
https://docs.djangoproject.com/en/3.2/topics/class-based-views/mixins/
07/2021

[87] Django Ejemplos relaciones 1 a 1
https://docs.djangoproject.com/en/3.0/topics/db/examples/one_to_one/
07/2021

[88] Django Ejemplos relaciones 1 a N
https://docs.djangoproject.com/en/3.0/topics/db/examples/many_to_one/
07/2021

[89] Django Ejemplos relaciones N a N
https://docs.djangoproject.com/en/3.0/topics/db/examples/many_to_many/
07/2021

[90] Django MTV
https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names
07/2021

[91] Django settings.py
https://docs.djangoproject.com/en/3.2/ref/settings/
07/2021

[92] Django Static files
https://docs.djangoproject.com/en/3.2/topics/files/
07/2021

[93] Django Static files
https://docs.djangoproject.com/en/3.2/topics/files/#the-built-in-filesystem-storage-class
07/2021

[94] Django Google auth
https://www.youtube.com/watch?v=kj9llVn5vJI
07/2021

[95] Django Google auth
https://www.youtube.com/watch?v=NG48CLLsb1A
07/2021

[96] Repositorio del proyecto
https://github.com/Sovengar/gamingwebapp
07/2021

Anexo I
Django

Django [49] es un framework open source basado en Python para el desarrollo web en Python. Cuenta con una documentación muy pulida [50] y resulta muy fácil encontrar recursos y ayuda para su aprendizaje. Un proyecto en Django es una carpeta con el nombre del propio proyecto, que adicionalmente estará compuesto de apps (que se podrían entender como módulos del proyecto) ubicadas en sus propias carpetas [51].

Django sigue la arquitectura MTV (Model Template View) que consta de tres componentes principales:

- Modelo (Model).- Comprende toda la funcionalidad CRUD relacionada con el acceso a la base de datos (altas, consultas, actualizaciones y bajas). Una parte importante del modelo es el fichero "models.py", que define la estructura de datos de la aplicación, haciendo uso del ORM (Object Relational Mapping) de Django. Un proyecto completo puede estar formado por varias aplicaciones (organizadas en

carpetas dentro del propio proyecto) y cada una de ellas debe contener su propio fichero models.py.

- Plantilla (Template).- Las plantillas son ficheros HTML que contienen embebido etiquetas de plantilla de Django (entre llaves '{...}') para poder incrustar en ellos elementos de Python que no pueden ser directamente ejecutados por el navegador. Cada aplicación contendrá una carpeta "templates" con sus plantillas que definen cómo se van a visualizar los contenidos en el navegador.

- Vista (View).- Todas las vistas de una aplicación se definen en el fichero 'views.py' (dentro de la carpeta de la aplicación), adicionalmente, el fichero 'urls.py' determina qué vista debe ejecutarse en función de la URL solicitada desde el navegador. La vista, según corresponda, llamará al modelo para obtener datos o hacer alguna modificación, o llamará a alguna de las plantillas para mostrar en el navegador, es decir, y determina qué es lo que debe mostrarse al usuario final.

La nomenclatura de Django puede resultar confusa, ya que la analogía con MVC (Modelo, Vista, Controlador) sería la siguiente [90]:

- Modelo MVC ⇔ Modelo MTV
- Vista MVC ⇔ Plantillas MTV + Vista MTV
- Controlador MVC ⇔ El propio framework Django

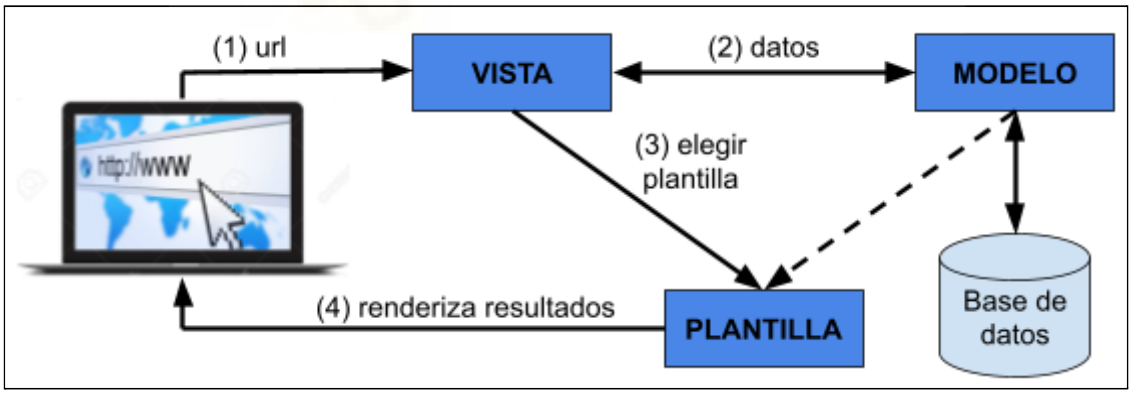

Figura AI.1 - Modelo MTV de Django

AI.1 - CARPETA PRINCIPAL DEL PROYECTO

La carpeta principal del proyecto es una carpeta con el nombre del propio proyecto (en nuestro caso 'gamingwebapp', ver figura AI.1) que incluye el siguiente contenido:

- __init__.py: Convierte un directorio en un módulo (paquete) que contiene otros módulos para poder importarlos.
- asgi.py: Define una interfaz de comunicación asíncrona con el servidor (el acrónimo asig significa '*Asynchronous Server Gateway Interface*')
- urls.py: Define las rutas del proyecto a las distintas aplicaciones.
- wsgi.py: Similar a 'asgi.py', establece una interfaz de comunicación síncrona con el servidor web (el acrónimo wsgi significa '*Web Server Gateway Interface*').
- settings.py: Contiene configuraciones básicas, como el time zone, aplicaciones instaladas en el proyecto, la conexión con la base de datos, static y media files, configurar heroku y aws s3, etc. [91]

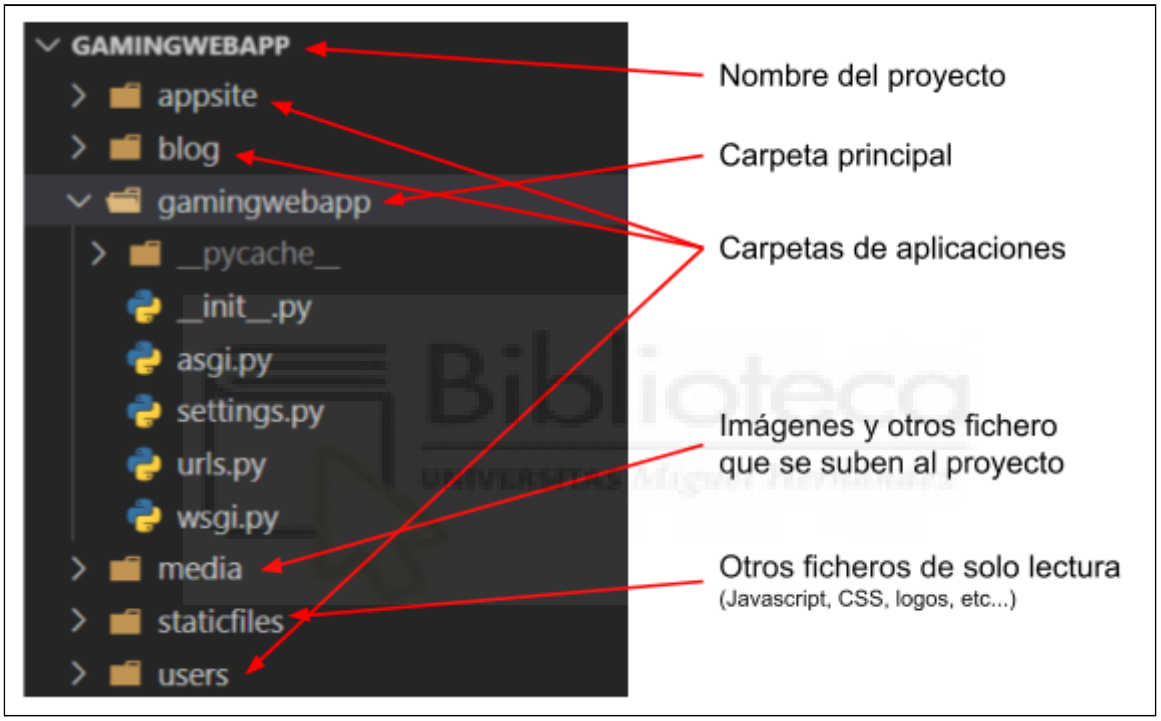

Figura AI.2 - Ejemplo carpetas de un proyecto Django

Por defecto todo proyecto en Django genera una BD en SQLite. Para usar otro motor de BD hay que especificar los parámetros ENGINE, NAME, USER, PASSWORD, HOST y PORT en el fichero 'settings.py'.

Otras variables importantes son la secret key, la cual nos permite ejecutar la aplicación si tenemos la contraseña; y la variable DEBUG, que si está a True devuelve una traza de error cuando ocurra uno, entre otras funciones que se comentarán más adelante.

También, dentro del fichero 'settings.py', se declara la lista '*INSTALLED_APPS*' que contiene todas las apps del proyecto, el desarrollador tiene que incluir las apps que haya creado, por defecto ya vienen enlazadas las aplicaciones nativas de Django, estas son:

- admin.- Ofrece una interfaz gráfica para añadir o modificar datos en los modelos (tablas de la base de datos), para acceder a ella hay que identificarse con una cuenta con permisos de administrador.

- auth.- Proporciona una serie de medidas y configuraciones para la seguridad, como lo son la gestión de las cuentas de los usuarios, grupos o permisos y las sesiones de los usuarios basadas en cookies.

- contenttypes.- Permite seguir el rastro de los modelos instalados en nuestro proyecto, ofreciendo una interfaz genérica para trabajar con los modelos.

- sessions.- Gestiona las sesiones de los usuarios, incluidos los anónimos. Permite guardar información en el lado del servidor de los visitantes y abstrae las cookies enviadas y recibidas.

- staticfiles.- Se encarga de almacenar y mostrar los ficheros, un static file es, como indica el nombre, un fichero estático, como una foto o un fichero .js o .css, Django maneja los ficheros estáticos de manera muy particular [56] [57][58], dependiendo de si lo ejecutamos en local o en producción (y el tipo de producción) variará su implementación.

- messages.- Permite usar mensajes customizados o los que ofrece Django para mostrar mensajes en las templates (p.e., si el usuario se ha logueado correctamente se puede mostrar en la plantilla el mensaje success con el texto deseado).

AI.2 - CARPETAS DE APLICACIONES

Cada app creada por el desarrollador tendrá su correspondiente carpeta, no obstante, no todas las apps tienen carpeta, por ejemplo, las incluidas de terceros o las que vienen por defecto con cada proyecto.

Cada app puede tener los siguientes ficheros (no todos son obligatorios): admin, apps, forms, models, signals, tests, urls, views y las siguientes carpetas: migrations, _pycache, templates (en las siguientes referencias [54][59-77] se puede encontrar información sobre su uso y configuración). A continuación se describen los ficheros y carpetas más relevantes:

- Fichero urls: Debe ser creado manualmente, se pueden introducir las rutas de nuestra aplicación, siempre y cuando se puedan acceder a ellas desde el fichero urls.py de la carpeta principal [70].

- Fichero views: Se crearán las funciones que se asocian con las rutas en urls.py. Dependiendo de las necesidades, la vista se hará de una manera u otra, puede ser una creada manualmente o heredando de una clase, como ListView, DetailView, DeleteView, UpdateView, CreateView, etc. No hay un patrón que se repita, pero si tienen en común el uso de la variable diccionario context, en la cual se tendrán los datos que se quieren mostrar en el template u otros parámetros de validación [73] [74][75][76][77].

- Fichero models: Es donde se diseñan las tablas de la app para la BD del proyecto. Se escribe en python, no SQL, ya que se usa el ORM que incorpora Django, este hará la conversión de Python al SQL necesario para el SGBD que estemos usando. Además, el fichero podrá contener funciones clásicas, que usarán los modelos; en el Anexo AI.4 se explica más en detalle.

- Fichero signal: Se especifican los triggers, por ejemplo, si se produce el evento X haz Y. Hay muchos tipos de señales, por ejemplo antes de guardar un registro o después, más ejemplos [54].

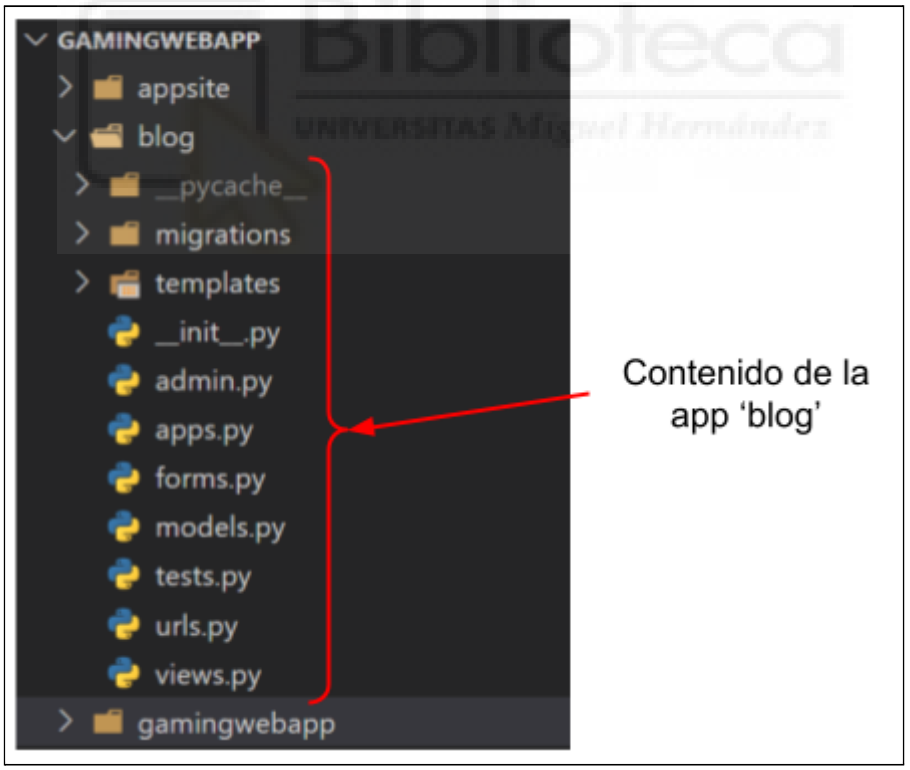

Figura AI.3 - Carpeta de la aplicación '*blog*' en Django

- Fichero forms: Los formularios HTML, en Django, se hacen mediante Python, heredando ya sea de la clase forms.Form o de la clase forms.ModelForm [61][62] [63][64][65][66].

- Fichero admin: Modifica la lógica e interfaz de la app admin, en el proyecto. Es decir, si se necesita que en el panel de admin salga cierto modelo, se debe añadir en este fichero, siempre y cuando se tenga en cuenta que dicho modelo pertenezca a dicha app. Por otra parte, gestionar la visibilidad de las relaciones 1 a N también se hacen en este fichero [71].

- Fichero app: Es donde se crea realmente la app, la app se interpreta como una clase, así puede ser importada en settings.py. Si se utilizan signals se deben añadir aquí con el método def ready(self): import appname.signals

- Fichero tests: Realiza los testeos. Es necesario hacer testeos a funciones o inputs para asegurar que funcionan bajo todos los escenarios. Por ello, los ajustes que se harán en este fichero no se aplicarán a nuestra BBDD, ya que crea una nueva. Se ejecutan X tests, cada uno aislado del resto, de tal forma que 1 test no puede influenciar otros tests. Un test por cada función. Lo normal es una función setup para iniciar la data y el resto funciones para testear los modelos y/o las vistas [72].

- Carpeta migrations: Contiene una serie de ficheros python ordenados, con el comando makemigrations detectará los cambios realizados y lo almacenará en un fichero. Estos cambios no se aplican a la BD, para eso es necesario ejecutar el comando 'migrate', que leerá estos ficheros, los transformará a SQL y los aplicará a la BD.

- Carpeta templates: Contiene los ficheros HTML, usan el lenguaje Template de Django [55] para mostrar los datos que se reciben por el context o el request GET/POST. Para evitar colisión de nombres (ya que Django busca el fichero en todas las rutas), se recomienda crear una carpeta dentro de la app con el nombre de la app. Por tanto, si se usa la ruta "appname/fichero.html", en físico debe ser appname/templates/appname/fichero.html

AI.3 - FICHEROS

Django hace una distinción entre los ficheros que inicialmente forman parte del proyecto [56] y los ficheros que se añaden con posterioridad mediante un upload [92]. Los ficheros estáticos son los ficheros que un proyecto en Django puede necesitar para funcionar o mostrar la apariencia deseada, por ejemplo, los ficheros .js, .css, .svg, .png, etc. Para la gestión de ficheros estáticos Django dispone de la app nativa '*django.contrib.staticfiles*' que debe incluirse en la variable 'INSTALLED_APPS ' en el fichero '*settings.py*', además, en el mismo fichero de configuración, se deben establecer las siguientes variables:

- STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
  STATIC_ROOT: Es la ruta absoluta a un directorio en el que el script "collectstatic" de Django reunirá todos los archivos estáticos referenciados en nuestras plantillas.

- STATIC_URL = '/static/'
  STATIC_URL: Es la localización URL base desde la cual se servirán los archivos estáticos, por ejemplo en una CDN. Se usa para variables de plantilla estáticas a las que se acceden en nuestra plantilla base.

De esta manera, Django ya reconoce la carpeta '*staticfiles*' dentro del directorio base para poder acceder a los recursos que dicha carpeta contiene. Para su uso en un template, primero hay que indicarle al motor del template que use la carpeta '*static*' en dicha plantilla con el siguiente tag especial de Django:

```
{% load static %}
```

A continuación ya se pueden referenciar los ficheros contenidos en dicha carpeta, por ejemplo, para mostrar una imagen con el tag '<img>' se haría lo siguiente:

```
<img src="{% static 'appsite/img/logo.png' %}">
```

La imagen estaría en la ruta '*/static/appsite/img/logo.png*'.

En desarrollo, si se usa django.contrib.staticfiles y DEBUG=True no será necesario ejecutar el comando collectstatic, ya que lo hará Django automáticamente.

Por otra parte, están los ficheros que se suben, y se gestionan con el concepto de MEDIA. Por defecto, Django guarda los ficheros que se suben localmente mediante las variables settings MEDIA_ROOT y MEDIA_URL. Esto es así por motivos de rendimiento, desde el modelo/tabla, se guarda un objeto de tipo File usando el Built-In system que trae Django [93], como por ejemplo ImageField(), este campo tiene un parámetro en el que especificar dónde se va a guardar (dentro de la carpeta MEDIA) los ficheros de este campo.

- MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
  MEDIA_ROOT: Es la ruta absoluta al directorio que contiene los ficheros subidos por los usuarios. MEDIA_ROOT y STATIC_ROOT deben tener valores diferentes por motivos de seguridad.

- MEDIA_URL = '/media/'
  MEDIA_URL: URL que se encarga de servir el contenido de MEDIA_ROOT. MEDIA_URL y STATIC_URL deben tener valores diferentes.

Para que el navegador pueda trabajar con el contenido 'media' cuando se trabaja en local o se está en etapa de desarrollo, es necesario importar la siguiente función a la variable urlpatterns del fichero urls.py de la carpeta principal.

```
urlpatterns = [
    # ... the rest of your URLconf goes here ...
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Esto funcionará sólamente en Debug=True y sólo si el prefijo es local, es decir, /media/, y no una URL, es decir, http://media.example.com/.

AI.4 - ORM

Una de las características más importantes de Django es su ORM (Object-Relational Mapper), el cual permite interactuar con la base de datos, como se haría con SQL, pero con Python, y obteniendo así el resultado en Python también.

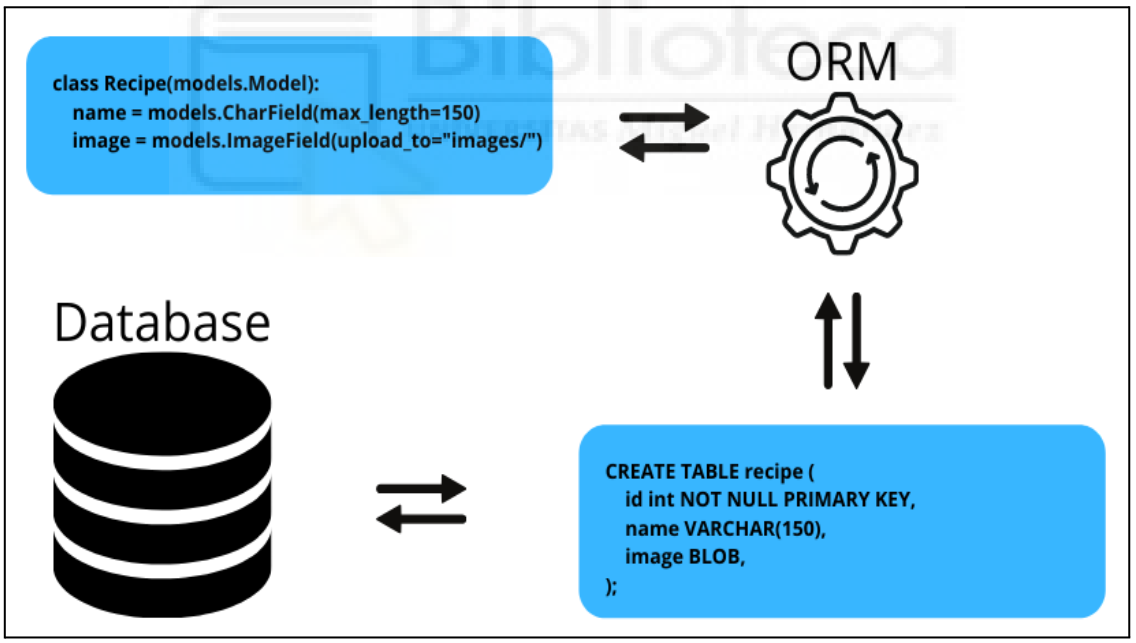

Figura AI.4 - Diagrama ORM

AI.4.1.- Modelos

Como se puede observar en la Figura AI.4, un modelo es una tabla, y para tener un modelo se necesita crear una clase que herede de models.Model, con las variables de la clase se establecen los campos de la tabla. El nombre de la tabla de este modelo será

‘*appname_modelname*’. Django añade automáticamente el campo id (id = models.AutoField(primary_key=True)).

En la referencia [67] se puede observar los distintos tipos que puede tener un campo o variable, como en la Figura AI.4 se tiene el campo CharField o ImageField. Hay que comentar en especial 2:

- .FileField(upload_to=None, max_length=100, **options)
  upload = models.FileField(upload_to='uploads/%Y/%m/%d/')
  El fichero será guardado en MEDIA_ROOT/uploads/2021/07/X

- .ImageField(upload_to=None, height_field=None, width_field=None, max_length=100, **options)

Ambos necesitan un MEDIA_ROOT, MEDIA_URL y ambos tienen propiedades y métodos que heredan, como .name, .size, .url, .open(path), .close(), .save(name, content, save=boolean), .delete(save=boolean)

Como se puede observar, hay campos que necesitan ciertos parámetros y tienen otros opcionales, estos parámetros opcionales son usados en la mayoría de campos, como lo son blank, choices, default, unique, validators, etc. Listado completo en [68].

Por otra parte, hay una distinción entre los campos normales como los vistos en [67] y los campos que forman las relaciones entre las tablas, como se pueden ver en [29]:

- ForeignKey: No es necesario especificar qué atributo es en la otra tabla, con indicar el modelo es suficiente. Si está en otro fichero (otro models.py) se usará ‘*appname.modelname*’, si es consigo mismo ‘*self*. Django añadirá automáticamente el sufijo “_id” al nombre del campo a la hora de crear el campo/columna.

- ManyToManyField: Django crea por debajo un join table para representar la relación. Este campo sólo debe estar en uno de los 2 modelos, no importa cual. Sin embargo, generalmente se suele poner en el objeto que va a ser editado en un form. No obstante, se puede crear manualmente la tabla N a N con el atributo through y through_fields o con 2 atributos foráneos.

- OneToOneField: Django no crea una tabla para esta relación, no obstante, es similar a un ForeignKey pero con unique=True, donde la relación reversa devolverá un sólo objeto.

Por último, para ver más utilidades que se pueden hacer con los modelos, como usar funciones para calcular la edad, usar la clase interna META para tener más metadata en tu

clase o unique_together para tener claves primarias compuestas, véanse las referencias [59][69].

AI.4.2.- Migraciones

Django ofrece el concepto de '*migrations*' para tener un control de versiones del esquema de la base de datos que se utiliza. Con el comando *'makemigrations'* Django detecta los cambios del esquema actual al esquema modificado, genera el código necesario para aplicar los cambios del estado actual al modificado en lenguaje Python y lo guarda creando un fichero. Los ficheros siguen un orden, es decir, si el estado actual era 0007_..., se ejecuta el comando *'makemigrations'* y detecta cambios, guardará estos cambios creando el fichero 0008_…

Para aplicar estos cambios, se utiliza el comando *'migrate'*, este comando leerá los ficheros generados por 'makemigrations', transformará el código Python en código SQL y los aplicará a la Base de datos.

AI.4.3.- Queries

Se entiende por Queries lo que se puede hacer con la API de Django, como es crear, recoger, actualizar y borrar objetos. Hay muchas maneras de crear un registro:

- Clase(...)
- Clase.objects.create(...)
- .add()
- .set(obj1, obj2, obj3, ...)

Tras crearlo en Python, hay que crear el registro, para ello se usa el método save(). Si se almacenan en una variable, la variable pasa a apuntar al registro creado.

Para actualizar un registro, es equivalente a actualizar un objeto; se obtiene la referencia al objeto/registro, se modifica la variable o variables y se ejecuta el método save() para ese objeto. Para eliminar un registro, se obtiene la referencia al objeto/registro y se ejecuta su método delete().

Para obtener un registro o una serie de ellos hay que entender el concepto de QuerySet. Un QuerySet es un objeto que contiene objetos, es el equivalente al resultado de un SELECT en SQL, donde el propio SELECT sería el objeto y dentro de él cada registro sería un objeto. En general, los resultados de un QuerySet no son solicitados de la base de datos hasta el momento que se solicitan realmente. Algunos ejemplos de Querysets:

- *'Clase.objects.xxx'* o '*objetoinstanciado.related_name.xxx'* devuelven un queryset.
- .first(): Devuelve el primer objeto del queryset.
- .all(): Devuelve un QuerySet con todos los registros de dicha tabla.
- .filter(): Equivalente al WHERE en SQL.

Si se quiere profundizar en las queries y no sólo unas pinceladas de lo más básico, se pueden consultar las referencias [60][30][87][88][89].

AI.5 - APPS DE TERCEROS

Como muestra la Figura AI.5, para poder usar AWS S3 se ha instalado el paquete/app '*storages*' [78], para el login con Google se ha instalado la app '*allauth*' [79][94][95], para estilizar los forms la app '*crispy_forms*' [80], para poder usar un control de formulario tipo select con todos los países la app '*django_countries*' [81], y para redimensionar imágenes desde el template la app '*daguerre*' [82].

```
#AWS S3
'storages',

#Third-party apps
'crispy_forms',
'widget_tweaks',
'django_countries',
'daguerre',

#Third party app Allauth
'django.contrib.sites',
'allauth',
'allauth.account',
'allauth.socialaccount',
'allauth.socialaccount.providers.google'
```

Figura AI.5 - Apps de terceros

AI.6 - DJANGO CLI

En esta sección se enumeran los comandos o scripts más utilizados en Django:

- `django-admin startproject projectname`
  Crea un proyecto vacío

- `python manage.py startapp appname`
  Crea una app dentro del proyecto

- `python manage.py runserver 0:5000`
  Ejecuta un servidor local con python que se autorefresca con cada request, no con cada archivo añadido. Lo que se ejecuta internamente sería el comando 'gunicorn gamingwebapp.wsgi --log-file -'

- `python manage.py shell`
  Levanta un shell similar al intérprete de python.
  Nos permite hacer modificaciones a la base de datos.

- `python manage.py makemigrations`
  Busca cambios en models.py y genera un file 'migration' migrations/0001_initial.py p.e, contendría los cambios necesarios en la DB.

- `python manage.py migrate`
  Aplica las migrations/*.py en orden de todas las aplicaciones.

- `python manage.py appname migrate [valor]`
  Aplica la migración de ese .py concreto.
  'python manage.py appname migrate zero' revertiría todas las migraciones.

- `python manage.py sqlmigrate appname [número del fichero]`
  Muestra el código en SQL de ese fichero.

- `python manage.py test`
  Ejecuta el código de test.py creando BD auxiliares.

- `python manage.py createsuperuser`
  Django se encarga de añadir el superuser a la tabla users.

Anexo II
Heroku

Heroku es un servicio de tipo PaaS (*Platform as a Service*) que permite desarrollar y correr aplicaciones en la nube, dando soporte a varios lenguajes de programación, entre ellos, Python. Está basado en contenedores y, dado que es un servicio de plataforma, facilita a los administradores de aplicaciones las tareas de gestión, así como el escalado de las mismas si fuera necesario, permitiendo que estos se centren fundamentalmente en el desarrollo. Para que Heroku funcione hay que conocer sus tres aspectos principales:

- Los "dynos".- Es el nombre con el que se denomina a los clusters en Heroku. Por defecto, una app gratuita utiliza un dyno gratuito. Un dyno es como una contenedor ligero que ejecuta los comandos ejecutados en un fichero llamado Procfile. Un dyno gratuito dejará de ejecutarse tras media hora de inactividad, o lo que es lo mismo, tras media hora de no recibir tráfico.

- Los ficheros clave.- Definen formas de operar, por ejemplo, detectan si una app está en Python, o PHP, que procesos se deben ejecutar, de qué tipo son, etc.

- Las variables de entorno.- Son variables almacenadas en zonas no visibles del programa o repositorio, de tal manera que puedan utilizarse pero sin que se pueda saber su valor. Esto es útil, por ejemplo, para almacenar contraseñas de APIs de terceros. Heroku buscará estas variables en el fichero .env si trabajamos con Heroku en local, o en config:var si está desplegado.

AII.1.- FICHEROS CLAVE EN HEROKU

El fichero '*Procfile*' es un fichero que debe estar en el directorio raíz de la aplicación, dentro irá el código a ejecutar para levantar la app.

> Ejemplo Django --> web: gunicorn project_name.wsgi --log-file -
> Es un proceso de tipo HTTP routing, por tanto recibirá tráfico web cuando se haya levantado.

Si se trabaja en windows y se quiere ejecutar Heroku en local, en vez de usar el fichero Procfile se tendrá que crear uno llamado Procfile.windows.

> Ejemplo Django --> web: python manage.py runserver 0.0.0.0:5000

Hay diferentes maneras de hacer el deploy, si se elige el deploy con Github Button, será necesario incorporar el fichero '*App.json*' [46], en él se declaran las variables de entorno, add-ons y otra información requerida para ejecutar la app en Heroku. Se ejecutará desde el button, ya sea en la página oficial [44] o en el README.

En el fichero '.env' se establecen todas las variables de entorno que necesite nuestra aplicación para ser ejecutada en local (cuando se ejecute la aplicación con el comando Heroku local). Por otra parte, para que el host conozca estas variables se usará el comando heroku config:set CONFIGVAR_NAME="VALUE"

El fichero '*runtime.txt*' se establece qué lenguaje usará la aplicación, por ejemplo, en el caso del presente proyecto, python 3.7.6.

Cada vez que se hace un deploy, Heroku revisa el fichero '*requirements.txt*' e instala las dependencias que lee en él. Esta lista debe ser actualizada cada vez que añadamos un módulo nuevo. Para obtener una lista de todos los módulos que se están usando se ejecuta el comando correspondiente, por ejemplo '*pip freeze*' o '*pip list*', en python. Además, siempre se puede forzar su instalación con '*pip install -r requirements.txt*'.

Adicionalmente, '*.gitignore*', no pertenece a Heroku pero es recomendable añadir los ficheros clave ya que contienen mucha información útil.

AII.2.- EJECUTAR LA APLICACIÓN EN LOCAL

Puede parecer absurdo esta implementación, pero en vez de ejecutar el servidor en local con Django (Django ejecuta desde su script manage.py el servidor gunicorn) se puede hacer con Heroku, si se quiere probar algo puntual en local, no se podría porque Django no leería las variables del fichero .env, se tendrían que especificar también en las variables de entorno de Windows, o en Linux/Mac en el fichero .bashrc

Para solucionar esto, la manera ética sería comprando/actualizando a un nivel Hobby o profesional, de esta manera, también se puede, mediante comando, especificar que se ejecuten un número determinado de dynos, cada uno de ellos ejecutando tu proceso de tipo web. La manera no ética es configurando mediante una página de terceros un bot que ejecute pings a cierta dirección ip, de esta manera se mantendría el tráfico y por tanto el dyno no dejaría de ejecutar el proceso.

Para ejecutarlo en local, se obviarán estos dos comandos:

1. pip install -r requirements.txt
2. python manage.py collecstatic

Ya que al estar trabajando en local, se supone que ya están todas las dependencias instaladas, pero si se cambia de dispositivo se tendría que ejecutar el primero de ellos. Por otra parte, el comando 'python manage.py collecstatic' se ejecutará automáticamente (siempre y cuando esté la app django.contrib.staticfiles en INSTALLED_APPS y DEBUG=True). El comando para ejecutar en local varía según el Sistema Operativo:

- Linux:
  ```
  heroku local web
  ```

- Windows:
  ```
  heroku local web -f Procfile.windows
  ```

AII.3.- FORMAS DE DESPLIEGUE

Hay varias maneras de hacer Deploying con Heroku y no son excluyentes, es decir, se pueden combinar. El método a escoger depende del proceso específico de puesta a producción, los requerimientos y las apps.

Para equipos pequeños que están empezando, deploying con Git es la mejor opción debido a su simplicidad y ajuste del contexto. También Heroku button sería igual de sencillo, permitiendo hacer el deploy de toda la aplicación con un sólo click en el botón. Si en Git se

usa con mucha frecuencia CI (Continuous Integration) y CD (Continuous Delivery), podría ser más conveniente integrar con GitHub, ya que se ejecutarán automáticamente los deploys al hacer un commit.

Si los requerimientos son más sofisticados, se pueden usar otras estrategias, por ejemplo, para usar una plataforma de desarrollo personalizada (o stack custom), sería recomendable usar Dockers [47]. Como su palabra indica, un docker es un contenedor, es una forma de empaquetar todas las dependencias que necesita el programa para funcionar, no instalándolas de manera directa en el propio equipo (el propio sistema operativo), sino en el Docker. Se podría ver como una especie de máquina virtual.

Como se ha dicho anteriormente, estos métodos no son excluyentes, por ejemplo, es posible hacer el deploy de un Docker, creando una review app desde GitHub con Pull request, hacer el testeo de la app para entonces hacer el deploy de la versión final usando git push. [48]

AII.4.- HEROKU CLI

Como se ha visto, hay que instalar Heroku en el propio equipo, con él vendrá la línea de comandos de Heroku, a continuación se muestra una breve lista de los comandos que incorpora heroku y qué hace cada uno:

- `heroku login`
  Permite vincular la cuenta en Heroku con el terminal. Es decir, un login desde el terminal.

- `heroku create appname`
  Crea una app vacía y un repositorio remoto para la misma, además añade a nuestro git el repositorio remoto con 'git remote add heroku link.git'. El parámetro 'appname' es opcional, si se omite se genera uno aleatorio.

- `heroku open`
  Abre el navegador o una pestaña en el mismo con la aplicación web.

- `heroku local`
  Ejecuta todos los procesos definidos en el fichero Procfile. Utiliza las config vars del fichero .env, utiliza la base de datos local.

- `heroku local web`
  Ejecuta el proceso web definido en el fichero Procfile.

- `heroku local -f [valor]`
  Establece manualmente qué fichero Procfile usar.

- `heroku local -e [valor]`
  Establece manualmente qué fichero .env usar.

- `heroku local -p [valor]`
  Establece manualmente qué puerto utilizar.

- `heroku run [valor]`
  Ejecuta un comando desde la terminal del servidor.

- `heroku config`
  Muestra las config vars asignadas en el servidor.

- `heroku config:set [VNAME=VALUE]`
  Añade la config var a las config vars que dispone el servidor.

- `heroku config:unset [VNAME]`
  Elimina la config var de las config vars que dispone el servidor.

- `heroku ps`
  Muestra el número de dynos que se están ejecutando.

- `heroku ps:scale web=[Número]`
  Modifica el número de dynos para el proceso web.

- `heroku addons`
  Muestra los addons instalados.

- `heroku addons:create [add-onanme]`
  Añade un add-on

- `heroku addons:open [add-onanme]`
  Ejecuta el add-on

- `heroku logs --tail`
  Muestra los logs del servidor, p.e HTTP requests.

- `heroku releases`
  Muestra las versiones de la app

- `heroku rollback v[Número]`
  Permite volver a la versión especificada

- `heroku pg`
  Muestra la información básica de la BD postgreesql

- `heroku pg:psql`
  Conecta remotamente a la BD y permite ver los registros.

- `heroku pg:reset DATABASE`
  Resetea la Base de datos PostgreSQL.

Anexo III
Casos de uso

En este anexo se presentan las plantillas descriptivas (tablas) de los 36 casos de uso de los diversos usuarios de la aplicación.

Tabla AIII.1.- Casos de uso 1: Registrar cuenta

C.U. 1	*Registrar cuenta*
Actores	Invitado
Descripción	Un invitado o guest en la página puede registrarse en la aplicación rellenando un formulario o con una cuenta en Gmail.
Dependencias	
Precondición	
Secuencia normal	P1 - El usuario introduce sus credenciales en el formulario o pulsa en el icono G, para iniciar sesión con la cuenta de Google. P2 - Pulsar el botón “Sign Up”.
Poscondición	Si el formulario es válido, se creará una cuenta, el actor invitado pasará a ser un usuario registrado.
Excepciones	● Si el formulario no es válido, se indica con un mensaje y se vuelve al formulario de registro.
Comentarios	Un administrador o superusuario también puede registrar una cuenta desde el panel de admin.

Tabla AIII.2.- Casos de uso 2: Iniciar sesión

C.U. 2	*Iniciar sesión*
Actores	Invitado
Descripción	El usuario inicia sesión en la aplicación, identificándose, ya sea con sus credenciales o con su cuenta de Google.
Dependencias	
Precondición	
Secuencia normal	P1 - El usuario introduce sus credenciales en el formulario o pulsa en el icono G, para iniciar sesión con la cuenta de Google. P2 - Pulsar el botón “Sign In”.
Poscondición	Si el usuario y la contraseña es correcta, dependiendo de sus permisos, el usuario invitado pasa a ser un usuario registrado, administrador o superusuario.
Excepciones	● P2- Si el usuario y la contraseña no son correctas se indica con un mensaje y se vuelve al formulario de identificación.
Comentarios	Un administrador o superusuario también puede registrar una cuenta desde el panel de admin.

Tabla AIII.3.- Casos de uso 3: Ver perfil

C.U. 3	*Ver perfil*
Actores	Usuario registrado, Administrador, Superusario, Vendedor
Descripción	El actor puede ver su perfil, el administrador también puede verlo desde la URI /admin.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - Pulsar en el avatar ubicado en el Navbar P2 - Pulsar en “Profile”
Poscondición	Si el usuario tiene una cuenta vendedor, se mostrarán sus datos
Excepciones	
Comentarios	

Tabla AIII.4.- Casos de uso 4: Modificar perfil

C.U. 4	*Modificar perfil*
Actores	Usuario registrado, Administrador, Superusario, Vendedor
Descripción	Un actor que ha accedido a su perfil puede modificar su username, su email o su foto de usuario. Opcionalmente puede solicitar un reseteo de contraseña.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - Pulsar en el avatar ubicado en el Navbar P2 - Pulsar en “Profile” P3 - Modificar P4 - Pulsar en el botón “Update profile”
Poscondición	
Excepciones	
Comentarios	El administrador y Superusuario también pueden modificarlo desde la URI /admin

Tabla AIII.5.- Casos de uso 5: Ver posts

C.U. 5	*Ver posts*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	Cualquier actor puede ver los posts desde la URI /blog o pulsando sobre el autor de un post ver los posts de dicho usuario
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsar en “Blog” en el Navbar
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede ver los posts desde la URI /admin

Tabla AIII.6.- Casos de uso 6: Ver post

C.U. 6	*Ver post*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	Cualquier usuario puede ver un post concreto
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsar en “Blog” en el Navbar P2 - Pulsar en “Read more” en el post correspondiente
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede ver un post desde la URI /admin

Tabla AIII.7.- Casos de uso 7: Crear post

C.U. 7	*Crear post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede crear un post
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado
Secuencia normal	P1 - Pulsar en el avatar ubicado en el Navbar P2 - Pulsar en “Make a new post” P3 - Completar el form P4 - Pulsar en “POST”
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede crear un post desde la URI /admin

Tabla AIII.8.- Casos de uso 8: Modificar post

C.U. 8	*Modificar post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El dueño del post o un admin/superusuario puede modificar un post
Dependencias	*C.U. 2, C.U. 7*
Precondición	El actor debe estar logeado. El actor debe haber creado el post previamente.
Secuencia normal	P1 - Pulsar en “Blog” en el Navbar P2 - Pulsar en “Read more” en el post correspondiente P3 - Pulsar en “Update” P4 - Modificar el post P5 - Pulsar en “POST”
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede modificar un post desde la URI /admin

Tabla AIII.9.- Casos de uso 9: Eliminar post

C.U. 9	*Eliminar post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El dueño del post o un admin/superusuario puede eliminar un post.
Dependencias	*C.U. 2, C.U. 7*
Precondición	El actor debe estar logeado. El actor debe haber creado el post previamente.
Secuencia normal	P1 - Pulsar en “Blog” en el Navbar P2 - Pulsar en “Read more” en el post correspondiente P3 - Pulsar en “Update” P4 - Modificar el post P5 - Pulsar en “POST”
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede eliminar un post desde la URI /admin

Tabla AIII.10.- Casos de uso 10: Ver artículos

C.U. 10	*Ver artículos*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver los artículos que se venden
Dependencias	
Precondición	
Secuencia normal	P1 - Desde el home o index, se ven artículos nuevos. P2 - Scroll hacia abajo para ver más juegos. P3 - Cambiar de página si hubiera más juegos.
Poscondición	
Excepciones	P1 - Si se pulsa el botón “ALL NEW RELEASES” se puede ver los juegos nuevos
Comentarios	Un administrador o superusuario puede ver los artículos desde la URI /admin. Desde el buscador también se pueden ver artículos.

Tabla AIII.11.- Casos de uso 11: Ver artículo

C.U. 11	*Ver artículo*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	Cualquier actor puede ver un artículo concreto
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsa sobre el artículo para una vista detallada del mismo.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede ver el artículos desde la URI /admin

Tabla AIII.12.- Casos de uso 12: Añadir a la cesta

C.U. 12	*Añadir a la cesta*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver un artículo concreto
Dependencias	*C.U. 2, C.U. 11*
Precondición	Se debe tener la vista del artículo presente.
Secuencia normal	P1 - Pulsa sobre el botón “ADD TO CART” según la opción que quieras añadir.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede añadir a la cesta desde la URI /admin

Tabla AIII.13.- Casos de uso 13: Ver cesta

C.U. 13	*Ver cesta*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver su cesta con los artículos.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - Pulsa sobre el icono con forma de carro de la compra, ubicado en el navbar en la parte derecha.
Poscondición	
Excepciones	Se puede acceder a la cesta desde el C.U. 12
Comentarios	Un administrador o superusuario puede añadir a la cesta desde la URI /admin

Tabla AIII.14.- Casos de uso 14: Modificar cesta

C.U. 14	*Modificar cesta*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede eliminar un artículo de su cesta.
Dependencias	*C.U.2, C.U. 13*
Precondición	Se debe de tener el carrito/cesta abierto
Secuencia normal	P1 - Pulsa sobre el icono rojo con una X blanca en la parte derecha del artículo que se quiere eliminar.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede modificar la cesta desde la URI /admin

Tabla AIII.15.- Casos de uso 15: Crear artículo

C.U. 15	*Crear artículo*
Actores	Administrador, Superusuario
Descripción	El actor puede crear un artículo desde la URI /admin
Dependencias	*C.U.2*
Precondición	El usuario debe estar logeado.
Secuencia normal	P1 - Acceder a la ruta /admin desde el navbar o tecleandola manualmente en la URL del navegador. P2 - En los models de Appsite, pulsar en el modelo Games. P3 - Pulsar en el botón “ADD GAME” situado en la esquina superior derecha.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.16.- Casos de uso 16: Modificar artículo

C.U. 16	*Modificar artículo*
Actores	Administrador, Superusuario
Descripción	El actor puede modificar un artículo desde la URI /admin
Dependencias	*C.U.2, C.U. 15*
Precondición	Es necesario que el artículo haya sido creado para poder modificarlo.
Secuencia normal	P1 - Acceder a la ruta /admin desde el navbar o tecleandola manualmente en la URL del navegador. P2 - En los models de Appsite, pulsar en el modelo Games. P3 - Pulsar en el artículo que se requiere modificar.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.17.- Casos de uso 17: Resetear contraseña

C.U. 17	*Resetear contraseña*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede solicitar un reset de contraseña.
Dependencias	*C.U.2, C.U. 3*
Precondición	El reseteo de contraseña se hace desde el Perfil.
Secuencia normal	P1 - Pulsar en el link “Forgot password?” P2 - Introducir el email de la cuenta. P3 - Confirmar el reset. P4 - Buscar el correo en tu aplicación de correos. P5 - Abrir el correo P6 - Pulsar en el enlace de reseteo de contraseña. P7 - Establecer la nueva contraseña.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede modificar la contraseña desde la URI /admin, visitando el model User.

Tabla AIII.18.- Casos de uso 18: Modificar permisos

C.U. 18	*Modificar permisos*
Actores	Superusuario
Descripción	El actor puede cambiar los permisos de cualquier actor.
Dependencias	*C.U.2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - Acceder a la ruta /admin desde el navbar o tecleandola manualmente en la URL del navegador. P2 - Abrir el modelo Users P3 - Abrir el usuario que se quiere modificar sus permisos P4 - Seleccionar los permisos en la sección “Permissions”
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.19.- Casos de uso 19: Ver opiniones

C.U. 19	*Ver opiniones*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver las opiniones de los artículos que se venden
Dependencias	*C.U. 11*
Precondición	Se debe tener la vista abierta con el artículo.
Secuencia normal	P1 - Pulsar en el tab “Opinions”
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede ver las opiniones desde la URI /admin.

Tabla AIII.20.- Casos de uso 20: Comentar opinión

C.U. 20	*Comentar opinión*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede comentar su opinión de un artículo.
Dependencias	*C.U. 2, C.U. 19*
Precondición	Se debe haber realizado el C.U. 19
Secuencia normal	P1 - Escribir tu opinión en el text input. P2 - Pulsar en el botón “POST”.
Poscondición	P1 - Si el actor ha comentado previamente, ya no puede comentar.
Excepciones	
Comentarios	Un administrador o superusuario puede comentar una opinión desde la URI /admin.

Tabla AIII.21.- Casos de uso 21: Editar opinión

C.U. 21	*Editar opinión*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede editar su opinión de un artículo.
Dependencias	*C.U.2, C.U. 20*
Precondición	Se debe haber realizado el C.U. 20
Secuencia normal	P1 - Pulsar en el botón “EDIT”.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede editar una opinión desde la URI /admin.

Tabla AIII.22.- Casos de uso 22: Borrar opinión

C.U. 22	*Borrar opinión*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede editar su opinión de un artículo.
Dependencias	*C.U.2, C.U. 20*
Precondición	Se debe haber realizado el C.U. 20
Secuencia normal	P1 - Pulsar en el botón “DELETE”.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede borrar una opinión desde la URI /admin.

Tabla AIII.23.- Casos de uso 23: Ver terms

C.U. 23	*Ver terms*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver los términos y condiciones de la página.
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsar en el enlace “Terms” ubicado en el Footer.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.24.- Casos de uso 24: Ver chat

C.U. 24	*Ver chat*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver el chat interactivo de la página.
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsar en el enlace “Party Room” ubicado en el Navbar.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.25.- Casos de uso 25: Escribir en chat

C.U. 25	*Escribir en chat*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede escribir en el chat interactivo de la página.
Dependencias	*C.U. 24*
Precondición	
Secuencia normal	P1 - Escribir en el text box. P2 - Pulsar intro o el botón con forma de flecha.
Poscondición	
Excepciones	Si el actor ha sido baneado, no podrá escribir.
Comentarios	

Tabla AIII.26.- Casos de uso 26: Buscar juegos

C.U. 26	*Buscar juegos*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede buscar juegos.
Dependencias	
Precondición	
Secuencia normal	P1 - Escribir en el search bar ubicado en el Navbar. P2 - Pulsar intro o en el icono con la lupa.
Poscondición	Se mostrarán los juegos que coinciden con los parámetros de búsqueda establecidos.
Excepciones	Si el actor ha introducido un texto que no coincide con ningún artículo o juego, no se verá ningún juego.
Comentarios	

Tabla AIII.27.- Casos de uso 27: Ver FAQ

C.U. 27	*Ver FAQ*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver el FAQ de la página.
Dependencias	
Precondición	
Secuencia normal	P1 - Pulsar en el enlace “FAQ” ubicado en el navbar o en el enlace “FAQ” ubicado en el Footer.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.28.- Casos de uso 28: Ver comentarios

C.U. 28	*Ver comentarios*
Actores	Invitado, Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver los comentarios en un post.
Dependencias	*C.U. 6*
Precondición	El actor debe abrir el post para ver sus comentarios.
Secuencia normal	P1 - En el post, scroll hacia abajo.
Poscondición	
Excepciones	Si el post no tiene comentarios, no se visualizará ninguno.
Comentarios	Un administrador o superusuario puede ver los comentarios de un post desde la URI /admin.

Tabla AIII.29.- Casos de uso 29: Comentar post

C.U. 29	*Comentar post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede comentar en un post.
Dependencias	*C.U. 2, C.U. 6*
Precondición	El actor debe abrir el post para ver sus comentarios.
Secuencia normal	P1 - En el post, scroll hacia abajo. P2 - Escribir tu comentario en el text input. P3 - Pulsar en el botón “POST”.
Poscondición	
Excepciones	Si el actor ya ha comentado en el post, no puede volver a comentar.
Comentarios	Un administrador o superusuario puede comentar en un post desde la URI /admin.

Tabla AIII.30.- Casos de uso 30: Editar post

C.U. 30	*Editar post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede editar un post.
Dependencias	*C.U. 2, C.U. 6*
Precondición	El actor debe abrir el post para poder editarlo
Secuencia normal	P1 - Pulsar el botón “UPDATE”. P2 - Editar el contenido. P3- Pulsar el botón “POST”.
Poscondición	
Excepciones	P1 - Sólo el autor del post puede editarlo
Comentarios	Un administrador o superusuario puede editar un post desde la URI /admin.

Tabla AIII.31.- Casos de uso 31: Borrar post

C.U. 31	*Borrar post*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede borrar un post.
Dependencias	*C.U. 2, C.U. 6*
Precondición	El actor debe abrir el post para poder editarlo
Secuencia normal	P1 - Pulsar el botón “DELETE”. P2 - Pulsar en el botón “DELETE”.
Poscondición	
Excepciones	P1 - Sólo el autor del post puede borrarlo. P2 - Si se pulsa el botón “I AM AFRAID, LET ME GO BACK” no se borra el post.
Comentarios	Un administrador o superusuario puede borrar un post desde la URI /admin.

Tabla AIII.32.- Casos de uso 32: Crear cuenta vendedor

C.U. 32	*Crear cuenta vendedor*
Actores	Usuario registrado, Administrador, Superusuario
Descripción	El actor puede crear una cuenta de vendedor rellenando un formulario.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - En el navbar pulsar sobre el avatar. P2 - Clickar en la opción “Sell a game” P3 - Rellena el formulario.
Poscondición	
Excepciones	P2 - Si ya se tiene cuenta vendedor, esta opción nos lleva al C.U. 34
Comentarios	Un administrador o superusuario puede crear una cuenta de vendedor desde la URI /admin.

Tabla AIII.33.- Casos de uso 33: Ver keys

C.U. 33	*Ver keys*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede ver los keys que ha comprado, vende o ha vendido.
Dependencias	*C.U. 2, C.U. 3*
Precondición	El actor debe estar logeado y en su perfil.
Secuencia normal	P1 - Pulsar el botón “KEYS”.
Poscondición	
Excepciones	
Comentarios	Un administrador o superusuario puede ver las keys desde la URI /admin.

Tabla AIII.34.- Casos de uso 34: Vender keys

C.U. 34	*Vender keys*
Actores	Vendedor, Administrador, Superusuario
Descripción	El actor puede crear una key, la cual se pone a la venta, rellenando un formulario.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - En el navbar pulsar sobre el avatar. P2 - En el desplegable elegir la opción “Sell a game”. P3 - Rellenar el formulario.
Poscondición	
Excepciones	
Comentarios	Un admin o superusuario puede crear keys desde la URI /admin.

Tabla AIII.35.- Casos de uso 35: Ver panel admin

C.U. 35	*Ver panel admin*
Actores	Administrador, Superusuario
Descripción	El actor puede ver el panel de admin con la URI /admin
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado.
Secuencia normal	P1 - Escribe manualmente /admin en la URL del navegador o pulsa la opción “Admin Panel” en el navbar.
Poscondición	
Excepciones	
Comentarios	

Tabla AIII.36.- Casos de uso 36: Cerrar sesión

C.U. 36	*Cerrar sesión*
Actores	Usuario registrado, Vendedor, Administrador, Superusuario
Descripción	El actor puede deslogearse de la página.
Dependencias	*C.U. 2*
Precondición	El actor debe estar logeado para poder deslogearse.
Secuencia normal	P1 - En el navbar pulsar sobre el avatar. P2 - Pulsar la opción “Logout”.
Poscondición	
Excepciones	
Comentarios	

Anexo IV
Diseño físico de la base de datos

En las figuras de este anexo se muestra el diseño físico de todas las tablas de la base de datos implementada para la aplicación:

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('account_emailaddress_id_s...
email	2	varchar(254)	254		default	[v]	
verified	3	bool	1			[v]	
primary	4	bool	1			[v]	
user_id	5	int4				[v]	

Figura AIV.1.- Diseño físico de la tabla ‘account_emailaddress’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('account_emailconfirmatio...
created	2	timestam...	35			[v]	
sent	3	timestam...	35			[ ]	
key	4	varchar(64)	64		default	[v]	
email_add...	5	int4				[v]	

Figura AIV.2.- Diseño físico de la tabla 'account_emailconfirmation'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_game_id_seq'::regc...
name	2	varchar(50)	50		default	[v]	
short_desc	3	varchar(500)	500		default	[v]	
long_desc	4	varchar(10...	10,000		default	[v]	
image	5	varchar(100)	100		default	[v]	
banner_img	6	varchar(100)	100		default	[v]	
stock	7	int4				[v]	
release_date	8	timestam...	35			[v]	
new_game	9	bool	1			[v]	
lowest_price	10	int4				[ ]	
languages	11	varchar(200)	200		default	[v]	
system_re...	12	varchar(500)	500		default	[v]	

Figura AIV.3.- Diseño físico de la tabla 'appsite_game'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_game_genres_id_se...
game_id	2	int4				[v]	
genre_id	3	int4				[v]	

Figura AIV.4.- Diseño físico de la tabla 'appsite_game_genres'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_genre_id_seq'::regc...
name	2	varchar(50)	50		default	[v]	

Figura AIV.5.- Diseño físico de la tabla 'appsite_genre'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_key_id_seq'::regcla...
key_value	2	varchar(17)	17		default	[v]	
price	3	int4				[ ]	
region	4	varchar(50)	50		default	[v]	
used	5	bool	1			[v]	
game_id	6	int4				[v]	
owner_id	7	int4				[v]	
seller_id	8	int4				[v]	

Figura AIV.6.- Diseño físico de la tabla 'appsite_key'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_opinion_id_seq'::re...
opinion	2	varchar(10...	1,000		default	[v]	
date_posted	3	timestam...	35			[v]	
client_id	4	int4				[v]	
game_id	5	int4				[v]	

Figura AIV.7.- Diseño físico de la tabla 'appsite_opinion'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_shopping_cart_id_s...
client_id	2	int4				[v]	

Figura AIV.8.- Diseño físico de la tabla 'appsite_shopping_cart'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('appsite_shopping_cart_key...
shopping_...	2	int4				[v]	
key_id	3	int4				[v]	

Figura AIV.9.- Diseño físico de la tabla 'appsite_shopping_cart_keys'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_group_id_seq'::regcla...
name	2	varchar(150)	150		default	[v]	

Figura AIV.10.- Diseño físico de la tabla 'auth_group'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_group_permissions_id...
group_id	2	int4				[v]	
permissio...	3	int4				[v]	

Figura AIV.11.- Diseño físico de la tabla 'auth_group_permissions'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_permission_id_seq'::re...
name	2	varchar(255)	255		default	[v]	
content_ty...	3	int4				[v]	
codename	4	varchar(100)	100		default	[v]	

Figura AIV.12.- Diseño físico de la tabla 'auth_permission'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_user_id_seq'::regclass)
password	2	varchar(128)	128		default	[v]	
last_login	3	timestam...	35			[ ]	
is_superuser	4	bool	1			[v]	
username	5	varchar(150)	150		default	[v]	
first_name	6	varchar(30)	30		default	[v]	
last_name	7	varchar(150)	150		default	[v]	
email	8	varchar(254)	254		default	[v]	
is_staff	9	bool	1			[v]	
is_active	10	bool	1			[v]	
date_joined	11	timestam...	35			[v]	

Figura AIV.13.- Diseño físico de la tabla ‘auth_user’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_user_groups_id_seq'::r...
user_id	2	int4				[v]	
group_id	3	int4				[v]	

Figura AIV.14.- Diseño físico de la tabla ‘auth_user_groups’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('auth_user_user_permission...
user_id	2	int4				[v]	
permissio...	3	int4				[v]	

Figura AIV.15.- Diseño físico de la tabla ‘auth_user_ser_permissions’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('blog_comment_id_seq'::re...
content	2	text			default	[v]	
date_posted	3	timestam...	35			[v]	
author_id	4	int4				[v]	
post_id	5	int4				[v]	

Figura AIV.16.- Diseño físico de la tabla ‘blog_comment’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('blog_post_id_seq'::regclass)
title	2	varchar(100)	100		default	[v]	
content	3	text			default	[v]	
date_posted	4	timestam...	35			[v]	
author_id	5	int4				[v]	
genre	6	varchar(50)	50		default	[v]	
img_prepost	7	varchar(100)	100		default	[v]	

Figura AIV.17.- Diseño físico de la tabla ‘blog_post’

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('daguerre_adjustedimage_i...
storage_pa...	2	varchar(200)	200		default	[v]	
adjusted	3	varchar(45)	45		default	[v]	
requested	4	varchar(100)	100		default	[v]	

Figura AIV.18.- Diseño físico de la tabla 'daguerre_adjustedimage'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('daguerre_area_id_seq'::regc...
storage_pa...	2	varchar(300)	300		default	[v]	
x1	3	int4				[v]	
y1	4	int4				[v]	
x2	5	int4				[v]	
y2	6	int4				[v]	
name	7	varchar(20)	20		default	[v]	
priority	8	int4				[v]	

Figura AIV.19.- Diseño físico de la tabla 'daguerre_area'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('django_admin_log_id_seq':...
action_time	2	timestam...	35			[v]	
object_id	3	text			default	[ ]	
object_repr	4	varchar(200)	200		default	[v]	
action_flag	5	int2				[v]	
change_m...	6	text			default	[v]	
content_ty...	7	int4				[ ]	
user_id	8	int4				[v]	

Figura AIV.20.- Diseño físico de la tabla 'django_admin_log'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('django_content_type_id_se...
app_label	3	varchar(100)	100		default	[v]	
model	4	varchar(100)	100		default	[v]	

Figura AIV.21.- Diseño físico de la tabla 'django_content_type'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('django_migrations_id_seq':...
app	2	varchar(255)	255		default	[v]	
name	3	varchar(255)	255		default	[v]	
applied	4	timestam...	35			[v]	

Figura AIV.22.- Diseño físico de la tabla 'django_migrations'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
session_key	1	varchar(40)	40		default	[v]	
session_data	2	text			default	[v]	
expire_date	3	timestam...	35			[v]	

Figura AIV.23.- Diseño físico de la tabla 'django_session'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('django_site_id_seq'::regclass)
domain	2	varchar(100)	100		default	[v]	
name	3	varchar(50)	50		default	[v]	

Figura AIV.24.- Diseño físico de la tabla 'django_site'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('socialaccount_socialaccou...
provider	2	varchar(30)	30		default	[v]	
uid	3	varchar(191)	191		default	[v]	
last_login	4	timestam...	35			[v]	
date_joined	5	timestam...	35			[v]	
extra_data	6	text			default	[v]	
user_id	7	int4				[v]	

Figura AIV.25.- Diseño físico de la tabla 'socialaccount_socialaccount'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('socialaccount_socialapp_id...
provider	2	varchar(30)	30		default	[v]	
name	3	varchar(40)	40		default	[v]	
client_id	4	varchar(191)	191		default	[v]	
secret	5	varchar(191)	191		default	[v]	
key	6	varchar(191)	191		default	[v]	

Figura AIV.26.- Diseño físico de la tabla 'socialaccount_socialapp'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('socialaccount_socialapp_si...
socialapp_id	2	int4				[v]	
site_id	3	int4				[v]	

Figura AIV.27.- Diseño físico de la tabla 'socialaccount_socialapp_sites'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('socialaccount_socialtoken_...
token	2	text			default	[v]	
token_secret	3	text			default	[v]	
expires_at	4	timestam...	35			[ ]	
account_id	5	int4				[v]	
app_id	6	int4				[v]	

Figura AIV.28.- Diseño físico de la tabla 'socialaccount_socialtoken'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('users_client_id_seq'::regcla...
user_id	2	int4				[v]	
money	3	int4				[v]	

Figura AIV.29.- Diseño físico de la tabla 'users_client'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('users_employee_id_seq'::re...
department	2	varchar(100)	100		default	[v]	
hours	3	numeric(2...				[v]	
plus	4	int4				[v]	
user_id	5	int4				[v]	

Figura AIV.30.- Diseño físico de la tabla 'users_employee'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('users_profile_id_seq'::regcl...
image	2	varchar(100)	100		default	[v]	
avatar	3	varchar(100)	100		default	[v]	
user_id	4	int4				[v]	

Figura AIV.31.- Diseño físico de la tabla 'users_profile'

Column Name	#	Data type	Length	Identity	Collation	Not Null	Default
id	1	serial				[v]	nextval('users_seller_id_seq'::regclass)
birth_date	2	date	13			[ ]	
city	3	varchar(100)	100		default	[v]	
country	4	varchar(2)	2		default	[v]	
zip	5	varchar(5)	5		default	[v]	
street_and...	6	varchar(200)	200		default	[v]	
phone_nu...	7	varchar(17)	17		default	[v]	
authentica...	8	varchar(100)	100		default	[v]	
valid	9	bool	1			[v]	
rating	10	int4				[v]	
client_id	11	int4				[v]	

Figura AIV.32.- Diseño físico de la tabla 'users_seller'

El diagrama relacional está dividido en 2 Figuras, la Figura AIV.1 y AIV.2, en la AIV.1 se muestran las tablas que han sido creadas desde los models.py, ya que el resto son de Django o de aplicaciones de terceros, en la Figura AIV.2 se muestran el resto de tablas.

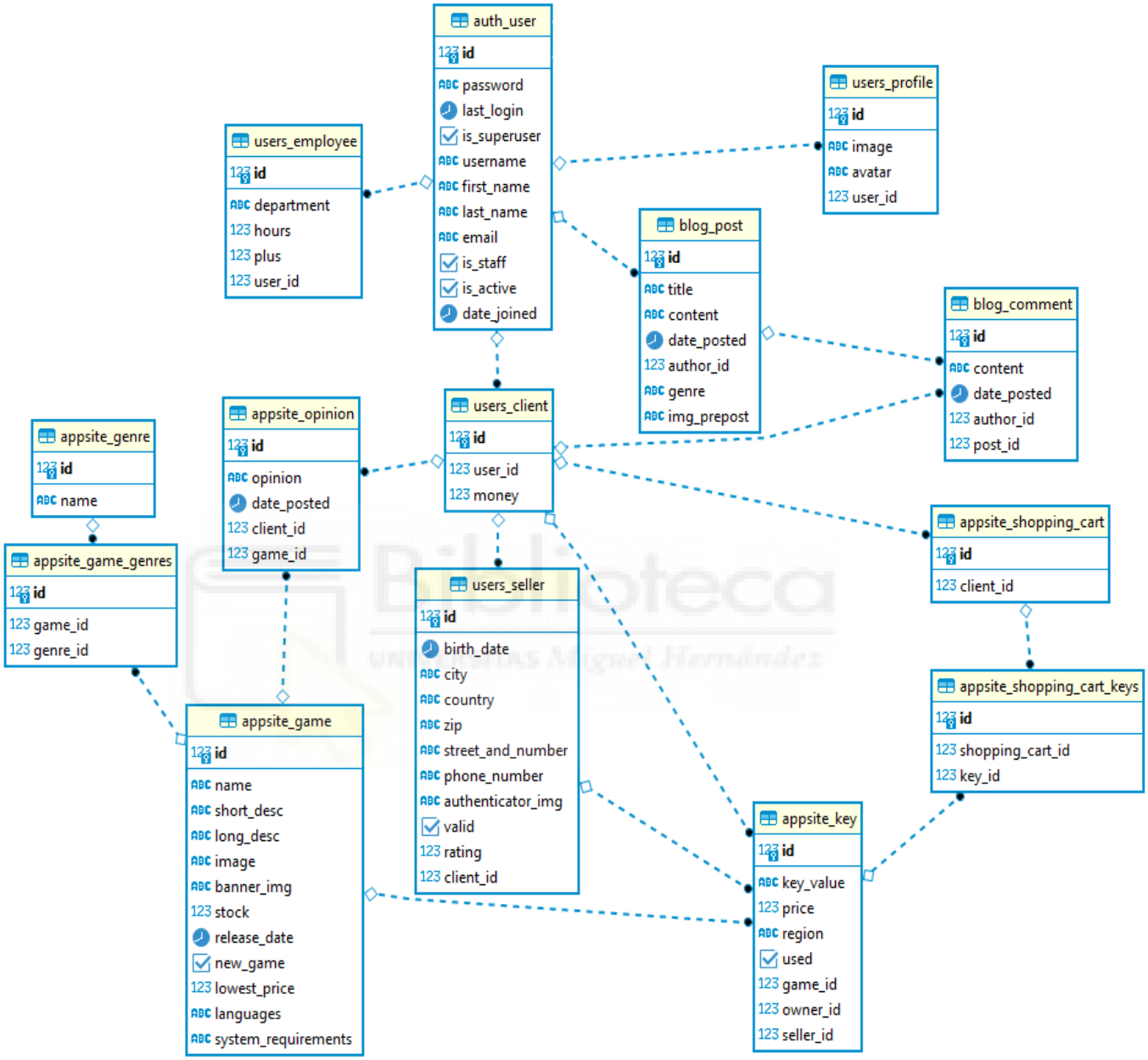

Figura AIV.33 - Diagrama relacional parcial de la base de datos del proyecto, tablas de los models.py

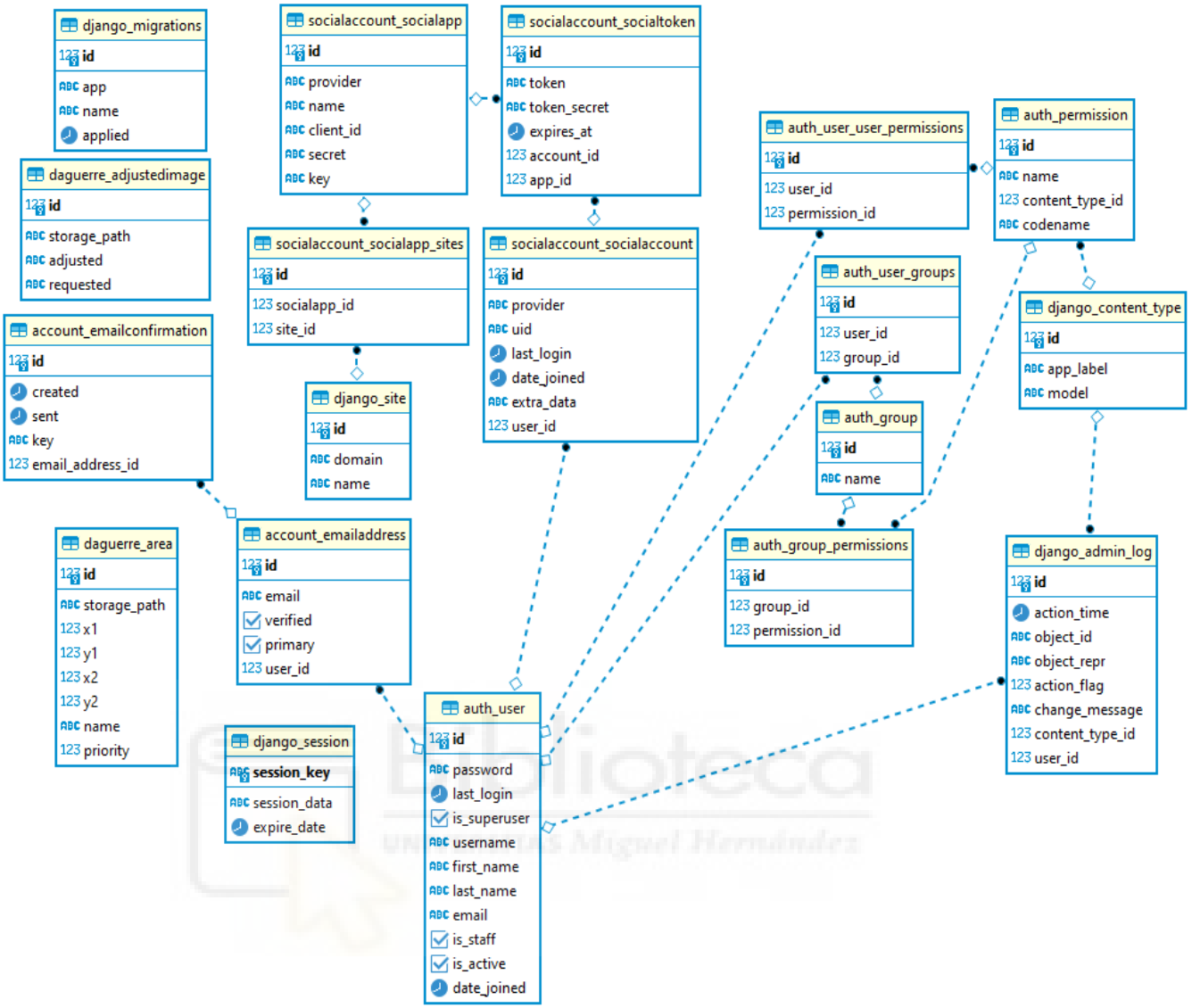

Figura AIV.34 - Diagrama relacional parcial de la Base de datos, tablas de terceros.