



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de Grado

Ingeniería Informática - Ingeniería del Software

My Crossing

**Realizado por
Moisés Pantión Loza y Adrián Gracia Barroso**

**Dirigido por
José Ramón Portillo Fernández**

**Departamento
Matemática Aplicada I**

Sevilla, Junio de 2021

Resumen

En Marzo del pasado año 2020, salió a la venta la última entrega de la saga de videojuegos “Animal Crossing”, publicado por Nintendo. Debido a la pandemia del COVID-19 que coincidió con la salida del videojuego, muchas personas fueron obligadas a quedarse en casa realizando cuarentena, y gracias a los servidores online que permiten la conexión entre jugadores, se creó una comunidad bastante amplia alrededor del videojuego que se vio potenciada debido a dicha pandemia.

De esta forma, no pasó mucho tiempo hasta que empezaron a salir varias páginas web y aplicaciones cuyo objetivo era informar y conectar a los jugadores que disfrutaban del juego. Del mismo modo, este trabajo al que hemos bautizado bajo el título de “My Crossing”, no es sino un sistema de información que, como muchos otros, sirve de ayuda para el jugador proporcionándole herramientas e información sobre el juego.

No obstante, buscábamos proporcionarle al usuario una aplicación lo más completa posible y que le ayudase tanto en el día a día como en tareas más esporádicas, ya que el trabajo de buscar varias herramientas online para distintos aspectos del juego puede llegar a ser bastante tedioso, por lo que como usuarios y parte de la comunidad que somos, decidimos realizar un compendio de aquellas herramientas más importantes, así como implementar funcionalidades e ideas que hasta ahora no se habían visto en los demás sitios web y que son de gran utilidad.

La aplicación web está desarrollada principalmente con PHP y Angular, ya que además de realizar el sistema también queríamos ampliar nuestro conocimiento sobre esta última tecnología ya que es bastante útil para el desarrollo web y está actualmente en auge.

Agradecimientos

Gran parte de este proyecto no hubiese sido posible sin herramientas y artefactos generados por otras personas que, como nosotros, también comparten la afición por este juego. A continuación se cita a aquellos usuarios que nos han proporcionado dichos artefactos.

- Purplepixel: Iconos de Kamilo y Gullivarr
- Mattlewis92: Angular Calendar
- Norviah: Animal Crossing NPM Database
- Alexis8717: Hoja de cálculo de Animal Crossing: New Horizons
- MarkNickerson: Iconos para el logo de la aplicación
- Equipo de Turnip Prophet: predicción del precio de los nabos
- Ninji: Información sobre el funcionamiento del mercado de nabos, así como de la mudanza de los vecinos

Agradecer también a la increíble comunidad de este juego que, incluso después de más de un año desde su lanzamiento, sigue investigando y confeccionando herramientas y aplicaciones para disfrutar aun más de este videojuego.

Por último pero no por ello menos importante, el proyecto no habría sido posible sin la gran colaboración de nuestro tutor José Ramón Portillo Fernández, el cual nos ha animado en todo momento a elaborar un buen proyecto y nos ha aconsejado ante las dudas que han ido surgiendo.

Índice general

Índice general	III
Índice de figuras	V
1 Introducción	1
2 Definición de objetivos	2
3 Análisis de antecedentes y aportación realizada	4
3.1 Análisis de antecedentes	4
3.1.1 Servicios de cálculo de nabos	4
3.1.2 Servicios de cálculo de probabilidades sobre mudanzas	5
3.1.3 Nook's Island	5
3.2 Aportación realizada	6
4 Análisis temporal y costes de desarrollo	8
4.1 Análisis temporal	8
4.1.1 Iteraciones	8
4.1.2 Estimación vs Realidad	9
4.2 Costes de desarrollo	10
4.2.1 Costes Directos	10
4.2.2 Costes Indirectos	10
4.2.3 Amortizaciones	10
4.2.4 Total	11
5 Análisis de requisitos y diseño	12
5.1 Participantes del proyecto	12
5.2 Requisitos	13
5.2.1 Requisitos de Información	13
5.2.2 Requisitos Funcionales	15
5.2.3 Requisitos no Funcionales	18
5.3 Diagrama de Clases	19
5.4 Diseño	23
6 Implementación	31
6.1 Entorno de desarrollo	31
6.2 Tecnologías utilizadas	32
6.2.1 Angular	33
6.2.2 XAMPP y Apache	34
6.2.3 PHP	35
6.2.4 MariaDB y MySQL	36

6.2.5	Git	37
6.3	Dificultades encontradas y soluciones propuestas	38
6.3.1	It. 3 - Perfil	38
6.3.2	It. 3 - Mantener sesiones	38
6.3.3	It. 3 - Features del perfil	39
6.3.4	It. 4 - API incompleta	39
6.3.5	It. 5 - Código reciclado	40
6.3.6	It. 5 - Feature demasiado compleja	41
6.3.7	It. 6 - Integración continua	41
6.3.8	It. 6 - Despliegue	42
6.3.9	It. 6 - CORS Policy	43
6.3.10	It. 6 - AdBlock	43
7	Pruebas	44
8	Comparación con otras alternativas	67
9	Manual de Usuario	69
	Conclusiones y desarrollo futuro	93

Índice de figuras

3.1 Turnip Prophet	4
3.2 Mudanza	5
3.3 Nook's Island	6
5.1 Inicio 1	23
5.2 Inicio 2	23
5.3 Listado genérico	24
5.4 Eventos	25
5.5 Eventos - Detallado	25
5.6 Registro	26
5.7 Perfil	27
5.8 Cazar ahora 1	28
5.9 Cazar ahora 2	28
5.10 Calculadora de Nabos	29
5.11 Probador	30
6.1 Logotipo Visual Studio Code	31
6.2 Logotipo Angular	33
6.3 Logotipo XAMPP	34
6.4 Logotipo Servidor Web Apache	34
6.5 Logotipo PHP	35
6.6 Logotipo MariaDatabase	36
6.7 Logotipo MySQL	36
6.8 Logotipo Git	37
8.1 Comparación alternativas	67
9.1 Inicio	69
9.2 Registro	70
9.3 Inicio con sesión iniciada	70
9.4 Catálogo peces	71
9.5 Peces obtenidos	71
9.6 Info hemisferio norte	72
9.7 Info hemisferio sur	72
9.8 Paginación	72
9.9 Catálogo obras de arte	73
9.10 Falsificación	73
9.11 Variaciones	74
9.12 Variación	74
9.13 Catálogo canciones	74
9.14 Barra de navegación	75

9.15 Cazar ahora	75
9.16 Filtros	75
9.17 Mercado nabos	76
9.18 Datos	76
9.19 Tabla predicción	77
9.20 Predicción precisa	77
9.21 Información mercado	78
9.22 Información patrones	78
9.23 Calendario	79
9.24 Cumpleaños	79
9.25 Evento	80
9.26 Halloween	80
9.27 Desplegable perfil	81
9.28 Perfil	81
9.29 Editar perfil	81
9.30 Perfil actualizado	82
9.31 Modo edición	82
9.32 Nuevas tareas	83
9.33 Menú tareas	83
9.34 Tareas editadas	83
9.35 Tareas realizada	83
9.36 Visitantes semanales	84
9.37 Semana rellanda	84
9.38 Menú visitantes	85
9.39 Confirmación guardado	85
9.40 Mis vecinos	86
9.41 Información vecino	86
9.42 Menú vecinos	87
9.43 Menú amistad	87
9.44 Información amistad	87
9.45 Mudanza	88
9.46 Cálculo porcentaje	88
9.47 Colecciones especiales	89
9.48 Check obtenido	89
9.49 Álbum fotos	90
9.50 Añadir imagen	90
9.51 Álbum con fotos	90
9.52 Borrar imagen	90
9.53 Catálogo sueños	91
9.54 Sueño	91
9.55 Catálogo filtrado	91
9.56 Crear sueño	92
9.57 Confirmación borrado	92

CAPÍTULO 1

Introducción

En Internet podemos encontrar gran cantidad de herramientas para calcular diferentes aspectos y datos del videojuego “*Animal Crossing: New Horizons*”. El problema recae en que muchas herramientas que podrían ser bastante útiles aún no han sido desarrolladas, así como la falta de un servicio web que recoja un compendio de las más importantes e interesantes. Esto hace perder bastante tiempo a todos aquellos jugadores que deseen consultar ciertos datos, ya que tendrían que navegar por distintos sitios, en vez de disponer de todo lo que necesitan en un mismo lugar.

Debido a esto, llegamos a la decisión de crear una aplicación web que contenga una colección de herramientas, así como de desarrollar algunas nuevas por nuestra cuenta, todo esto con una interfaz simple que permita acceder a cada una sin ninguna dificultad, y además intentando quedarnos dentro de la estética ya propuesta por el juego.

En nuestra aplicación, el usuario tiene a su alcance una serie de herramientas pensadas para los distintos ámbitos del juego: tareas personalizables diarias, un calendario para no perderse los posibles eventos del juego, una forma de predecir el mercado, un álbum de fotos personal, varios catálogos que cubren todo lo que el jugador puede encontrar en el juego, así como una forma para catalogar los ítems y disponer de una colección personal, etc.

Las herramientas y posibilidades son tan extensas que no podemos agruparlas todas, pero sí que podemos realizar un conjunto de las más importantes y de aquellas que le serán de más utilidad al usuario, y eso es lo que hemos hecho. Pensamos que es una aplicación muy útil, bastante ampliable de cara al futuro (ya que el juego sigue actualizándose con nuevo contenido), y que puede ser de gran ayuda para muchas personas que a día de hoy sigan disfrutando del juego.

CAPÍTULO 2

Definición de objetivos

Para que podamos dar este proyecto por finalizado, debemos cumplir una serie de objetivos:

- **Objetivos académicos:**

- Centralizar varias de las distintas herramientas que ya existen en una sola aplicación web, de forma que el usuario disponga de toda la información en un mismo lugar.
- Crear como mínimo una herramienta nueva que aporte una funcionalidad útil de cara al videojuego.
- Permitir el registro de usuarios y el almacenaje de datos del mismo
- Evitar la reiterada introducción de datos de forma manual. Algunas herramientas necesitarán ciertos datos del jugador para realizar los cálculos, y dado que es una pesada tarea el introducirlos uno a uno varias veces, queremos intentar automatizarlo dentro de lo posible para que el usuario tenga que introducir los datos el mínimo numero de veces.

- **Objetivos personales:**

- Realizar el apartado de front-end de este proyecto con la tecnología Angular, no solo debido a su utilidad para el desarrollo web de una sola página, sino también por su popularidad en el sector laboral, ya que pensamos que puede sernos útil de cara al futuro. Esta va a ser la primera vez que tratemos con Angular, por lo que pensamos que aprender a usar una tecnología desde cero y aplicarla de forma que se obtenga el mejor resultado posible es un gran reto que nos gustaría afrontar.
- Usar Spring Boot para el back-end, ya que como íbamos a empezar desde cero con Angular, pensamos que sería mejor usar algo que ya conocemos y que se nos ha estado enseñando durante gran parte de la carrera. Dicho esto, va a ser la primera vez que empecemos un proyecto desde cero sin contar con una guía, por lo que aunque conozcamos el lenguaje y la tecnología, es una buena oportunidad para poner en practica lo que hemos aprendido y demostrar que hemos asimilado los conocimientos.*

*A medida que se ha seguido el desarrollo de la aplicación, dado que no conocíamos Angular ni el alcance que tiene, pensábamos que solo se encargaría de la parte visual, pero tras haber aprendido y desarrollado algunas features, hemos visto que no solo se limita a lo visual, sino que también se encarga de gran parte de la lógica que hay por detrás.

Debido a esto, llegamos a un punto en que lo único que necesitábamos era realizar una conexión a la base de datos desde Angular, y probando configuraciones hemos descubierto que usando XAMPP (MariaDB y Apache) se podía acceder a la base de datos mediante un archivo PHP que realizase la conexión y las peticiones oportunas, de forma que angular recogiera los datos mediante una petición HTTP (usando el back-end a modo de API).

Es por esto que hemos optado por usar esta configuración en vez de usar Spring ya que, aunque es cierto que conocíamos el framework y el lenguaje, no nos resultó necesario usarlo ya que con XAMPP disponemos de todo lo que necesitábamos y conseguimos el objetivo principal que es la integración de los componentes de forma funcional para poder desarrollar sobre dicha configuración.

Además, aunque hemos visto algo de PHP durante el curso, no ha sido mucho, por lo que se nos presenta otra oportunidad para ampliar nuestras capacidades y conocimientos investigando más sobre un lenguaje bastante usado actualmente, y aumentar nuestras habilidades como desarrolladores.

CAPÍTULO 3

Análisis de antecedentes y aportación realizada

3.1– Análisis de antecedentes

A continuación se mostrarán algunas de las herramientas a las que uno debería de acceder para obtener gran parte de la información que pudiera necesitar sobre el juego:

3.1.1. Servicios de cálculo de nabos

En el videojuego *Animal Crossing: New Horizons* hay implementado un mercado similar al de las acciones, pero con nabos. El precio de compra y el de venta, aunque limitados por un rango, varía cada semana. Esta funcionalidad es usada por los jugadores para amasar grandes fortunas comprando cientos de nabos y después, haciendo uso de servicios de cálculo y predicción de patrones de venta, venderlos cuando se pueda sacar el mayor beneficio.

Muchos servicios se centran exclusivamente en esta función del cálculo de futuros patrones de venta, y además precisan de bastante información, al igual que en nuestro servicio, para poder realizar los cálculos necesarios para dar una aproximación cercana a la realidad.



Figura 3.1: Turnip Prophet

Este tipo de servicio lo proveen varias páginas web, como Turnip Prophet (véase la figura 3.1), siendo esta la página más conocida y con una interfaz más cercana al usuario, siguiendo un diseño similar a la estética de *Animal Crossing: New Horizons*; o Turnip Price Calculator, la cual es una web un poco más técnica que ofrece varias gráficas en la que obtener información adicional sobre las probabilidades de los patrones de venta.

Todos estos servicios web, incluido el nuestro, han podido hacer uso de dicha herramienta gracias al usuario *Ninji*, el cual analizó el código del videojuego para desarrollar un algoritmo que pudiera obtener la aproximación de los patrones de venta de la próxima semana.

3.1.2. Servicios de cálculo de probabilidades sobre mudanzas

En este videojuego existe la posibilidad de que tus vecinos decidan irse de tu isla. Esta funcionalidad, al igual que la de los nabos, sigue un algoritmo que se puede usar para calcular la probabilidad de que alguien se mude en un día en concreto, y si se da el caso, averiguar cuál es el vecino que tiene más probabilidades de mudarse.

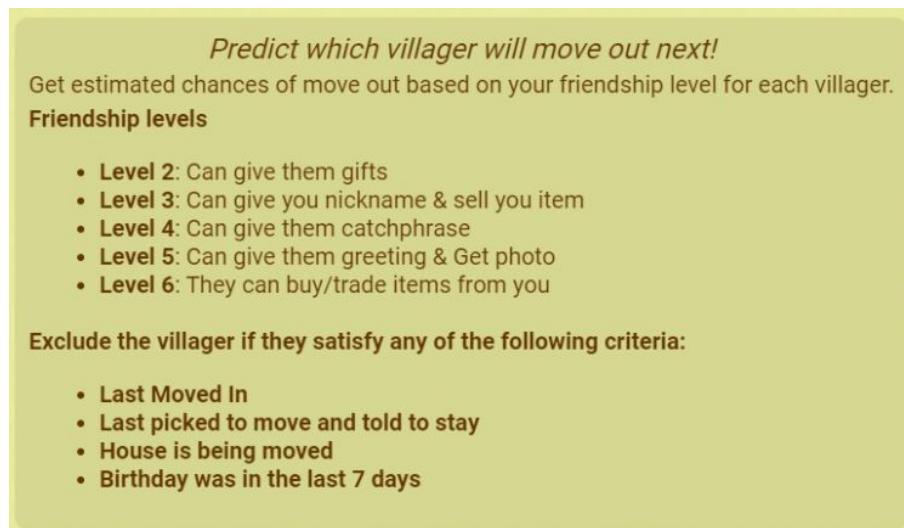


Figura 3.2: Mudanza

Para realizar esos cálculos existen varias páginas web (véase la figura 3.2), como puede ser NookPlaza, la cual no solo ofrece la funcionalidad de calcular las mudanzas de vecinos, sino que además ofrece algunas pequeñas funcionalidades más. Otro ejemplo de este servicio sería Yue's Move-Out Calculator, donde además se puede calcular la probabilidad de mudanza en caso de que más de un jugador viva en la misma isla, añadiendo datos sobre cada jugador para realizar una media y obtener una predicción fiable.

Al igual que con el algoritmo de los nabos, este servicio también ha sido desarrollado por el usuario *Ninji*.

3.1.3. Nook's Island

En esta página podemos encontrar varios servicios a destacar. Para empezar, dispone de un mercado donde cualquier jugador puede poner a la venta artículos que no necesite y desee intercambiar por bienes dentro del juego. Asimismo, también pueden comprar productos que se encuentren anunciados. Todo el servicio del mercado es informativo, ya que la página no lleva un sistema de venta como tal, sino que solo informa a los usuarios y los pone en contacto para que sean ellos los que se reúnan de forma online en el videojuego y realicen la transacción. Gracias a este servicio, se puede contar con una gran interacción de la comunidad mediante la página web, lo que le da mucha vida.



Figura 3.3: Nook's Island

Esta es una funcionalidad que hemos decidido no añadir ya que hemos considerado que con el conjunto de herramientas que vamos a centralizar, estábamos realizando bastante trabajo y añadir una funcionalidad tan extensa como esta puede resultar demasiado ambicioso, ya que aunque siendo meramente informativo, habría que realizar algún tipo de conexión entre usuarios de forma que se pudiesen comunicar. Además, el objetivo de nuestro sistema era centralizar gran parte de las herramientas que un usuario pueda necesitar a la hora de jugar de forma individual, como si de una enciclopedia con funcionalidades extra se tratase, por lo que no se busca la interacción entre usuarios. Sin embargo, no se descarta completamente en caso de que haya que aumentar el alcance del proyecto.

Entre otras funcionalidades, Nook's Island cuenta también con un sistema para compartir Códigos de Sueño y Diseños personalizados, así como una enciclopedia de criaturas. Sin embargo, creemos que tanto el apartado visual como la accesibilidad de la web es bastante mejorable. Estas funciones estarán en nuestro sistema de una forma algo distinta, cambiando la información que se muestra de forma que lo más importante se encuentre a primera vista y los detalles menos importantes se queden en segundo plano. De esta forma se espera que el usuario pueda hacer uso de ellas de una forma más fácil e intuitiva.

3.2– Aportación realizada

Los servicios mostrados anteriormente no son los únicos que se pueden encontrar en Internet, hay una extensa cantidad de herramientas disponibles (KevinPayravi, 2020a) pero la mayoría se encuentran en sitios web distintos y algunas pueden llegar a ser algo confusas y poco intuitivas para el usuario.

Para hacerle más fácil la búsqueda de información al usuario, nuestro proyecto centraliza varias de las herramientas existentes en un mismo sitio web, buscando obtener una interfaz sencilla y simple que pueda ser apta para todos los públicos, incluidos los más pequeños, ya que no hay que olvidar que este videojuego abarca un público bastante extenso.

La centralización de las herramientas resulta en un ahorro de tiempo notable para los usuarios que busquen información de forma frecuente, además de dar a conocer aspectos del juego y funciones que puede que no conozcan del todo. De esta forma buscamos que les pueda ser útil para obtener ciertos objetos o hitos en el juego y que puedan disponer de todo lo que necesiten en un mismo sitio, sin tener que registrarse en varias aplicaciones de forma innecesaria.

Nuestra aplicación, además de seguir dentro de lo posible dentro de la estética general del juego (cosa que no suelen cumplir el resto de aplicaciones), ofrece herramientas básicas como listados de criaturas y objetos, así como la posibilidad de añadir algunos a la colección personal para así poder llevar un seguimiento.

Pero no solo nos hemos limitado a eso, sino que además ofrece una serie de herramientas no tan comunes y que son de gran utilidad para el usuario, como pueden ser:

- La calculadora de nabos: herramienta que sirve para calcular y predecir los precios de compra y venta de los nabos de manera semanal, lo cual le da al usuario una increíble ventaja a la hora de usar el mercado de nabos, permitiéndole amasar una fortuna dentro del juego en bastante poco tiempo. Esta herramienta, aunque ya existe en otras aplicaciones, suele estar implementada de forma aislada, por lo que hace que el usuario necesite una aplicación para una sola herramienta, siendo esto uno de los puntos que queremos solucionar con nuestra aplicación.
- Tareas diarias y visitantes semanales: estas dos herramientas, aun siendo de las más útiles ya que permiten llevar un seguimiento diario y semanal (por lo que se les da mucho uso), suelen aparecer casi de manera exclusiva en aplicaciones móvil, por lo que la implementación en una aplicación web disponible en ordenadores es una novedad que permite que los usuarios no tengan que limitarse al uso de una aplicación en el móvil, y además puedan acceder de manera rápida y sencilla desde cualquier lugar siempre que dispongan de un dispositivo inteligente y conexión a Internet.
- Listado de criaturas en función de la fecha y hora: otra de las herramientas más útiles y que suele aparecer mas frecuentemente en aplicaciones móviles. Esta herramienta es un listado de los tres tipos de criaturas que se pueden capturar en el juego, pero lo realmente importante es que, dado que en el juego las criaturas aparecen dependiendo del mes o la hora del día, el listado muestra al igual que en el juego aquellas criaturas que se pueden capturar en tiempo real, y además permite cambiar la fecha para que el usuario pueda planear con antelación la llegada de un nuevo mes (y por tanto, nuevas criaturas), siendo esto una fuente de información realmente útil para el jugador.
- Calcular la probabilidad de mudanza: uno de los aspectos más importantes del juego son los vecinos con los que te puedes relacionar, y periódicamente estos vecinos irán pensando en mudarse de la isla del jugador, lo que puede ser una oportunidad para buscar nuevos vecinos. Este proceso es bastante tedioso para un jugador que no conozca mucha información al respecto, ya que conlleva esperar varios días (incluso semanas) si no se conocen todos los aspectos a tener en cuenta. Esta herramienta permite calcular la probabilidad de mudanza de cada vecino, de forma que el jugador pueda conocer con la mayor precisión posible cuál será el siguiente vecino en mudarse y actuar en consecuencia. Es una herramienta sumamente importante y de la que muy pocas aplicaciones disponen, por lo que es un punto bastante importante en nuestra aplicación.
- Álbum de fotos y reproductor de música: estas dos herramientas son completamente innovadoras, ya que no hemos encontrado aplicaciones que ya las tuvieran. Se trata tanto de un álbum de fotos personal para que el usuario pueda reunir sus capturas del juego en un mismo sitio, así como un reproductor de canciones del juego junto al listado de las mismas. El listado de canciones es algo de lo que si disponen varias aplicaciones, pero ninguna permite reproducir dichas canciones por lo que es un punto a favor para nuestra aplicación.

CAPÍTULO 4

Análisis temporal y costes de desarrollo

4.1– Análisis temporal

4.1.1. Iteraciones

Dividiremos el trabajo en varias iteraciones, de forma que el proyecto se quede lo más planificado posible para que, a la hora de trabajar, solo haya que seguir las iteraciones:

Iteración 1

En esta iteración buscamos dejar empezado tanto el proyecto como la documentación, creando una base sobre la que trabajar más adelante. Además dejamos preparadas las herramientas a utilizar, tanto de trabajo (Microsoft Visual Studio Code, XAMPP), como de redacción (TeXStudio, Google Docs), almacenamiento en la nube (GitHub, Google Drive) y gestión del tiempo (Toggl). Realizamos el primer capítulo de la documentación y diseñamos los mockups.

Iteración 2

Planificamos el proyecto de forma que se quede dividido para las siguientes iteraciones y continuamos con la documentación. Redactamos los apartados que podemos (ya que algunos de ellos requieren que se haya avanzado más) de los capítulos 2, 3, 4 y 5. Además se realiza un curso de Angular (LeParadoxHD, 2019) para prepararnos para la siguiente iteración.

Iteración 3

Esta iteración está pensada como introducción a Angular. Se planea tener lista la página de inicio, el registro de usuario y login, el acceso a perfil y su respectiva información sobre el usuario, así como las diversas acciones que puede realizar desde el mismo (a excepción del álbum de fotos). Se busca establecer la estructura de la base de datos, realizar la vista (final a ser posible) de las páginas mencionadas así como los elementos back-end necesarios para permitir a los usuarios registrarse en la plataforma, loguearse y acceder a su perfil. Además se realizarán las pruebas respectivas.

Tras haber realizado la iteración, se descubrió que había cierta diferencia entre lo que pensábamos que sería el perfil y lo que ha acabado siendo (Más información en el capítulo 6).

Iteración 4

Se busca implementar la API y los elementos relacionados con esta. Habría que hacer un repaso de lo hecho en la iteración anterior para aplicar las imágenes y valores que se necesiten de la API. Además se planea realizar los distintos catálogos (a excepción del de sueños y canciones) así como el listado de criaturas en tiempo real. Se realizarán las pruebas respectivas.

Iteración 5

En esta iteración nos queremos centrar en la implementación de las funciones de calculadora de nabos, calendario de eventos, álbum de fotos, catálogo de sueño, catálogo de canciones y vestuario, así como las pruebas respectivas.

Una vez realizada la iteración, tuvimos que afrontar algunas decisiones que han tenido efecto en lo que finalmente se ha acabado desarrollando (Más información en el capítulo 6).

Iteración 6

En la última iteración queremos terminar la redacción de la documentación y revisarla de forma que se quede lista para la entrega, arreglar los posibles fallos menores que queden del proyecto, comprobar que la aplicación está terminada y realizar el despliegue.

4.1.2. Estimación vs Realidad

Iteración	Horas estimadas	Horas reales	Diferencia (h)
1	15	12	-3
2	50	56	+6
3	200	210	+10
4	160	148	-12
5	200	153	-47
6	25	55	+30
Total	650	634	-16

Al haber eliminado diseños y probador (ver capítulo 6), se han reducido en parte las horas reales en el sprint 5. Sin embargo, debido a que hemos acabado realizando los tests en el sprint 6, así como a los distintos fallos de despliegue a los que nos hemos enfrentado, hemos tenido que dedicarle más tiempo del que pensábamos a dicho sprint, por lo que hemos contado con tiempo de margen y como se puede observar, no ha supuesto un problema respecto a la planificación.

4.2– Costes de desarrollo

4.2.1. Costes Directos

Personal

Dado que aunque sepamos algo de PHP gracias a algunas asignaturas cursadas anteriormente, no tenemos ninguna experiencia con Angular, calcularemos nuestros sueldos clasificándonos como programadores junior. Dicho conjunto cobra una media de 19.000€ al año, lo cual sería 1.584€ al mes, que al dividirlos en un horario de trabajo de 40 horas semanales, 160 horas al mes, equivaldría a 10€ la hora.

Esto hay que multiplicarlo por dos ya que el equipo del proyecto está formado por dos personas, es decir, dos programadores junior con sus respectivos sueldos, por lo que sería un total de 20€ la hora.

4.2.2. Costes Indirectos

Realizaremos el proyecto en Microsoft Visual Studio Code, un entorno de desarrollo gratuito, en una base de TypeScript con Angular usando el back-end en PHP a modo de API encargado de realizar las peticiones a una base de datos de MariaDatabase (MySQL), cuyas licencias son gratuitas.

Para el almacenamiento de datos en la nube usaremos GitHub y Google Drive, los cuales disponen de licencia gratuita con aspectos limitados.

Para la redacción utilizaremos tanto Google Docs como TeXStudio, ambos con licencia gratuita.

Para la gestión de tiempo y comunicación usaremos Toggl y WhatsApp respectivamente, las cuales también disponen de licencia gratuita (con limitaciones en el caso de Toggl).

Los únicos costes indirectos que podríamos encontrar sería la electricidad y la conexión a Internet, lo cual supone de media 35€ por persona, 70€ en total al mes.

4.2.3. Amortizaciones

Para realizar el proyecto hemos usado los siguientes equipos:

	Equipo 1	Equipo 2
Procesador	Intel Core i-5 4460	Intel Core i-5 4460
Almacenamiento	1 Tb	250 Gb
Memoria	16 GB	8 GB
Gráfica	NVIDIA GeForce GTX 950	NVIDIA GeForce 750 GTX
Precio	800€	650€

Además de esto para comunicarnos hemos usado nuestros teléfonos móviles, por eso hemos decidido añadirlo al documento:

	Móvil 1	Móvil 2
Modelo	Samsung Galaxy J7	Xiaomi MiA4
Precio	200€	130€

4.2.4. Total

Finalmente, sumaremos todos los cálculos anteriores para dar una cifra al precio total:

Concepto	Coste
Directo	10€/h = 6.500€
Indirecto	70€/mes = 560€
Amortizaciones	1.780€*10% = 178€
Total	7.238€

CAPÍTULO 5

Análisis de requisitos y diseño

5.1– Participantes del proyecto

Organización	Departamento de Matemática Aplicada I
Dirección	E.T.S. de Ingeniería Informática, Av. Reina Mercedes s/n, 41012 Sevilla
Teléfono	954 552 766
Email	secretarma1@us.es

Participante	Moisés Pantón Loza
Organización	TFG MyCrossing
Rol	Desarrollador

Participante	Adrián Gracia Barroso
Organización	TFG MyCrossing
Rol	Desarrollador

Participante	José Ramón Portillo Fernández
Organización	Departamento de Matemática Aplicada I
Rol	Tutor

5.2– Requisitos

5.2.1. Requisitos de Información

IRQ-001	Información sobre Insectos
<p>El sistema deberá almacenar y mostrar los siguientes datos sobre los insectos (tanto para hemisferio norte como para hemisferio sur):</p> <ul style="list-style-type: none">• Nombre• Disponibilidad (fecha y hora)• Lugar de obtención• Clima• Imagen• Precio• Obtenido por el usuario	

IRQ-002	Información sobre Peces
<p>El sistema deberá almacenar y mostrar los siguientes datos sobre los peces (tanto para hemisferio norte como para hemisferio sur):</p> <ul style="list-style-type: none">• Nombre• Disponibilidad (fecha y hora)• Lugar de obtención• Sombra• Imagen• Precio• Obtenido por el usuario	

IRQ-003	Información sobre Criaturas de mar
<p>El sistema deberá almacenar y mostrar los siguientes datos sobre las criaturas de mar (tanto para hemisferio norte como para hemisferio sur):</p> <ul style="list-style-type: none">• Nombre• Disponibilidad (fecha y hora)• Sombra• Velocidad• Imagen• Precio• Obtenido por el usuario	

IRQ-004	Información sobre Fósiles
<p>El sistema deberá almacenar y mostrar los siguientes datos sobre los fósiles:</p> <ul style="list-style-type: none">• Nombre• Imagen• Obtenido por el usuario	

IRQ-005	Información sobre Arte
El sistema deberá almacenar y mostrar los siguientes datos sobre las obras de arte:	
<ul style="list-style-type: none">• Nombre• Imagen• Existencia de copia falsa e imagen de la misma• Obtenido por el usuario	
IRQ-006	Información sobre Aldeanos
El sistema deberá almacenar y mostrar los siguientes datos sobre los aldeanos:	
<ul style="list-style-type: none">• Nombre• Imagen• Exterior de su casa• Personalidad• Cumpleaños• Especie• Género• Colores favoritos• Estilos favoritos	
IRQ-007	Información sobre Canciones
El sistema deberá almacenar y mostrar los siguientes datos sobre las canciones:	
<ul style="list-style-type: none">• Nombre• Imagen• Música• A qué humor pertenece• Obtenido por el usuario	
IRQ-008	Información sobre Ropa
El sistema deberá almacenar y mostrar los siguientes datos sobre las prendas de ropa:	
<ul style="list-style-type: none">• Nombre• Tipo de prenda• Imagen• Variaciones (e imágenes de las mismas)	
IRQ-009	Información sobre Muebles
El sistema deberá almacenar y mostrar los siguientes datos sobre los muebles:	
<ul style="list-style-type: none">• Nombre• Tipo de mueble• Imagen• Variaciones (e imágenes de las mismas)	

IRQ-010	Información sobre Eventos
El sistema deberá almacenar y mostrar los siguientes datos sobre los eventos: <ul style="list-style-type: none">• Nombre• Descripción• Fecha	

5.2.2. Requisitos Funcionales

FRQ-001	Registro
El sistema deberá permitir darse de alta en la página.	

FRQ-002	Inicio de sesión
El sistema deberá permitir iniciar sesión en la página.	

FRQ-003	Cierre de sesión
El sistema deberá permitir cerrar la sesión de la página.	

FRQ-004	Catálogo Peces
El sistema deberá permitir el acceso a un catálogo de Peces.	

FRQ-005	Catálogo Insectos
El sistema deberá permitir el acceso a un catálogo de Insectos.	

FRQ-006	Catálogo Criaturas marinas
El sistema deberá permitir el acceso a un catálogo de Criaturas marinas.	

FRQ-007	Catálogo Fósiles
El sistema deberá permitir el acceso a un catálogo de Fósiles.	

FRQ-008	Catálogo Arte
El sistema deberá permitir el acceso a un catálogo de Obras de Arte.	

FRQ-009	Catálogo Muebles
El sistema deberá permitir el acceso a un catálogo de Muebles.	
FRQ-010	Catálogo Ropa
El sistema deberá permitir el acceso a un catálogo de <u>Ropa</u> .	
FRQ-011	Catálogo Vecinos
El sistema deberá permitir el acceso a un catálogo de Vecinos.	
FRQ-012	Calendario de Eventos
El sistema deberá permitir el acceso a un calendario de Eventos.	
FRQ-013	Catálogo Canciones
El sistema deberá permitir el acceso a un catálogo de Canciones.	
FRQ-014	Criaturas en tiempo real
El sistema deberá permitir ver las criaturas disponibles para atrapar en tiempo real.	
FRQ-015	Calculador de nabos
El sistema deberá permitir el cálculo del patrón de venta de nabos, así como predecir el precio de venta.	
FRQ-016	Catálogo de códigos de sueño
El sistema deberá permitir el acceso a un catálogo de códigos de sueño que hayan sido subido por los usuarios.	
FRQ-017	Acceso Perfil
El sistema deberá permitir el acceso a un perfil propio.	
FRQ-018	Edición Perfil
El sistema deberá permitir la edición del perfil propio.	

FRQ-019	Visitantes semanales
El sistema deberá permitir ver y editar los visitantes semanales de la isla del usuario.	
FRQ-020	Tareas
El sistema deberá permitir ver, añadir, editar y borrar tareas diarias a elección del usuario.	
FRQ-021	Álbum
El sistema deberá permitir subir y borrar fotos de la isla del usuario.	
FRQ-022	Avance colección Peces
El sistema deberá permitir ver el avance personal de la colección de Peces, así como los que faltan por obtener.	
FRQ-023	Avance colección Insectos
El sistema deberá permitir ver el avance personal de la colección de Insectos, así como los que faltan por obtener.	
FRQ-024	Avance colección Criaturas marinas
El sistema deberá permitir ver el avance personal de la colección de Criaturas marinas, así como las que faltan por obtener.	
FRQ-025	Avance colección Arte
El sistema deberá permitir ver el avance personal de la colección de Obras de Arte, así como las que faltan por obtener.	
FRQ-026	Avance colección Canciones
El sistema deberá permitir ver el avance personal de la colección de Canciones, así como las que faltan por obtener.	

FRQ-027	Avance colección Fósiles
El sistema deberá permitir ver el avance personal de la colección de Fósiles, así como los que faltan por obtener.	
FRQ-028	Avance colección personal
El sistema deberá permitir registrar el avance personal de las colecciones varias.	
FRQ-029	Mis vecinos
El sistema deberá permitir añadir o eliminar los vecinos del usuario.	
FRQ-030	Probabilidad mudanza
El sistema deberá permitir calcular la probabilidad de que mis vecinos se muden de la isla.	

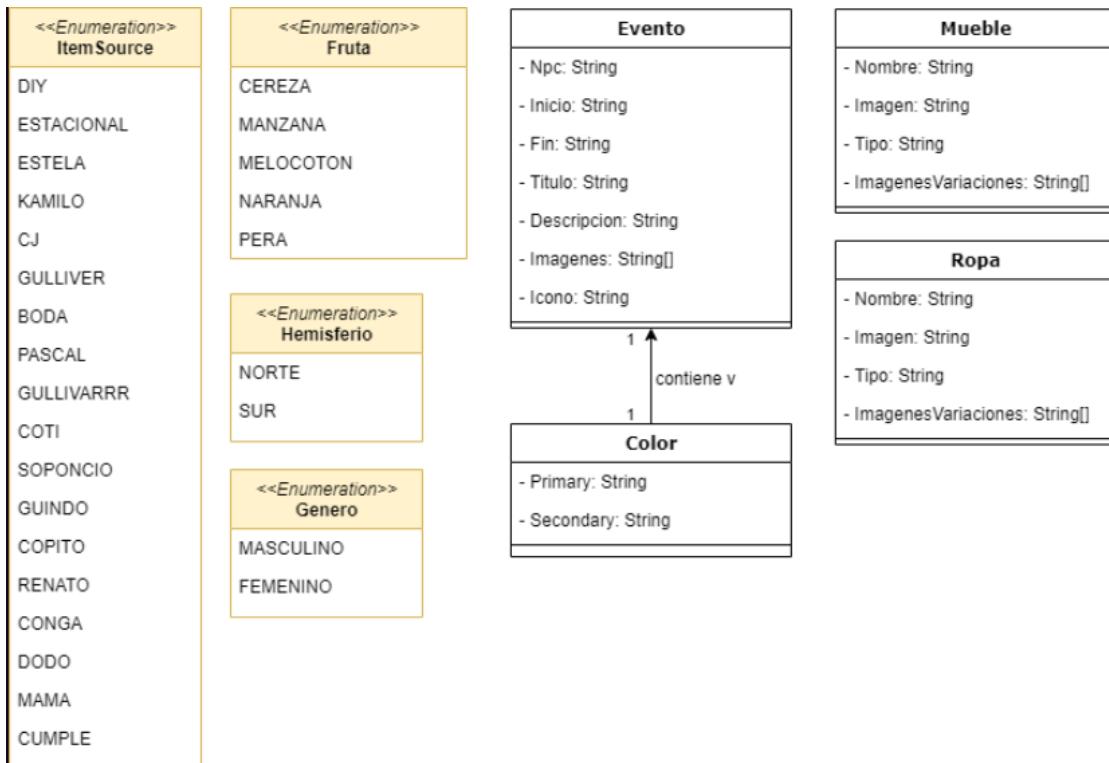
5.2.3. Requisitos no Funcionales

NFRQ-001	Tiempo de respuesta
El sistema deberá tener un tiempo medio de respuesta inferior a 1 segundo.	
NFRQ-002	Disponibilidad
El sistema deberá estar disponible las 24 horas del día.	

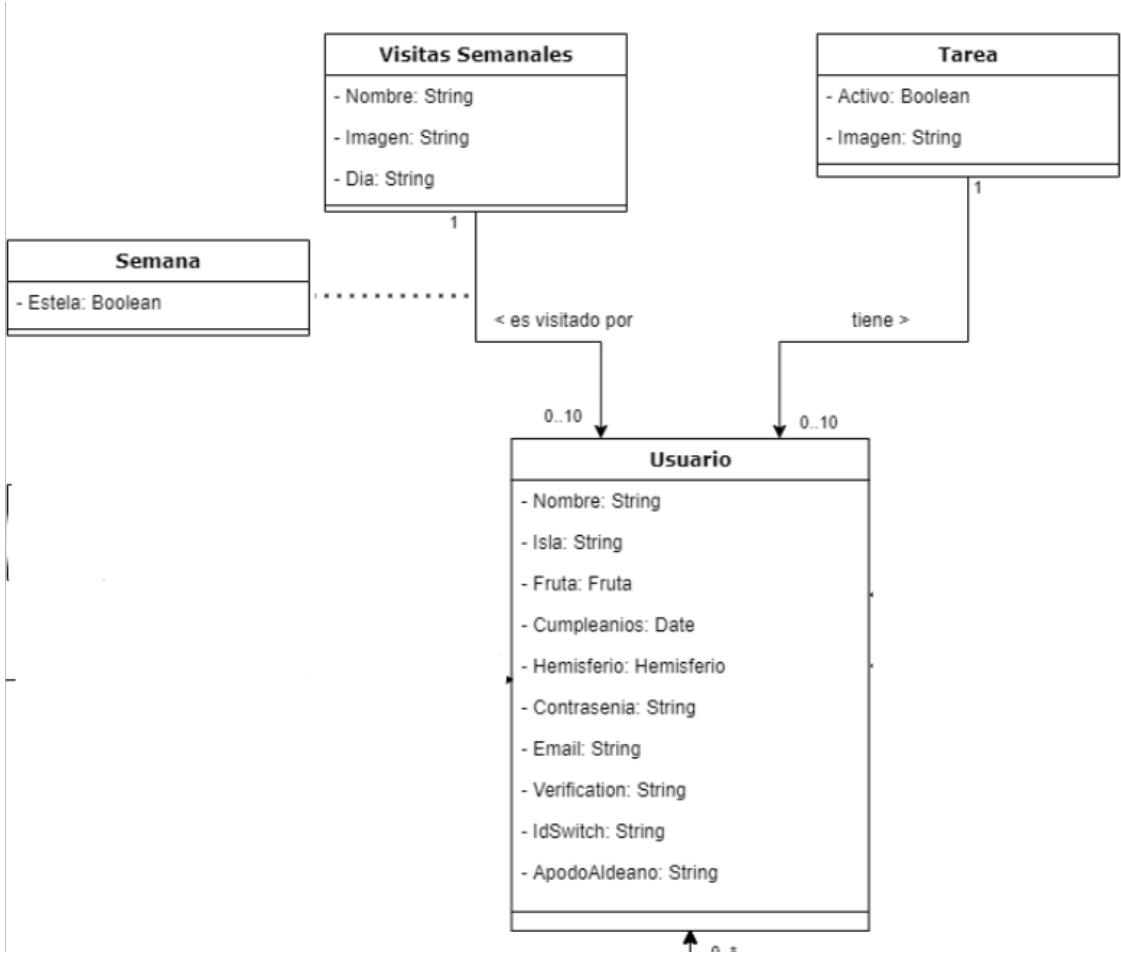
5.3– Diagrama de Clases

A continuación podemos ver un esquema a de alto nivel de las relaciones de las entidades de la aplicación, así como la información que contiene cada objeto. Dado que tratamos con grandes cantidades de información, gran parte de ella la tenemos que obtener a través de APIs.

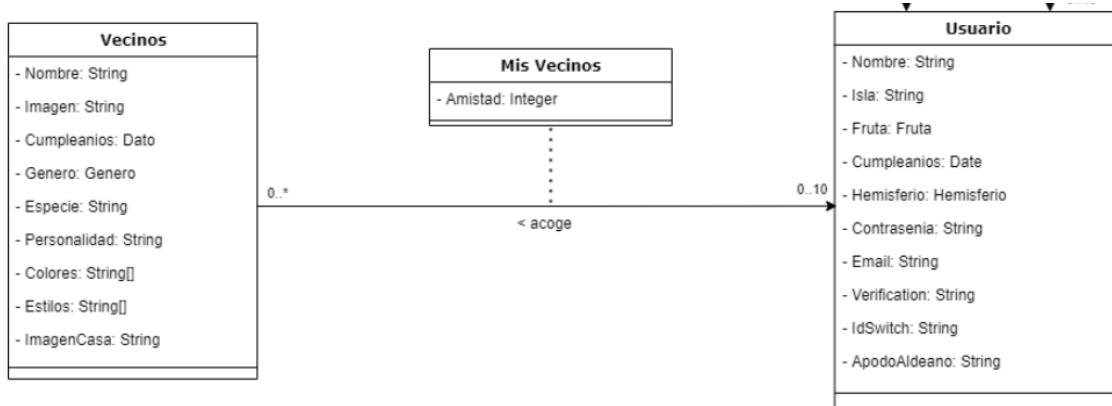
Debido al gran tamaño del diagrama de clases, vamos a verla dividida en partes más pequeñas de forma que muestren distintas relaciones y se pueda ver de forma más clara.



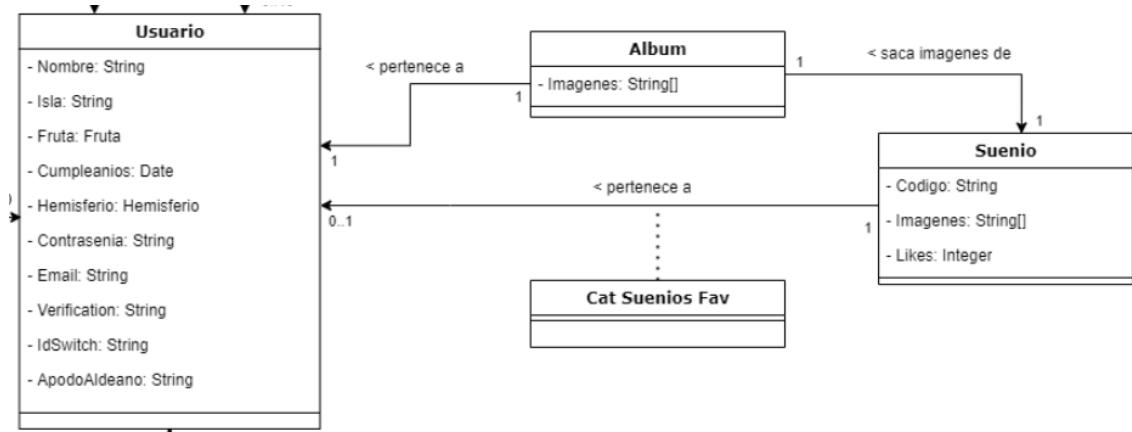
Primero podemos ver tanto los enumerados, como algunas clases que se encuentran más aisladas, ya que no forman parte del gran conjunto que rodea al usuario y por lo tanto, son datos puramente informativos y no se almacenará en base de datos. En este caso se trata de los muebles y las prendas de ropa (que como vemos, son idénticos), así como los eventos, que están relacionados con la clase Color, pero nada más lejos de dicha relación.



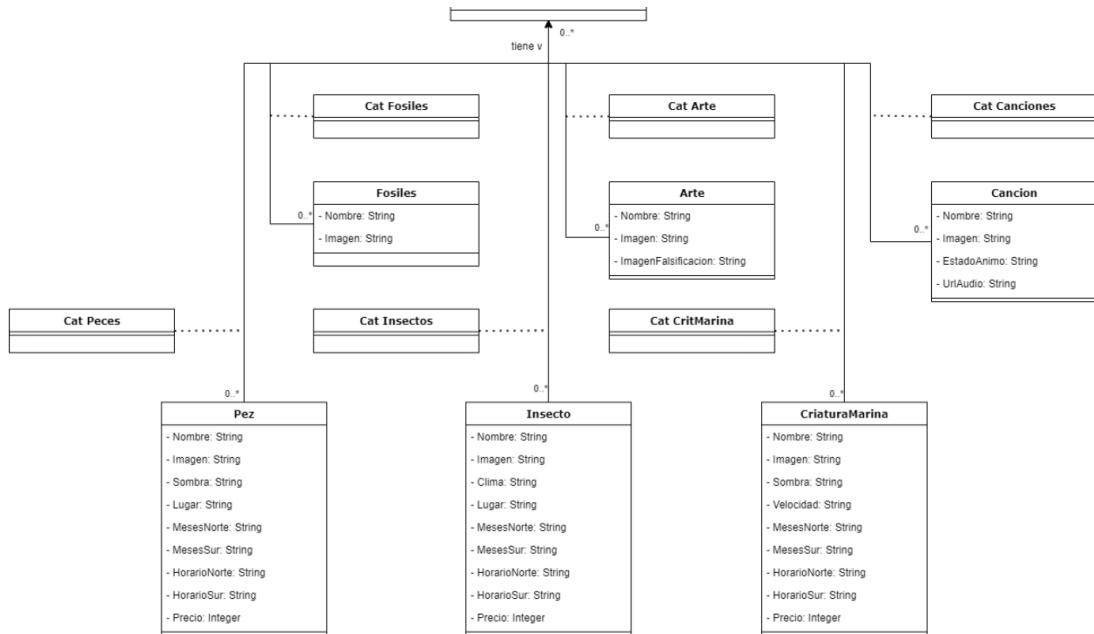
Pasando al conjunto principal, podemos empezar por la relación que comparte la clase Usuario con Visitas y Tareas. Un Usuario puede tener de 0 a 10 Tareas, al igual que pasa con Visitas Semanales, aunque esta tiene además una relación con usuario que indica si Estela (un personaje del juego) ha visitado ya la isla o no.



Luego, podemos ver la relación de Vecinos que, al mostrarse tanto en un catálogo (en el cual no se registra la relación con el usuario), como en el perfil (donde sí se opera y registra con dicha relación), vemos que tiene una relación Mis Vecinos que relaciona aquellos Vecinos que estén habitando en la isla del Usuario, además con el atributo Amistad para indicar la amistad que tiene cada uno con el jugador.

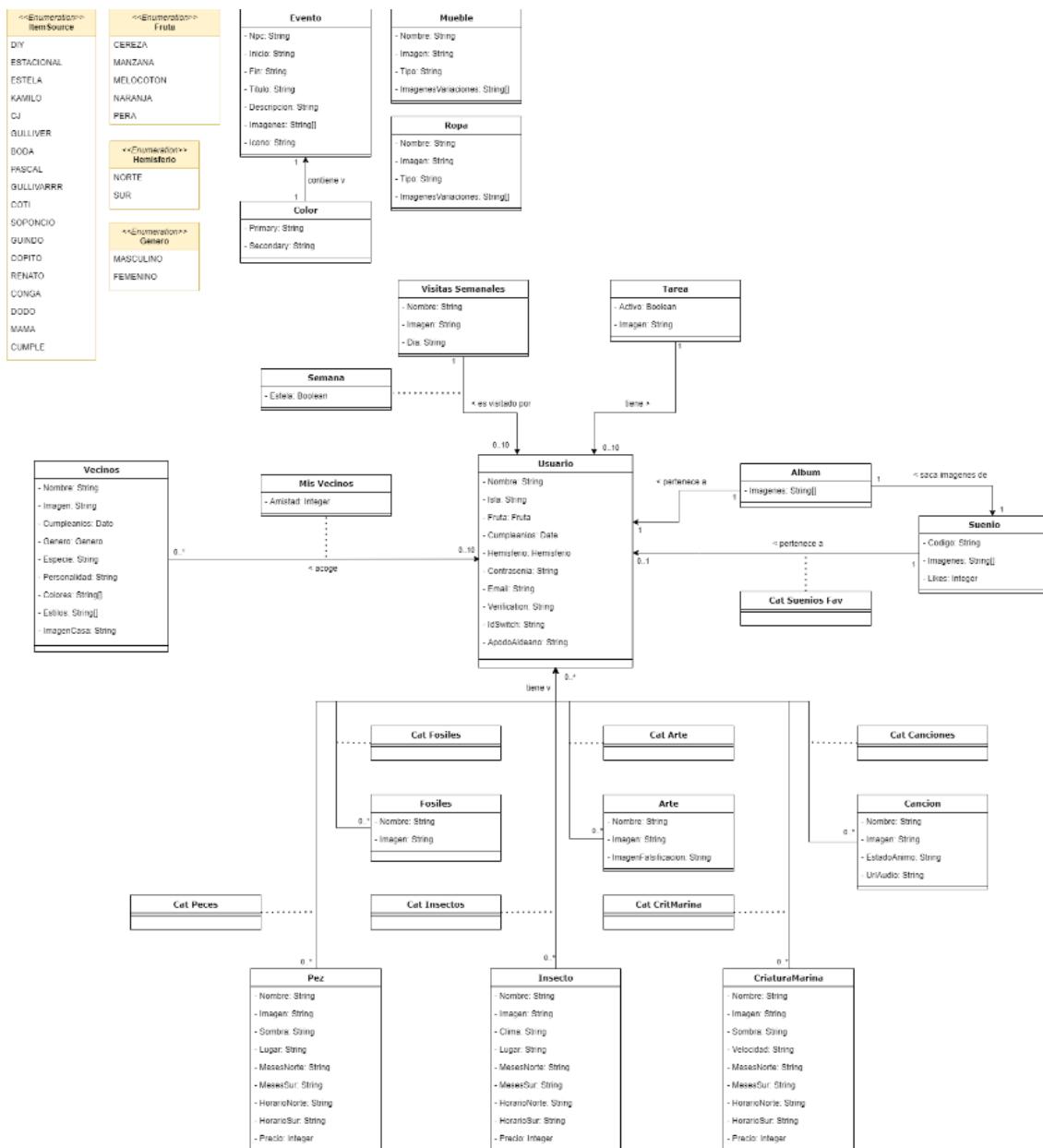


Seguimos con la relación de Usuario con Sueños. Esta entidad esta relacionada a su vez con Álbum, que también está relacionada con Usuario. Un Usuario dispone de un Álbum de fotos, las cuales se pueden utilizar para añadir imágenes al sueño de dicho Usuario. Además, un Usuario puede tener un listado de Sueños a los que le haya dado "Me Gusta".



Por último, tenemos una serie de listados que se relacionan todos de la misma manera con Usuario (la punta de la flecha superior). Cada entidad tiene su propia página de listado, y cada Usuario dispone de un catálogo personal de cada entidad en el cual se añaden aquellas entidades que ya haya obtenido.

Para terminar, se incluye una imagen completa del diagrama de clases para tener una vista global del mismo.



5.4– Diseño

A continuación ofrecemos una primera visión de lo que sería la interfaz gráfica del sistema, acompañado de algunas explicaciones para entender a priori el funcionamiento del sistema, ya que entraremos en detalle más adelante en el manual de usuario. En las imágenes aparecen anotaciones realizadas en rojo para el mejor entendimiento del estilo que se busca obtener de cara a la hora de implementar las vistas.

Empezando por la página de inicio (véase la figura 5.1), tendríamos el navegador en la zona superior para acceder a las diferentes secciones del sistema que no necesitan que el usuario se haya registrado, además de un botón para iniciar sesión en caso de que se disponga de una cuenta en la página.

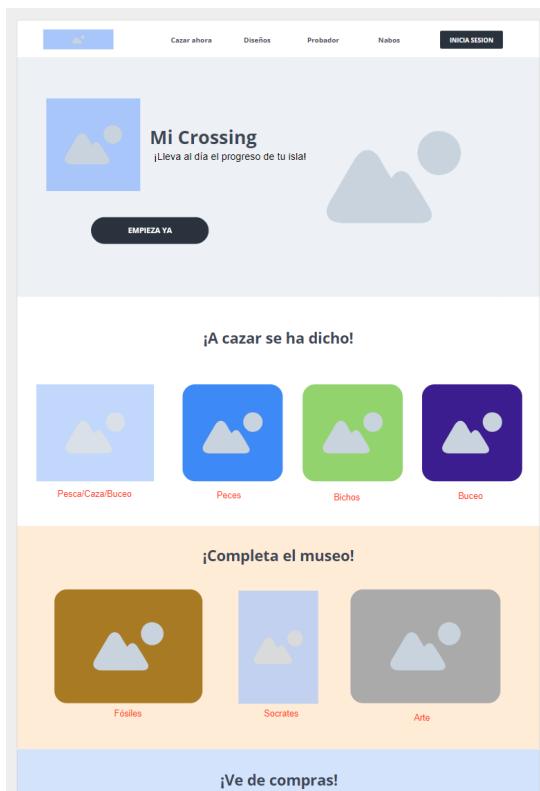


Figura 5.1: Inicio 1

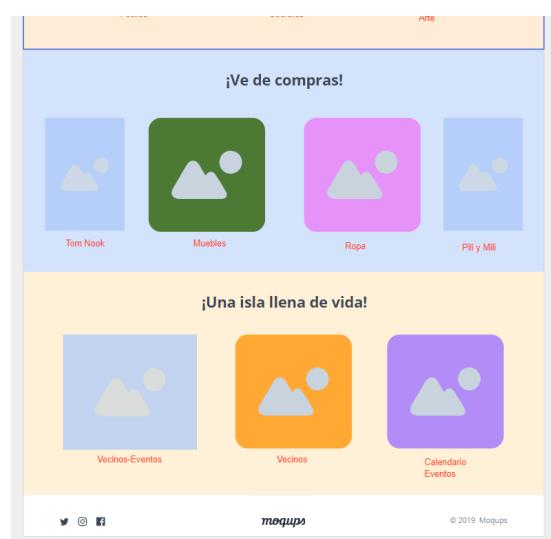


Figura 5.2: Inicio 2

Bajando vemos diferentes secciones, disponiendo en primer plano de la portada con un botón para acceder al registro. Más abajo se encuentran las diferentes secciones de enciclopedia con las que contará el sistema: peces, bichos, criaturas marinas, fósiles y obras de arte , así como del catálogo de muebles, ropa, vecinos e incluso un calendario de eventos (véase la figura 5.2).

Empezando por las secciones que hemos comentado en el párrafo anterior, a continuación tenemos un listado de peces, aunque el formato es genérico para los demás listados, cambiando algunos pequeños detalles en cada uno como ciertos filtros.

Podemos observar que arriba se encuentra la sección de filtros para que el usuario pueda realizar una búsqueda más específica de manera simple. Debajo de los filtros se encuentra el listado en sí, que muestra una recopilación de los peces (en este caso) disponibles en el juego. Cada ítem cuenta con una imagen, su nombre y algunos de los datos más relevantes, como precio de venta o localización, y si se hace clic en la celda se abre un menú con información más detallada, de forma que lo importante esté siempre a la vista y no haya que entrar en los menús de forma repetitiva.

En rojo se muestran aquellas criaturas que no se encontrarán disponibles el mes que viene, y además, si el usuario está registrado en el sistema y ha iniciado sesión, puede llevar recuento de los que ya ha capturado marcándolos con el tick superior derecho en cada ítem, cambiándose así el fondo a un color distinto para diferenciarlos a simple vista (véase la figura 5.3).

The screenshot shows a user interface for a fish encyclopedia. At the top, there's a navigation bar with tabs for 'General', 'Peces', and 'Fecha'. Under 'General', there are buttons for 'Anti-Spoilers', 'Ocultar capturados', and 'Desaparecen'. Under 'Peces', there are buttons for 'Río', 'Cascada', 'Estanque', 'Mar', and 'Desembocadura'. Under 'Fecha', there are buttons for months from 'Enero' to 'Diciembre', and a clock icon showing '18:45'. Below the navigation bar is the main content area titled 'Enciclopedia de Peces'.

The main content area displays a grid of fish cards. Each card contains an icon, the fish name, its price, and its location. Some cards have a checkmark in the top right corner. A note 'on hover, rectángulo con poca opacidad, meses y hora' is present. A note 'capturado' is above one card. A note 'desaparece mes que viene' is below another card. A note 'patrón de peces, movimiento' is at the bottom right.

Icon	Fish Name	Price	Location	Status
	Lubina	500	Mar	
	Pez Luna	2400	Mar	
	Salmón	500	Desembocadura	✓
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Pez Luna	2400	Mar	
	Salmón	500	Desembocadura	✓
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	
	Lubina	500	Mar	

Figura 5.3: Listado genérico

Para terminar con el contenido de la página de inicio (ya que a excepción del calendario, lo demás son todo listados como el que acabamos de ver), tendríamos el calendario de eventos (véase la figura 5.4).

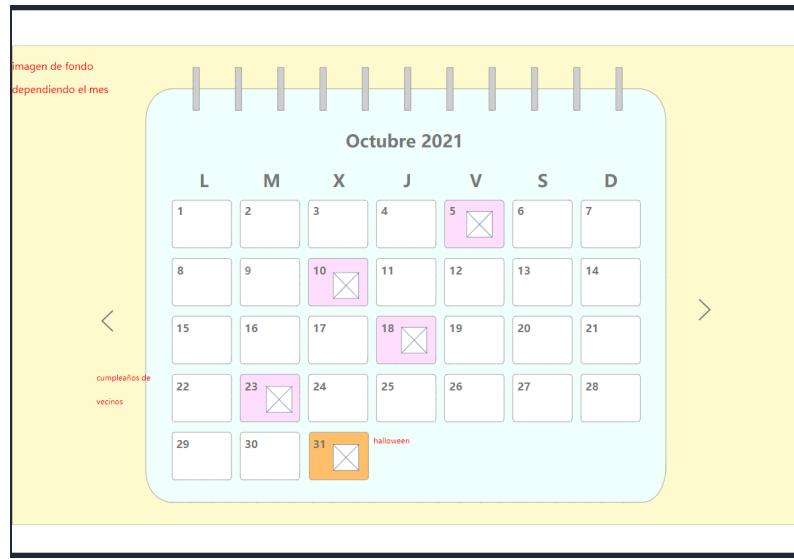


Figura 5.4: Eventos

Esta sección no es mas que un gran calendario donde el usuario puede ver tanto las festividades del juego como los cumpleaños de los vecinos. Sin embargo, dispondrá de información adicional para los eventos al hacer clic en ellos, de forma que se desplegará un menú (véase la figura 5.5) en el que podremos ver información útil de dicho evento, como visitantes, objetos temáticos, actividades especiales, etc. De esta forma el usuario puede tener una idea general de todo lo que se puede hacer en ese evento y planificarse mejor su agenda.

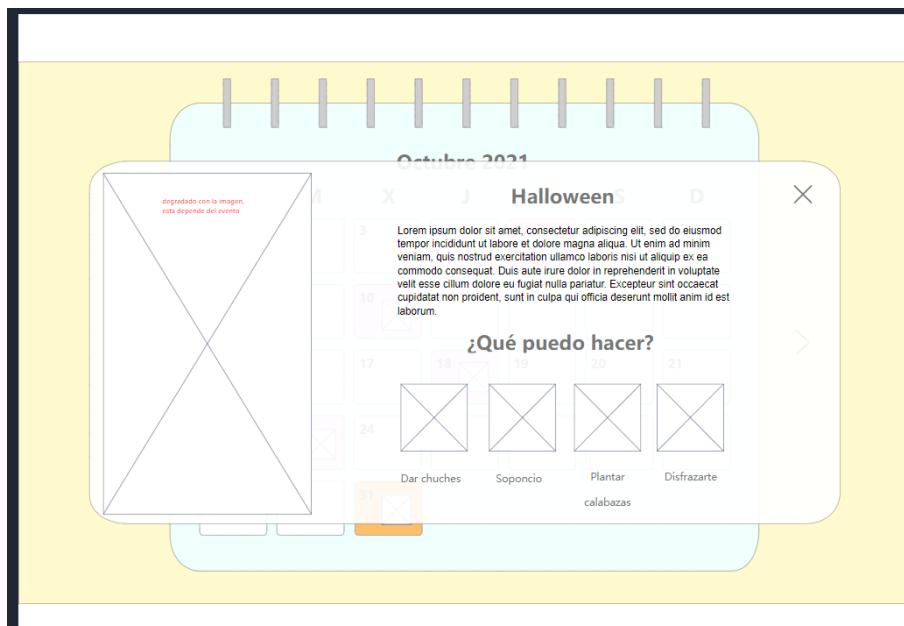


Figura 5.5: Eventos - Detallado

Una vez acabado con el contenido de la página principal, si el usuario quisiera registrarse accedería a la siguiente vista (véase la figura 5.6), la cual es una simple página de registro con información relevante al juego, como el nombre de la Isla, el tipo de fruta o la localización. Este último es bastante importante ya que dependiendo de si se encuentra en el hemisferio norte o sur, el usuario visualizará unos datos u otros.

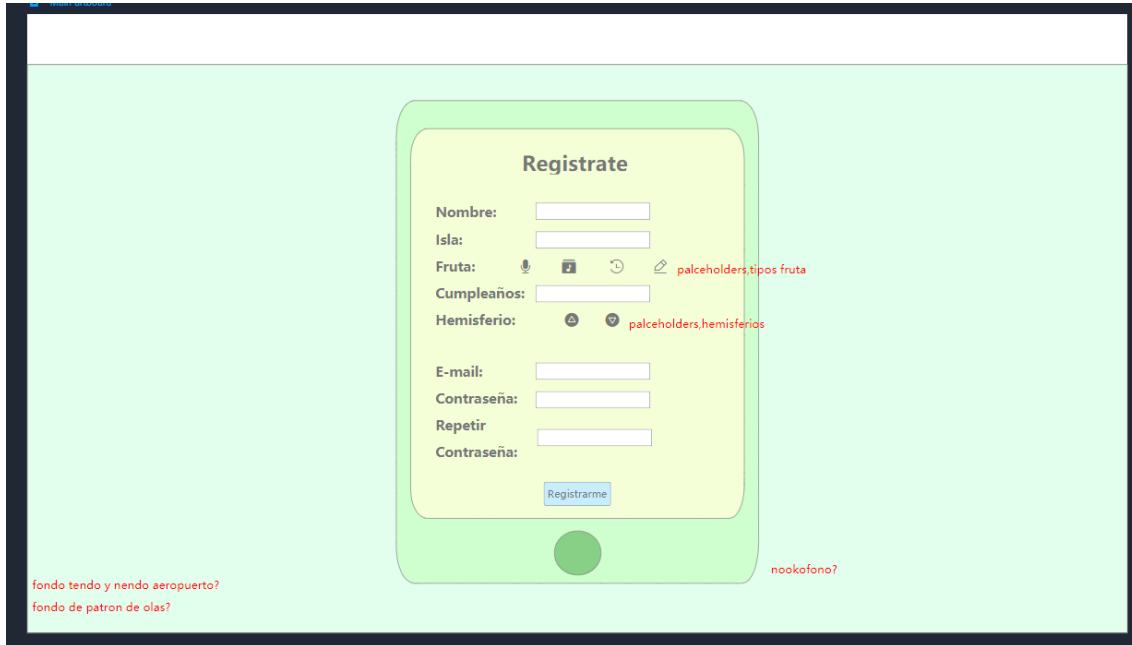


Figura 5.6: Registro

Si se completa el registro de forma satisfactoria, se puede acceder al perfil del usuario, el cual dispone de varias secciones (véase la figura 5.7).

A la izquierda se muestra la información del usuario a modo de resumen o biografía. A la derecha se encuentra la sección que se usará de forma diaria. Por una parte tenemos las tareas, las cuales se pueden editar con un ícono dependiendo del propósito que tengan y serán marcadas por el usuario una vez hayan sido realizadas, reiniciándose cada día. Por otra parte tenemos la lista de visitantes semanales, donde podremos observar los visitantes de la semana anterior e ir llenando los de esta semana en consecuencia.

Si bajamos, encontraremos un listado de los vecinos que actualmente residen en nuestra isla, con posibilidad de ir añadiendo hasta un máximo de diez vecinos. Al colocar el cursor sobre cualquiera de ellos, se mostrará información sobre el mismo.

Más abajo tenemos un pequeño listado de algunas colecciones de objetos especiales para que el usuario lleve un recuento de las que ya dispone y las pueda localizar de una manera más sencilla y organizada. Las colecciones irán variando dependiendo de que botón se encuentre activo, y dispondrá de un recuadro de texto para realizar una búsqueda específica.

Por último, tenemos una sección de galería que sirva a modo de portfolio para el usuario, para tener todas las capturas y/o vídeos recopilados en un mismo sitio.

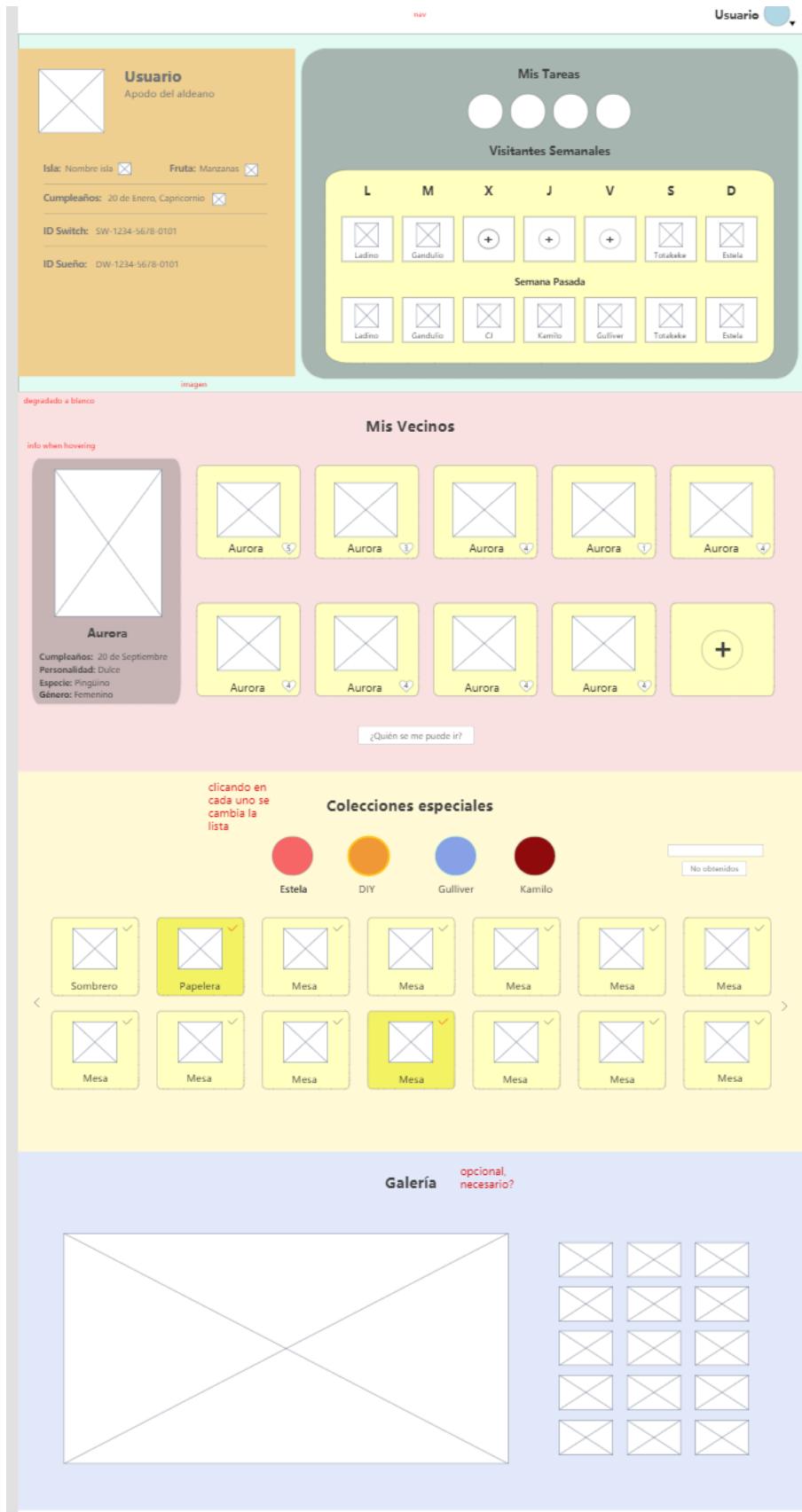


Figura 5.7: Perfil

Una vez acabado con el perfil, pasamos a las tres últimas vistas que se encuentran en el navegador principal de la página. Primero encontramos una página que es una recopilación de listados (como el mencionado anteriormente) de bichos, peces y criaturas marinas. La diferencia, aparte de encontrarse los tres juntos, es que en este listado aparecen todas las criaturas que se pueden atrapar actualmente, actualizándose a medida que lo hace el tiempo (véase la figura 5.8). Esto es útil ya que si un usuario quiere saber que criaturas puede cazar ahora en su isla, tan solo tiene que acceder a esta página y ya dispone de dicha información, sin registro ni pérdida de tiempo, ya que dispone de las tres colecciones de criaturas en la misma página.

Además, también dispone de una serie de filtros para realizar una búsqueda algo más específica, incluso con opciones de ocultar cierta información para evitarse así los "spoilers", de forma que sepa donde puede encontrar la criatura pero no sepa ni cuál ni cómo es, de esta forma puede tener una cierta ayuda a la hora de capturarla pero seguir teniendo la emoción de no saber qué es lo que le aguarda.

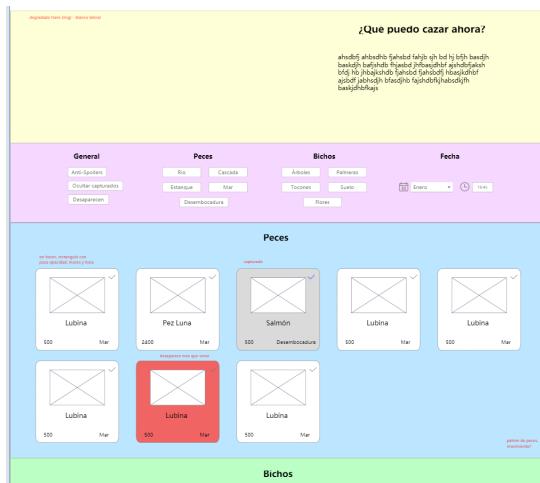


Figura 5.8: Cazar ahora 1

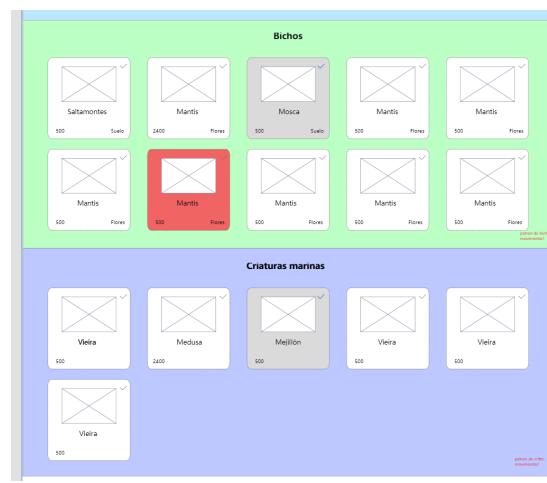


Figura 5.9: Cazar ahora 2

A continuación tendríamos la calculadora de nabos. En esta página se proporciona un poco de información acerca del mercado de nabos dentro del juego, de forma que el usuario entienda algo mejor como funciona y los distintos patrones de venta que existen (véase la figura 5.10).

En la parte inferior se encontraría la calculadora en sí, donde el usuario debe de escribir la información que haya ido recopilando para poder así realizar una predicción del patrón que hay actualmente en su isla. Además se acompañará de una gráfica y una tabla de precios por si quisiera información más detallada.

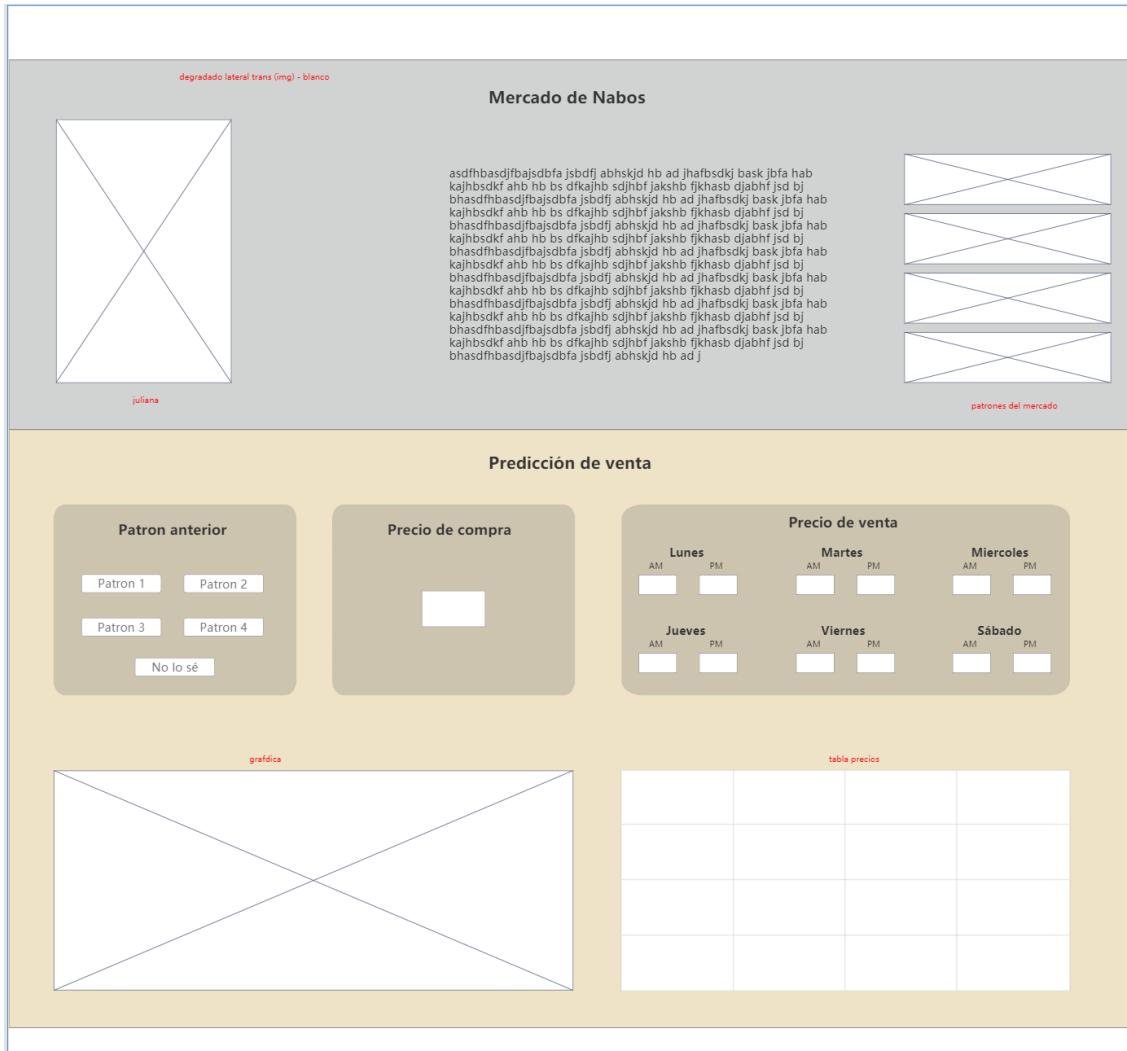


Figura 5.10: Calculadora de Nabos

Para terminar con este apartado y con el capítulo, por último veremos la vista del probador. Esta es una de las funcionalidades que crearemos y que consiste, como su propio nombre indica, en un probador de ropa (véase la figura 5.11).

En el centro tendremos una imagen de nuestro personaje, al cual podremos personalizar utilizando el menú superior, actualizándose a medida que seleccionemos las distintas opciones. Una vez elegido el personaje, la página dispondrá de un catálogo de ropa dividido en secciones (cabeza, torso, accesorios etc) para que el usuario pueda buscar la prenda que quiera probarse. Una vez la encuentre, con un clic sobre la prenda, esta aparecerá sobre el personaje central. Además se mostrarán en un desplegable todas las variaciones de color de las prendas para obtener así un catálogo completo. De esta forma el usuario puede realizar combinaciones de ropa para probar nuevos estilos y decidir que prendas necesita, en vez de ir comprándolas en el juego para luego cambiar de opinión.

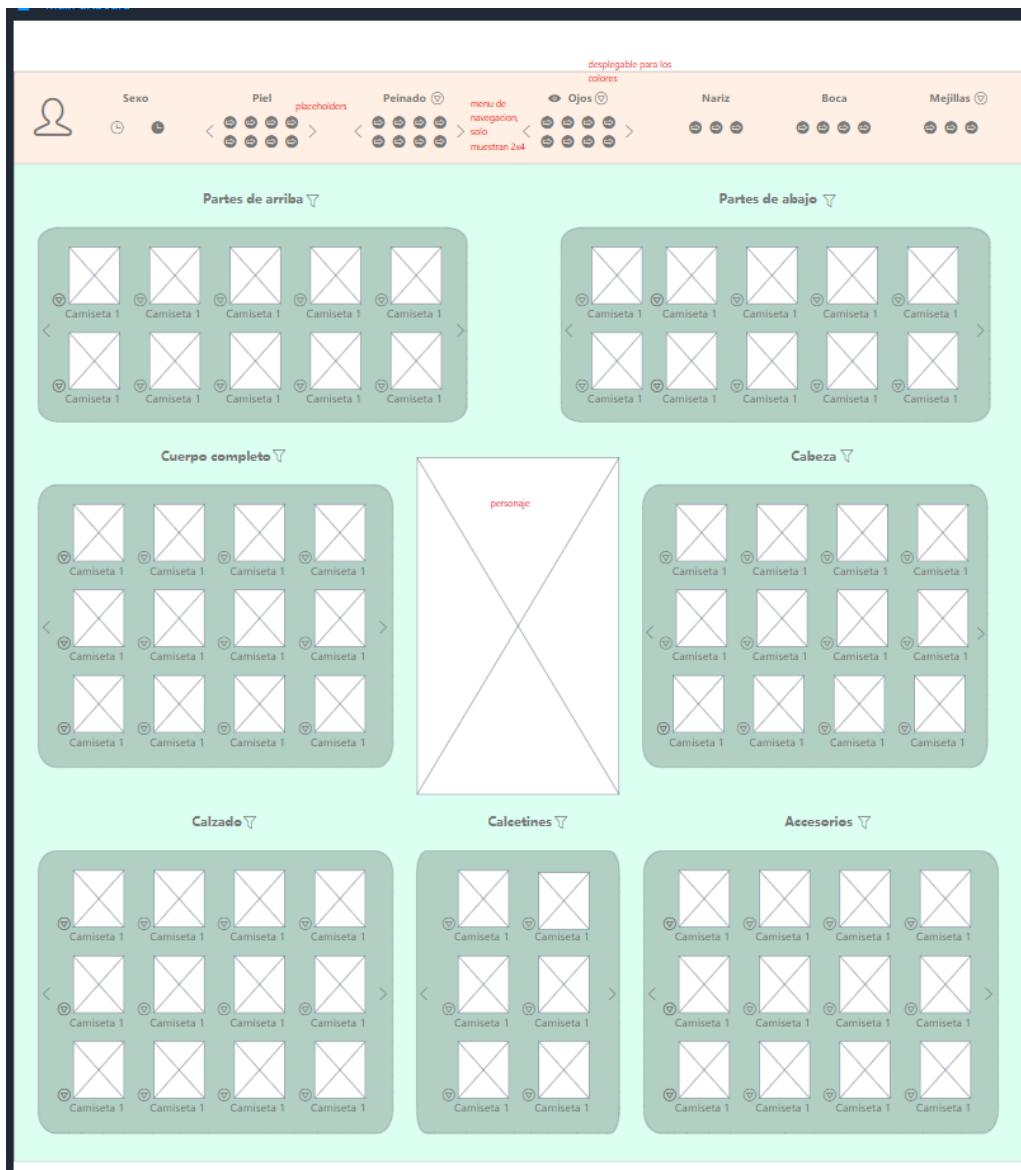


Figura 5.11: Probador

CAPÍTULO 6

Implementación

6.1– Entorno de desarrollo

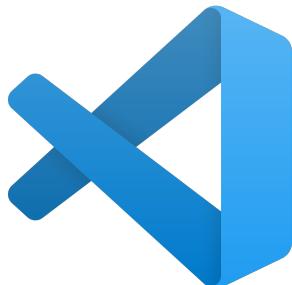


Figura 6.1: Logotipo Visual Studio Code

El entorno de desarrollo utilizado para este proyecto ha sido Microsoft Visual Studio Code, ya que es un IDE bastante versátil que soporta una gran variedad de lenguajes y que, al ser código abierto, presenta la posibilidad de instalar plugins creados por la comunidad para obtener funcionalidades útiles para cualquier tipo de lenguaje. Dado que vamos a trabajar con Angular, con la instalación de un paquete de plugins de Angular hemos podido trabajar de forma bastante cómoda y sin ningún problema.

Además, ofrece soporte para Git, de forma que se pueden realizar operaciones en el repositorio de forma bastante sencilla, así como gestionar los posibles errores que puedan surgir mediante la misma interfaz, por lo que resulta bastante cómodo.

La principal ventaja de Visual Studio Code es que, al ser un IDE universal, contamos con varias funciones generales de los IDE (debug, consola de comandos) pero aplicado a una gran variedad de lenguajes, de forma que con el simple hecho de tenerlo instalado ya se puede comenzar a trabajar (incluso sin la necesidad de plugins, aunque no es recomendable). Es por esto que con el simple hecho de disponer del IDE y descargar un par de plugins, ya se puede trabajar de forma cómoda, y además se puede adaptar de manera sorprendente ya que, si en algún momento hay que trabajar con otro lenguaje, se puede realizar perfectamente sin tener que descargar otro IDE. Directamente desde Visual Studio Code y con algún plugin extra, ya se puede incorporar el nuevo lenguaje y trabajar de forma cómoda con ambos, lo cual centraliza el trabajo en una misma aplicación y resulta más eficiente.

6.2– Tecnologías utilizadas

La idea principal de este proyecto era usar Angular como tecnología para el front-end por las razones ya explicadas anteriormente. Las demás tecnologías que fuéramos a usar no iban a tener tanto peso para nosotros como lo iba a tener esta, por lo que íbamos con la mente abierta dispuestos a usar la tecnología que, además de adaptarse bien a nuestro proyecto, fuera compatible con Angular. Al principio, pensamos en usar una base de Spring para el back-end, ya que es un framework con el que hemos trabajado bastante y además hemos adquirido experiencia en Java durante toda la carrera, pero debido a que no sabíamos nada sobre el funcionamiento de Angular (y todo lo que puede llegar a abarcar), pensamos que el peso de la aplicación se la llevaría el back-end (en Spring al principio). Sin embargo, una vez aprendimos y nos pusimos a dar nuestros primeros pasos en el framework, descubrimos que se puede realizar la gran mayoría de funcionalidades de la aplicación en Angular.

Además luego llegó la hora de conectar el proyecto en Angular con la base de datos. Estábamos acostumbrados a Spring, que realiza toda la gestión de base de datos por detrás en el mismo framework y tan solo es necesario inicializar los repositorios con las querys, así como las beans para popular la base de datos. Sin embargo, al estar trabajando con Angular, tuvimos que buscar información sobre como realizar esta conexión y descubrimos que se podía conectar a una base de datos gestionada mediante PHP con una simple petición HTTP al archivo que realizase la conexión y las operaciones oportunas. Esta idea nos gustó bastante ya que, no solo tendríamos que usar otro lenguaje (por lo que aprendemos más), sino que además podíamos utilizar la conexión con la base de datos a modo de API, haciendo peticiones HTTP desde Angular al correspondiente archivo PHP, el cual se encarga de abrir la conexión con la base de datos, realizar la consulta oportuna, y devolver los datos, los cuales Angular se encarga de recoger, transformarlos en caso de ser necesario, y finalmente, mostrarlos.

Es por esto que decidimos entonces usar XAMPP, ya que ofrece un servidor de Apache (en el cual se ejecutarían los archivos PHP que se encargasen de conectarse con la base de datos) así como una base de datos de MariaDatabase. Esto nos vino particularmente bien ya que en otras asignaturas habíamos usado XAMPP, y no solo eso sino que la base de datos de MariaDatabase esta basada en MySQL, que es el lenguaje que hemos dado durante toda la carrera, por lo que al estar trabajando ya con dos lenguajes que no conocemos del todo, uno familiar se recibe bastante bien.

Una vez montada la estructura de la base de datos y habiendo probado ya el funcionamiento de Angular y todo su alcance, nos dimos cuenta de que el back-end de Spring que habíamos pensado no era para nada necesario, ya que toda la lógica se realiza a través de este framework y las operaciones para obtener datos de la base de datos las íbamos a realizar con PHP, por lo que al final se descartó la idea de usar Spring como framework para el back-end ya que contábamos con todo lo que necesitábamos con la estructura que disponíamos.

6.2.1. Angular



Figura 6.2: Logotipo Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript de código abierto que suele ser utilizado para la creación de aplicaciones web de una sola página (SPA), además de seguir el diseño Modelo-Vista-Controlador (MVC).

Angular funciona mediante componentes, los cuales disponen de sus respectivas variables y métodos que se encargan de realizar la lógica, así como de operar con su vista asociada. De esta forma, obtenemos pequeños paquetes de código, cada uno con su vista y su lógica. Podría explicarse como si de un puzzle se tratase, pero con muchas posibles soluciones, siendo cada pieza un componente con su vista, sus variables y métodos. Cada pieza es funcional por sí misma, pero el objetivo es juntarlos con otros componentes para obtener la aplicación (o vista) completa.

Además de ser una forma bastante interesante de desarrollar una aplicación, es una muy buena forma de evitar la repetición de código, ya que si se necesita reusar un componente es tan fácil como importarlo y se encuentra listo. De la misma forma, se trabaja con servicios, directivas y etiquetas que pueden ser reutilizadas, reduciendo así la carga de trabajo considerablemente y aumentando la limpieza del código.

Además, una vez se esté ejecutando la aplicación, esta se actualiza con los nuevos cambios que se realicen, algo que resulta bastante cómodo a la hora de realizar algunas pruebas dado que no hay que estar ejecutando constantemente la aplicación.

6.2.2. XAMPP y Apache

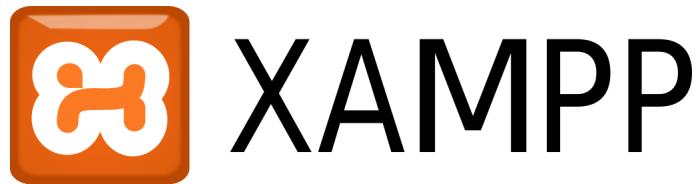


Figura 6.3: Logotipo XAMPP

XAMPP es un paquete de software libre que consiste en el sistema de gestión de bases de datos de MySQL, el servidor web Apache y los intérpretes para PHP y Perl.

Es una opción bastante buena para nuestro proyecto ya que resulta bastante sencillo de usar y ademas nos proporciona tanto el servidor web como la base de datos y el interprete para PHP, todo desde una misma aplicación.



Figura 6.4: Logotipo Servidor Web Apache

El servidor web de Apache es un servidor HTTP de código abierto multiplataforma con una arquitectura basada en módulos. Tiene una sección base que actúa como núcleo, y a esta base se le adjuntan módulos que aportan funciones básicas para el servidor web. Además, existen módulos externos para ampliar su funcionalidad, ya sea ofreciendo soporte para distintos lenguajes (Perl, Python, PHP y Ruby entre otros) u ofreciendo otras funciones como llevar un control del tráfico web.

6.2.3. PHP



Figura 6.5: Logotipo PHP

PHP es un lenguaje de programación libre especialmente adaptado para el desarrollo web y está orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Funciona del lado del servidor, por lo que el código es invisible al navegador y al cliente, ya que es el servidor el encargado de ejecutar el código y enviar el resultado HTML al navegador.

El código suele ser procesado en un servidor web por un intérprete PHP implementado como un módulo, y el resultado es representado mediante una petición HTTP. Es bastante versátil ya que puede ser desplegado en la mayoría de servidores web y en todos los sistemas operativos y plataformas sin coste alguno. Gracias a que es libre y a su flexibilidad, es uno de los lenguajes más populares como base para las aplicaciones web.

Su alcance es bastante amplio, ya que puede utilizarse insertando su código en archivos HTML, como para operar con bases de datos o incluso actuar como servicio REST. Es por esto que posee gran capacidad de conexión con la mayoría de motores de base de datos actuales (especialmente con MySQL y PostgreSQL).

6.2.4. MariaDB y MySQL



Figura 6.6: Logotipo MariaDatabase

MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL. Tiene una alta compatibilidad con MySQL (posee las mismas órdenes, interfaces, API y bibliotecas).



Figura 6.7: Logotipo MySQL

Respecto a MySQL, es un sistema de gestión de base de datos relacional considerada la base de datos de código abierto más popular del mundo. Si MariaDB era un sistema derivado con licencia GPL, MySQL está desarrollado bajo licencia comercial por parte de Oracle.

Su uso principal es orientado a aplicaciones web debido a la rapidez que posee a la hora de la lectura de datos. Además, al ser tan popular nos garantiza una gran base de información sobre posibles errores que podamos tener.

6.2.5. Git



Figura 6.8: Logotipo Git

Git es un software de control de versiones. Su propósito es llevar registro de los cambios en archivos, incluyendo la coordinación del trabajo que se realiza sobre archivos compartidos entre varias personas. A raíz de este software han surgido plataformas como GitHub o GitDesktop, todo enfocado a la gestión visual y sencilla de las funcionalidades que ofrece Git, así como el alojamiento de proyectos que utilicen dicho sistema de control de versiones.

Es uno de los software de control de versiones más populares, lo que ha provocado que muchas aplicaciones ofrezcan compatibilidad con Git, como puede ser Microsoft Visual Studio Code o Eclipse IDE por ejemplo.

Como ya se ha comentado, a raíz de Git nació GitHub, siendo también de las plataformas más populares para el alojamiento de proyectos y colaboración simultánea en los mismos. GitHub ofrece desde su servicio básico de alojar proyectos, hasta wikis por cada proyecto, tableros estilo Kanban, gráficas del trabajo y gran cantidad de adiciones para facilitar el trabajo colaborativo en dichos proyectos.

Además existen plugins que añaden funcionalidades a la plataforma o que la integran en otras aplicaciones para que su uso esté mas centralizado y sea más cómodo.

6.3– Dificultades encontradas y soluciones propuestas

Como es común en todos los proyectos, a medida que se ha ido avanzando en el desarrollo nos hemos ido encontrado con varios obstáculos (unos más grandes que otros) que nos han obligado a desviarnos un poco y buscar soluciones para poder continuar. A continuación se encuentran los obstáculos más significativos y cómo los hemos ido resolviendo:

6.3.1. It. 3 - Perfil

Esta ha sido la primera iteración en la que hemos empezado a programar y hemos desarrollado features. Al principio, dado que íbamos a trabajar con una API (Alexis8717, 2020) de la que sacaríamos todos los datos, pensamos que el perfil del usuario, así como las distintas funciones que habían dentro de este, no resultarían un gran trabajo ya que trataría más de consumir información y que, por lo tanto, sería un buen comienzo para empezar a iniciarnos en Angular.

Sin embargo, cuando nos pusimos a desarrollar código, vimos que no era tan simple y que todo lo que pensábamos que sería consumir información, eran en realidad features que había que realizar, con sus respectivos servicios, clases y lógica, ya que necesitábamos almacenar dicha información relacionándola con el usuario. Esto hizo que se aumentara bastante la carga de trabajo, así que para no cargarnos mucho en esta primera iteración de desarrollo, decidimos dejar la implementación de la API para el siguiente sprint (así como la realización de los tests, ya que no tiene sentido desarrollar unos tests para features que no están completas).

De este modo, aunque algo más laborioso de lo que se pensó en un principio, pudimos realizar un sprint de introducción al lenguaje (del que hemos aprendido bastante) que nos sirve como base para los siguientes sprints, ya que con cada feature que hemos ido desarrollando, hemos descubierto más y más sobre el framework, adquiriendo así las nociones necesarias para avanzar de forma eficiente.

6.3.2. It. 3 - Mantener sesiones

En esta iteración decidimos repartir las tareas de forma que uno de nosotros empezara implementando todo lo relacionado con el registro y el inicio de sesión mientras que el otro se iba encargando de otras tareas, en este caso, de las funciones dentro del perfil.

El problema comenzó con lo mencionado en el punto anterior: se combinó el aumento de carga de trabajo con las dificultades que encontramos a la hora de mantener una sesión en la página, ya que era necesario para la mayoría de tareas del perfil. Además, como el servidor de Apache donde se encuentra la base de datos y el servidor de Node.JS en el que se ejecuta Angular están localizados en puertos distintos, no se podía realizar una sesión de forma normal (al igual que muchas otras funciones de PHP como acceder a los datos de una operación POST mediante la variable por defecto de PHP).

Esto ocasionó que tuviéramos que dar un rodeo buscando posibles soluciones hasta dar con la actual, que se basa en almacenar un código de verificación que se borre exclusivamente al cerrar sesión, de forma que se mantiene la sesión mientras el usuario tenga asignado un código de verificación que no sea nulo. Debido a este rodeo y al tiempo que llevó tanto encontrar la solución como su implementación, hubo cierta dependencia entre tareas, lo que ocasionó un ligero aumento del tiempo inicial pensado para dichas tareas.

6.3.3. It. 3 - Features del perfil

En la feature de “Mis vecinos”, teníamos que calcular el porcentaje que tiene cada vecino de mudarse de isla, dependiendo de la amistad que tenga con el jugador y de ciertos parámetros que hayan podido ocurrir. Este algoritmo pensábamos que ya lo habrían sacado del juego y podríamos acceder a él, sin embargo no fue así, por lo que tuvimos que realizar cierta investigación para dar con la fórmula que calcula dicha probabilidad y acto seguido, implementar nosotros el algoritmo (Agradecimientos al usuario de Twitter *Ninji* que publicó la lógica y las matemáticas que sigue el juego para calcular la probabilidad de mudanza, información a partir de la cual pudimos implementar el algoritmo).

6.3.4. It. 4 - API incompleta

En la iteración 4 debíamos hacer un repaso por las features realizadas en la iteración anterior de forma que añadiésemos todo lo necesario relacionado con la API que íbamos a utilizar (Alexis8717, 2020) para acceder tanto a imágenes como a datos del juego, pero debido a que no nos habíamos informado tanto como debíamos sobre la API, descubrimos que no tiene ningún método de filtro mas que obtener objetos por su id, por lo que nos obligaba a hacer una petición con todos los objetos e ir filtrando por “fuente” (colección) para seleccionar los que nos interesaban. Esto era posible pero suponía un gran coste computacional, especialmente si lo implementamos de forma que cualquier usuario pueda hacer dicha petición.

Ante este problema, dimos con la idea de crear un usuario administrador (el cual no había hecho falta hasta ahora) que pudiera ejecutar dicha petición para actualizar el catálogo de colecciones, de forma que con una sola petición por parte del administrador, se actualizan los ítems disponibles para los usuarios, de forma que estos acceden a una tabla de la base de datos con datos mínimos (fuente e id). De esta forma, aunque guardamos algunos datos en la base de datos, nos ahorraremos tanto el realizar peticiones de forma constante en la que recibimos cientos de objetos, como el filtrarlos para poder obtener los deseados, aumentando así la velocidad de respuesta.

Sin embargo, una vez nos pusimos a implementar más a fondo la API, nos dimos cuenta de que, aunque bastante completa para algunos aspectos del juego, para otros no estaba tan completa y carecía de muchos datos que necesitábamos. Además solo implementaba métodos para obtener o todo el listado de datos, o uno solo (mediante su ID), pero en ningún momento nos dejaba filtrar, por lo que iba a ser un trabajo que íbamos a tener que realizar a mano y que además sería bastante costoso computacionalmente hablando.

Es por eso que realizamos otra búsqueda en la web para encontrar una solución distinta que nos proporcionase todo lo que necesitábamos. Primero encontramos otra API (KevinPayravi, 2020b) que, aunque disponía de más métodos y alguna información que nos podía resultar útil, estaba aun a mitad de desarrollo, por lo que también estaba algo incompleta. Se pensó en implementar ambas APIs, pero el resultado no era lo que buscábamos, era una solución algo compleja y no nos convencía del todo, pero parecía ser la única opción.

Sin embargo, tras seguir buscando encontramos una librería (Norviah, 2020) que se podía instalar directamente en Angular que disponía de toda la información que necesitábamos, así como ciertas traducciones e imágenes, por lo que decidimos usar esta librería ya que, no solo nos daba todo lo que necesitábamos sino que además su uso era extremadamente sencillo y además nos permitía desechar la idea del administrador, que no nos terminaba de convencer.

6.3.5. It. 5 - Código reciclado

En la iteración 5 introdujimos la calculadora de nabos y el calendario de eventos entre otras features. Para la calculadora necesitábamos el algoritmo que calculaba los precios establecidos en el juego. Dicho algoritmo había sido datamineado por un usuario ya mencionado anteriormente en esta memoria (Ninji, 2020).

El problema surge a raíz de que dicho algoritmo, que se encuentra subido en la plataforma GitHub, estaba para otro lenguaje distinto al de nuestro proyecto, por lo que no pudimos usarlo directamente. Al principio consideramos intentar adaptarlo, pero era tan extenso y complejo que no merecía la pena dedicarle tantas horas a esa tarea, por lo que optamos por otra solución.

De entre varias webs que ya disponen de esta herramienta, hay una (Autores, 2020) que está realizada al completo en JavaScript, por lo que se realiza todo el trabajo por parte del cliente. Investigando descubrimos que dicho código también se encontraba en GitHub bajo la licencia de Apache 2.0, la cual nos permite usar el código mientras que cumplamos ciertas reglas, por lo que decidimos usar lo que necesitáramos del código, ya que no solo calcula y predice los precios futuros, sino que genera una tabla de precios bastante útil para el usuario.

El problema es que, al ser código reciclado y además en ficheros JavaScript, nos llevamos un tiempo haciendo pruebas para comprender el funcionamiento y la integración con nuestra aplicación, ya que hasta ahora habíamos estado trabajando con los ficheros TypeScript de Angular y no habíamos integrado ningún fichero JS externo. Esto conllevó varias pruebas hasta que finalmente conseguimos realizar la integración de forma satisfactoria, pero a cambio de cierto tiempo empleado en búsqueda de posibles soluciones, comprensión del código e integración con nuestra aplicación.

Otro problema que surge a raíz de lo mismo fue que tanto para la calculadora de nabos como para el calendario (especialmente para éste último) tuvimos bastante limitada la personalización (Lewis, 2018). El calendario hace uso de una aplicación de Angular ya creada que nos ofrece un calendario con posibilidad de añadir nuestros propios eventos, así como muchas más posibilidades. El problema es que nuestro calendario es mucho más básico ya que lo único que busca es mostrar información de eventos preestablecidos, por lo que era necesario modificarlo para adaptarlo a nuestra aplicación.

Pero claro, aunque estaba preparado para poder personalizarlo bastante, había que realizar muchos cambios, por lo que al igual que nos ocurrió con la calculadora de nabos, tuvimos que dedicarle algo de tiempo a comprender como funcionaba y más aún a realizar los cambios. Lo malo es que el calendario se generaba automáticamente, es decir, no disponíamos de todo el HTML desde el inicio, por lo que los cambios para añadir o quitar elementos tuvimos que realizarlos mediante CSS y JS, lo que se nos complicó bastante ya que era código reciclado y teníamos que tener cuidado al operar para no destrozar el funcionamiento ni afectar a más de lo que queríamos.

Dicho esto, pudimos modificarlo de forma satisfactoria tras dedicarle algo de tiempo. Aunque no obtuvimos el resultado que queríamos al 100 %, si que nos acercamos bastante a lo que buscábamos, por lo que nos ahorró bastante más tiempo que si lo hubiéramos implementado desde cero.

6.3.6. It. 5 - Feature demasiado compleja

Una de las features que queríamos añadir en este sprint era el "Probador de ropa", que consistía que una página donde el usuario pudiera crear un personaje del juego a su gusto, así como vestirlo con las distintas prendas del juego para, de esta forma, poder probarse distintos conjuntos sin tener que esperar a que apareciesen en la tienda del juego o sin tener que gastarse mucho dinero en varias prendas de ropa.

En un principio esta herramienta se planteó de forma que la prenda que eligiese el usuario iría superpuesta a la imagen del personaje en la zona correspondiente (por ejemplo si es una camiseta, iría en el torso). Esta idea no estaba exenta de fallos ya que dependíamos de la postura de las prendas de ropa (la gran mayoría vienen en "T-pose") y había varias, especialmente aquellas de manga larga, que por ejemplo tenían una manga doblada, lo que nos impedía representar por completo la ilusión de que el personaje llevaba la prenda.

Aun así, no eran muchas prendas, por lo que al ser una herramienta tan interesante decidimos seguir adelante e incluirla en el planteamiento. Sin embargo, al llegar a este sprint donde tocaba su implementación, fue cuestión de poco tiempo el darnos cuenta de que era inviable. No solo es que hubiese algunas prendas con una manga doblada, sino que había tipos de prenda completos cuya imagen no podíamos representarlo como si el personaje estuviera llevándola puesta (por ejemplo, los zapatos se ven desde una posición cenital y los calcetines se muestran en pareja, por lo que no hay manera posible de representarlos).

Tras investigar todo lo posible, vimos que eran tantos los fallos visuales que se cometerían que no rentaba realizar una herramienta tan completa para obtener un funcionamiento tan mediocre. Intentamos optar por buscar alguna herramienta del estilo ya realizada o algo que pudiésemos utilizar para intentar salvar la herramienta, pero por desgracia no dimos con nada por lo que optamos por desechar la funcionalidad.

Finalmente intentamos buscar otra feature para reemplazar a ésta, pero ya no quedaba demasiado por añadirle a la página que le resultara útil al usuario más que párrafos y párrafos de información, que aunque útiles, no es lo que busca nuestra página, por lo que optamos por no realizar ninguna y dedicar el tiempo de implementación de dicha herramienta tanto al apartado visual de la página como a pulirla todo lo posible.

Cabe decir que la implementación del probador de ropa es posible realizarla, solo que para obtener un buen resultado habría que dedicarle horas y horas de recolección de imágenes del juego, así como de edición manual de las mismas mediante un programa de edición fotográfico para que se adapten al resultado que buscamos. Sin embargo, no solo es un trabajo manual (y por lo tanto, va en contra de lo que queremos ya que buscamos hacerlo todo de la forma más programática posible) sino que además sería demasiado extenso y en cuestión de trabajo/tiempo no podíamos dedicárselo.

6.3.7. It. 6 - Integración continua

Aunque la idea principal consistía en realizar las features así como sus respectivos tests en el mismo sprint en el que se desarrollaban, por cuestiones de tiempo y otras razones que variaban a medida que avanzábamos en el proyecto, nos vimos en la necesidad de ir aplazando el desarrollo de tests, hasta que finalmente decidimos dejarlos para el último sprint.

Cuando por fin pudimos empezar a realizar las pruebas, lo primero que pensamos fue en utilizar un sistema de integración continua (como puede ser TravisCI), de forma que se ejecu-

tasen los tests con cada subida al repositorio, obteniendo de esta forma un reporte de posibles errores periódicamente con el que poder asegurarnos de que la aplicación funciona correctamente en cada subida.

Sin embargo, la estructura que hemos establecido no era la más conveniente para TravisCI (o cualquier otro servicio de integración continua), ya que los tests combinaban tanto frontend (Angular) como backend (PHP), y como veríamos más adelante (y como se indica en el siguiente punto), dado que disponemos de un sistema dividido en dos servidores separados (PHP y Angular) que se comunican mediante peticiones HTTP, descubrimos que la integración continua no sería posible implementarla en nuestro proyecto.

Tras una exhaustiva investigación sobre el tema, buscando posibles soluciones o alternativas, se ha llegado a la conclusión de que la implementación daría más problemas que beneficios, por lo que decidimos prescindir de dicha herramienta y ejecutar los tests manualmente cuando fuera necesario.

6.3.8. It. 6 - Despliegue

Uno de los últimos problemas que hemos tenido ha sido el despliegue online de la aplicación. Dada la combinación que hemos usado de Angular + PHP + MariaDB, necesitábamos desplegar la aplicación en dos servidores distintos: uno para el front (Angular) y otro para el back (PHP). Además, también necesitábamos una base de datos desplegada para realizar las consultas, que fue lo que provocó el principal problema.

Vimos que había varios servicios online de alojamiento de bases de datos (en particular, de MariaDB con PhpMyAdmin, que era lo que necesitábamos), y además eran gratuitos, por lo que decidimos optar por uno de estos. Sin embargo, al ser gratuito tenía que tener algunos límites, siendo uno de estos la imposibilidad de crear eventos SQL.

Esto nos suponía un leve problema ya que las tareas del perfil se desactivan diariamente mediante un evento SQL, por lo que si usábamos esta opción, no dispondríamos de dicha función en la aplicación. Al ver esto, buscamos otros servicios de alojamiento gratuito de bases de datos, pero no solo seguían teniendo el mismo límite, sino que además eran más restrictivos en comparación con la que pensábamos usar.

Sin embargo, este problema puede ser solucionado de varias formas: o bien alojando nuestra BBDD en un servicio que permita la creación de eventos (aunque sería un servicio de pago); o bien optando por reiniciar las tareas de otra forma distinta en el código (aunque no lo consideramos óptimo ya que los eventos SQL están precisamente diseñados para realizar este tipo de tareas).

Es por eso que, siendo lo ideal resolverlo mediante la primera opción (ya que en un despliegue real de la aplicación necesitaríamos un servicio con mayor número de consultas y otras características, entre las cuales vendría el servicio de creación de eventos), hemos decidido usar esta opción gratuita de alojamiento y prescindir de dicho evento (especialmente, tratándose solo de un evento cuya función equivale a hacer clic en el botón de la tarea y apenas va a ser notable).

6.3.9. It. 6 - CORS Policy

La política de CORS ha ocasionado problemas durante gran parte del desarrollo de la aplicación. Sin embargo, haciendo algunos ajustes en las cabeceras HTTP conseguimos evitar la aparición de más errores mientras que seguíamos desarrollando.

El problema vino una vez acabado el desarrollo, cuando nos pusimos a desplegar la aplicación, ya que como necesitamos desplegar en dos aplicaciones diferentes el backend y el frontend, al realizar peticiones entre sí otra vez ocurrían los errores. Esto se debe a que CORS, cuando está en producción, antes de cualquier petición HTTP manda una petición con el método OPTION, para comprobar que es correcta, y una vez recibe una respuesta satisfactoria, manda la petición original.

En principio pensábamos que con las cabeceras ya valdría, pero estábamos equivocados ya que, al enviar la primera petición, necesita un parámetro en una de las cabeceras, y dicho parámetro no puede encontrarse en la cabecera de la segunda petición (la original), por lo que no podíamos asignar la misma cabecera a todas las peticiones

Sin embargo, tras una intensiva búsqueda en la web, dimos con una librería para Angular que con un par de líneas de código, configuraba automáticamente todo lo respectivo a CORS, y una vez instalado y configurado conseguimos que funcionase sin ningún problema.

6.3.10. It. 6 - AdBlock

Aunque no es un problema grave per se, si que es un apunte a tener en cuenta y es que, si el usuario tiene instalado en su navegador algún addon para evitar publicidad, por cómo funcionan estos addons, hay algunas imágenes que no serán visibles, o que se mostrarán con un placeholder, ya que la misma url de la que se obtiene la imagen puede tener unos caracteres determinados que estén en la lista negra del addon y éste proceda a bloquearlo, pensando que se trata de publicidad.

Aun así, este problema (que parece afectar exclusivamente a la carga de algunas imágenes) se puede solucionar desactivando dicho addon en la página.

CAPÍTULO 7

Pruebas

A continuación, se muestra el plan de pruebas que hemos llevado a cabo para probar el correcto funcionamiento de nuestra aplicación.

Caso de prueba	CP-001
Descripción	Autenticación -> Debería cerrar sesión.
Acciones	Se accede a authentication mediante logout con datos correctos.
Resultado	Cierra la sesión.
Éxito	Si.

Caso de prueba	CP-002
Descripción	Autenticación -> No debería cerrar sesión por usuario incorrecto.
Acciones	Se accede a authentication mediante logout con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-003
Descripción	Autenticación -> No debería cerrar sesión por verificación incorrecta.
Acciones	Se accede a authentication mediante logout con verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-004
Descripción	Autenticación -> Debería iniciar sesión.
Acciones	Se accede a authentication mediante login con datos correctos.
Resultado	Inicia la sesión.
Éxito	Si.

Caso de prueba	CP-005
Descripción	Autenticación -> No debería iniciar sesión por usuario incorrecto.
Acciones	Se accede a authentication mediante login con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-006
Descripción	Autenticación -> No debería iniciar sesión por contraseña incorrecta.
Acciones	Se accede a authentication mediante login con una contraseña incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-007
Descripción	Autenticación -> Debería leer.
Acciones	Se accede a authentication mediante read con datos correctos.
Resultado	Nos devuelve los datos del usuario.
Éxito	Si.

Caso de prueba	CP-008
Descripción	Autenticación -> No debería leer por usuario incorrecto.
Acciones	Se accede a authentication mediante read con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-009
Descripción	Autenticación -> No debería leer por verificación incorrecta.
Acciones	Se accede a authentication mediante read con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-010
Descripción	Autenticación -> Debería obtener claves.
Acciones	Se accede a authentication mediante getKey con datos correctos.
Resultado	Nos devuelve algunos datos del usuario.
Éxito	Si.

Caso de prueba	CP-011
Descripción	Autenticación -> No debería obtener claves por usuario incorrecto.
Acciones	Se accede a authentication mediante getKey con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-012
Descripción	Autenticación -> No debería obtener claves por verificación incorrecta.
Acciones	Se accede a authentication mediante getKey con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-013
Descripción	Autenticación -> Debería registrar.
Acciones	Se accede a authentication mediante register con datos correctos.
Resultado	Crea el usuario correctamente.
Éxito	Si.

Caso de prueba	CP-014
Descripción	Autenticación -> No debería registrar otra vez.
Acciones	Se accede a authentication mediante register con el mismo usuario anterior.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-015
Descripción	Cat-fósiles-> Debería leer un fósiles.
Acciones	Se accede a cat-fósiles mediante read con datos correctos.
Resultado	Nos devuelve los fósiles que posee el usuario.
Éxito	Si.

Caso de prueba	CP-016
Descripción	Cat-fósiles-> Debería borrar un fósil.
Acciones	Se accede a cat-fósiles mediante delete con datos correctos.
Resultado	Elimina el fósil del usuario.
Éxito	Si.

Caso de prueba	CP-017
Descripción	Cat-fósiles-> Debería añadir un fósil.
Acciones	Se accede a cat-fósiles mediante create con datos correctos.
Resultado	Crea el fósil para el usuario.
Éxito	Si.

Caso de prueba	CP-018
Descripción	Cat-insectos-> Debería leer.
Acciones	Se accede a Cat-insectos mediante read con datos correctos.
Resultado	Nos devuelve los insectos del usuario.
Éxito	Si.

Caso de prueba	CP-019
Descripción	Cat-insectos-> No debería leer por usuario incorrecto.
Acciones	Se accede a Cat-insectos mediante read con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-020
Descripción	Cat-insectos-> No debería leer por verificación incorrecta.
Acciones	Se accede a Cat-insectos mediante read con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-021
Descripción	Cat-insectos-> Debería crear.
Acciones	Se accede a Cat-insectos mediante create con datos correctos.
Resultado	Crea el insecto para el usuario.
Éxito	Si.

Caso de prueba	CP-022
Descripción	Cat-insectos-> No debería crear por usuario incorrecto.
Acciones	Se accede a Cat-insectos mediante create con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-023
Descripción	Cat-insectos-> No debería crear para otro usuario.
Acciones	Se accede a Cat-insectos mediante create con un usuario distinto al del insecto.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-024
Descripción	Cat-insectos-> No debería crear de nuevo el mismo insecto.
Acciones	Se accede a Cat-insectos mediante create con el mismo insecto que se creó correctamente antes.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-025
Descripción	Cat-insectos-> No debería crear insecto vacío.
Acciones	Se accede a Cat-insectos mediante create con un insecto vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-026
Descripción	Cat-insectos-> No debería crear por verificación incorrecta.
Acciones	Se accede a Cat-insectos mediante create con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-027
Descripción	Cat-insectos-> Debería borrar.
Acciones	Se accede a Cat-insectos mediante delete con datos correctos.
Resultado	Elimina el insecto para el usuario.
Éxito	Si.

Caso de prueba	CP-028
Descripción	Cat-insectos-> No debería eliminar por usuario incorrecto.
Acciones	Se accede a Cat-insectos mediante delete con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-029
Descripción	Cat-insectos-> No debería eliminar un insecto inexistente.
Acciones	Se accede a Cat-insectos mediante delete con un insecto que no existe.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-030
Descripción	Cat-insectos-> No debería eliminar un insecto vacío.
Acciones	Se accede a Cat-insectos mediante delete con un insecto vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-031
Descripción	Cat-insectos-> No debería eliminar por verificación incorrecta.
Acciones	Se accede a Cat-insectos mediante delete con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-032
Descripción	Cat-sueño-> Debería leer.
Acciones	Se accede a Cat-sueño mediante read con datos correctos.
Resultado	Nos devuelve los sueños existentes.
Éxito	Si.

Caso de prueba	CP-033
Descripción	Cat-sueño> Debería leer mis sueños.
Acciones	Se accede a Cat-sueño mediante readMine con datos correctos.
Resultado	Nos devuelve el sueño del usuario.
Éxito	Si.

Caso de prueba	CP-034
Descripción	Cat-sueño> No debería leer mis sueños por usuario incorrecto.
Acciones	Se accede a Cat-sueño mediante readMine con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-035
Descripción	Cat-sueño> No debería leer mis sueños por verificación incorrecta.
Acciones	Se accede a Cat-sueño mediante readMine con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-036
Descripción	Cat-sueño> Debería comprobar si existe.
Acciones	Se accede a Cat-sueño mediante exist con datos correctos.
Resultado	Nos devuelve si existe o no.
Éxito	Si.

Caso de prueba	CP-037
Descripción	Cat-sueño> No debería comprobar si existe por usuario incorrecto.
Acciones	Se accede a Cat-sueño mediante exist con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-038
Descripción	Cat-sueño> No debería comprobar si existe por verificación incorrecta.
Acciones	Se accede a Cat-sueño mediante exist con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-039
Descripción	Cat-sueños-> Debería crear.
Acciones	Se accede a Cat-sueños mediante create con datos correctos.
Resultado	Crea el sueño para el usuario.
Éxito	Si.

Caso de prueba	CP-040
Descripción	Cat-sueños-> No debería crear por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante create con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-041
Descripción	Cat-sueños-> No debería crear para otro usuario.
Acciones	Se accede a Cat-sueños mediante create con un usuario distinto al del sueño.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-042
Descripción	Cat-sueños-> No debería crear sueño con código de sueño incorrecto.
Acciones	Se accede a Cat-sueños mediante create con un código de sueño que no sigue el patrón.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-043
Descripción	Cat-sueños-> No debería crear con código de sueño vacío.
Acciones	Se accede a Cat-sueños mediante create con un código de sueño vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-044
Descripción	Cat-sueños-> No debería crear sin foto.
Acciones	Se accede a Cat-sueños mediante create sin foto.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-045
Descripción	Cat-sueños-> No debería crear por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante create con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-046
Descripción	Cat-sueños-> Debería actualizar.
Acciones	Se accede a Cat-sueños mediante update con datos correctos.
Resultado	Crea el sueño para el usuario.
Éxito	Si.

Caso de prueba	CP-047
Descripción	Cat-sueños-> No debería actualizar por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante update con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-048
Descripción	Cat-sueños-> No debería actualizar para otro usuario.
Acciones	Se accede a Cat-sueños mediante update con un usuario distinto al del sueño.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-049
Descripción	Cat-sueños-> No debería actualizar si el usuario no posee sueño.
Acciones	Se accede a Cat-sueños mediante update con un usuario que no posee ningún sueño.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-050
Descripción	Cat-sueños-> No debería actualizar sueño con código de sueño incorrecto.
Acciones	Se accede a Cat-sueños mediante update con un código de sueño que no sigue el patrón.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-051
Descripción	Cat-sueños-> No debería actualizar con código de sueño vacío.
Acciones	Se accede a Cat-sueños mediante update con un código de sueño vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-052
Descripción	Cat-sueños-> No debería actualizar sin foto.
Acciones	Se accede a Cat-sueños mediante update sin foto.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-053
Descripción	Cat-sueños-> No debería actualizar por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante update con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-054
Descripción	Cat-sueños-> Debería borrar.
Acciones	Se accede a Cat-sueños mediante delete con datos correctos.
Resultado	Elimina el sueño del usuario.
Éxito	Si.

Caso de prueba	CP-055
Descripción	Cat-sueños-> No debería eliminar por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante delete con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-056
Descripción	Cat-sueños-> No debería eliminar por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante delete con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-057
Descripción	Cat-sueños-> Debería leer mis likes.
Acciones	Se accede a Cat-sueños mediante readMisLikes con datos correctos.
Resultado	Nos devuelve los sueños a los que el usuario ha dado like.
Éxito	Si.

Caso de prueba	CP-058
Descripción	Cat-sueños-> No debería leer mis likes por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante readMisLikes con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-059
Descripción	Cat-sueños-> No debería leer mis likes por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante readMisLikes con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-060
Descripción	Cat-sueños-> Debería crear like.
Acciones	Se accede a Cat-sueños mediante creaLike con datos correctos.
Resultado	Crea el like.
Éxito	Si.

Caso de prueba	CP-061
Descripción	Cat-sueños-> No debería crear like por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante creaLike con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-062
Descripción	Cat-sueños-> No debería crear like por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante creaLike con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-063
Descripción	Cat-sueños-> No debería crear like por objeto inexistente.
Acciones	Se accede a Cat-sueños mediante creaLike con un código de sueño inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-064
Descripción	Cat-sueños-> Debería eliminar like.
Acciones	Se accede a Cat-sueños mediante deleteLike con datos correctos.
Resultado	Elimina el like.
Éxito	Si.

Caso de prueba	CP-065
Descripción	Cat-sueños-> No debería eliminar like por usuario incorrecto.
Acciones	Se accede a Cat-sueños mediante deleteLike con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-066
Descripción	Cat-sueños-> No debería eliminar like por verificación incorrecta.
Acciones	Se accede a Cat-sueños mediante deleteLike con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-067
Descripción	Cat-sueños-> No debería eliminar like por objeto inexistente.
Acciones	Se accede a Cat-sueños mediante deleteLike con un código de sueño inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-068
Descripción	Cat-vecinos-> Debería crear component.
Acciones	Se crea una instancia del componente Cat-vecinos.
Resultado	Se crea la instancia correctamente.
Éxito	Si.

Caso de prueba	CP-069
Descripción	Cat-vecinos-> Debería filtrar por género.
Acciones	Se cambia el filtro.
Resultado	El filtro queda cambiado.
Éxito	Si.

Caso de prueba	CP-070
Descripción	Cat-vecinos-> Debería filtrar por personalidad.
Acciones	Se cambia el filtro.
Resultado	El filtro queda cambiado.
Éxito	Si.

Caso de prueba	CP-071
Descripción	Cat-vecinos-> Debería tener lista.
Acciones	Se crea una instancia del component.
Resultado	La instancia posee una lista.
Éxito	Si.

Caso de prueba	CP-072
Descripción	Cat-vecinos-> Debería convertir fecha.
Acciones	Se llama al método mesToString con un fecha correcta.
Resultado	Convierte la fecha introducida al formato deseado.
Éxito	Si.

Caso de prueba	CP-073
Descripción	Filtra-vecinos-> Debería crear filtro.
Acciones	Se crea una instancia del filtro Filtra-vecinos.
Resultado	Se crea la instancia correctamente.
Éxito	Si.

Caso de prueba	CP-074
Descripción	Filtra-vecinos-> Debería filtrar por especie.
Acciones	Se cambia el filtro.
Resultado	El filtro queda cambiado.
Éxito	Si.

Caso de prueba	CP-075
Descripción	Filtra-vecinos-> Debería filtrar por personalidad.
Acciones	Se cambia el filtro.
Resultado	El filtro queda cambiado.
Éxito	Si.

Caso de prueba	CP-076
Descripción	Filtra-vecinos-> Debería filtrar por género.
Acciones	Se cambia el filtro.
Resultado	El filtro queda cambiado.
Éxito	Si.

Caso de prueba	CP-077
Descripción	Tarea-> Debería leer.
Acciones	Se accede a Tarea mediante read con datos correctos.
Resultado	Nos devuelve las tareas del usuario.
Éxito	Si.

Caso de prueba	CP-078
Descripción	Tarea-> No debería leer por usuario incorrecto.
Acciones	Se accede a Tarea mediante read con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-079
Descripción	Tarea-> No debería leer por verificación incorrecta.
Acciones	Se accede a Tarea mediante read con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-080
Descripción	Tarea-> Debería crear.
Acciones	Se accede a Tarea mediante create con datos correctos.
Resultado	Crea la tarea para el usuario.
Éxito	Si.

Caso de prueba	CP-081
Descripción	Tarea-> No debería crear por usuario incorrecto.
Acciones	Se accede a Tarea mediante create con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-082
Descripción	Tarea-> No debería crear para otro usuario.
Acciones	Se accede a Tarea mediante create con un usuario distinto al de la tarea.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-083
Descripción	Tarea-> No debería crear tarea vacía.
Acciones	Se accede a Tarea mediante create con un insecto vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-084
Descripción	Tarea-> No debería crear por verificación incorrecta.
Acciones	Se accede a Cat-insectos mediante create con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-085
Descripción	Tarea-> Debería borrar.
Acciones	Se accede a Tarea mediante delete con datos correctos.
Resultado	Elimina la tarea del usuario.
Éxito	Si.

Caso de prueba	CP-086
Descripción	Tarea-> No debería eliminar por usuario incorrecto.
Acciones	Se accede a Tarea mediante delete con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-087
Descripción	Tarea-> No debería eliminar por verificación incorrecta.
Acciones	Se accede a Tarea mediante delete con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-088
Descripción	Tarea-> No debería eliminar por tarea incorrecta.
Acciones	Se accede a Tarea mediante delete con una tarea inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-089
Descripción	Tarea-> No debería eliminar por tarea vacía.
Acciones	Se accede a Tarea mediante delete con una tarea vacía.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-090
Descripción	Tarea-> Debería actualizar.
Acciones	Se accede a Tarea mediante update con datos correctos.
Resultado	Actualiza las tareas del usuario.
Éxito	Si.

Caso de prueba	CP-091
Descripción	Tarea-> No debería actualizar por usuario incorrecto.
Acciones	Se accede a Tarea mediante update con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-092
Descripción	Tarea-> No debería actualizar para otro usuario.
Acciones	Se accede a Tarea mediante update con un usuario distinto al de la tarea.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-093
Descripción	Tarea-> No debería actualizar con una tarea vacía.
Acciones	Se accede a Tarea mediante update con una tarea vacía.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-094
Descripción	Tarea-> No debería actualizar por verificación incorrecta.
Acciones	Se accede a Tarea mediante update con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-095
Descripción	Visitas-> Debería crear.
Acciones	Se accede a Visita mediante create con datos correctos.
Resultado	Crea la visita vacía para el usuario.
Éxito	Si.

Caso de prueba	CP-096
Descripción	Visitas-> No debería crear de nuevo con el mismo usuario.
Acciones	Se accede a Visita mediante create con un usuario que ya posea visita.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-097
Descripción	Visitas-> No debería crear por usuario incorrecto.
Acciones	Se accede a Visita mediante create con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-098
Descripción	Visitas-> No debería crear por verificación incorrecta.
Acciones	Se accede a Visita mediante create con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-099
Descripción	Visitas-> Debería leer.
Acciones	Se accede a Visita mediante read con datos correctos.
Resultado	Nos devuelve las visitas del usuario.
Éxito	Si.

Caso de prueba	CP-100
Descripción	Visitas-> No debería leer por usuario incorrecto.
Acciones	Se accede a Visita mediante read con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-101
Descripción	Visitas-> No debería leer por verificación incorrecta.
Acciones	Se accede a Visita mediante read con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-102
Descripción	Visitas-> Debería actualizar.
Acciones	Se accede a Visita mediante update con datos correctos.
Resultado	Actualiza las visitas del usuario.
Éxito	Si.

Caso de prueba	CP-103
Descripción	Visitas-> No debería actualizar por usuario incorrecto.
Acciones	Se accede a Visita mediante update con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-104
Descripción	Visitas-> No debería actualizar para otro usuario.
Acciones	Se accede a Visita mediante update con un usuario distinto al de la tarea.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-105
Descripción	Visitas-> No debería actualizar con campo estela vacío.
Acciones	Se accede a Visita mediante update con campo estela vacío.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-106
Descripción	Visitas-> No debería actualizar con campo estela incorrecto.
Acciones	Se accede a Visita mediante update con campo estela incorrecto.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-107
Descripción	Visitas-> No debería actualizar por verificación incorrecta.
Acciones	Se accede a Visita mediante update con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-108
Descripción	Visitas-> Debería actualizar fecha.
Acciones	Se accede a Visita mediante set_fecha con datos correctos.
Resultado	Actualiza las fecha de visitas del usuario.
Éxito	Si.

Caso de prueba	CP-109
Descripción	Visitas-> No debería actualizar fecha por usuario incorrecto.
Acciones	Se accede a Visita mediante set_fecha con un usuario inexistente.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-110
Descripción	Visitas-> No debería actualizar fecha para otro usuario.
Acciones	Se accede a Visita mediante set_fecha con un usuario distinto al de la tarea.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

Caso de prueba	CP-111
Descripción	Visitas-> No debería actualizar fecha por verificación incorrecta.
Acciones	Se accede a Visita mediante set_fecha con una verificación incorrecta.
Resultado	Nos devuelve un mensaje de error.
Éxito	Si.

CAPÍTULO 8

Comparación con otras alternativas

El principal objetivo de nuestra aplicación era reunir y centralizar el mayor número de herramientas útiles posibles para aquellas personas que jugasen a “Animal Crossing: New Horizons”, especialmente aquellas que te permitan llevar un progreso de tu día a día en el juego. Además, buscábamos adaptar las aplicaciones móvil existentes a ordenador, ya que contienen herramientas bastante útiles que no están disponibles en la mayoría de aplicaciones web.

Por lo tanto, vamos a comparar las herramientas disponibles en una serie de alternativas con las disponibles en nuestra aplicación (véase la figura 8.1).

Herramienta	ACNH Life	NookNet	Nook's Island	ACNH Guide	Nook Plaza	MyCrossing
Catálogo museo (peces, insectos, crit. marinas, fósiles y obras de arte)	✓	✗	✓	✓	✓	✓
Catálogo vecinos	✓	✓	✓	✓	✓	✓
Catálogo canciones + reproductor	✗	✗	✗	✗	✗	✓
Catálogo items (muebles y ropa)	✓	✗	✓	✓	✓	✓
Calendario eventos	✓	✓	✗	✓	✗	✓
Catálogo sueños	✗	✓	✓	✓	✗	✓
Criaturas disponibles según fecha y hora	✓	✗	✗	✓	✗	✓
Calculadora nabos	✓	✓	✗	✓	✗	✓
Tareas diarias	✓	✗	✗	✓	✗	✓
Visitantes semanales	✓	✗	✗	✓	✗	✓
Album de fotos	✗	✗	✗	✗	✗	✓
Vecinos de la isla	✓	✓	✗	✓	✗	✓
Probabilidad mudanza vecinos	✗	✗	✗	✗	✓	✓
Seguimiento colecciones	✓	✗	✗	✓	✗	✓

Figura 8.1: Comparación alternativas

Como podemos observar, nuestros mayores dos competidores son ACNH Life y ACNH Guide, ambos aplicaciones de móvil que cuentan con las herramientas de tareas diarias y visitantes semanales entre otras. Sin embargo, nuestra aplicación destaca por dos factores principales: el catálogo de música con posibilidad de reproducir las canciones y el álbum de fotos personal.

Respecto a los otros competidores, vemos como destacamos además con el catálogo de criaturas disponibles según fecha y hora, el seguimiento de colecciones, las tareas diarias y los visitantes semanales y especialmente con el cálculo de la probabilidad de mudanza de los vecinos, siendo nuestro único competidor (de los estudiados) NookPlaza, por lo que también es un factor importante a destacar.

En conclusión, nuestra aplicación no solo destaca con varias herramientas no disponibles en las demás alternativas sino que además recopila varias de las herramientas ya existentes, tanto en móvil como en PC, de forma que le ofrece al usuario una gama de herramientas e información más amplia todo desde la misma aplicación, de forma que no tenga que usar varias aplicaciones (y sobre todo, en diferentes plataformas) para lograr el mismo resultado.

CAPÍTULO 9

Manual de Usuario

Al acceder a la dirección <https://mycrossing.herokuapp.com/> nos encontraremos con la siguiente pantalla:

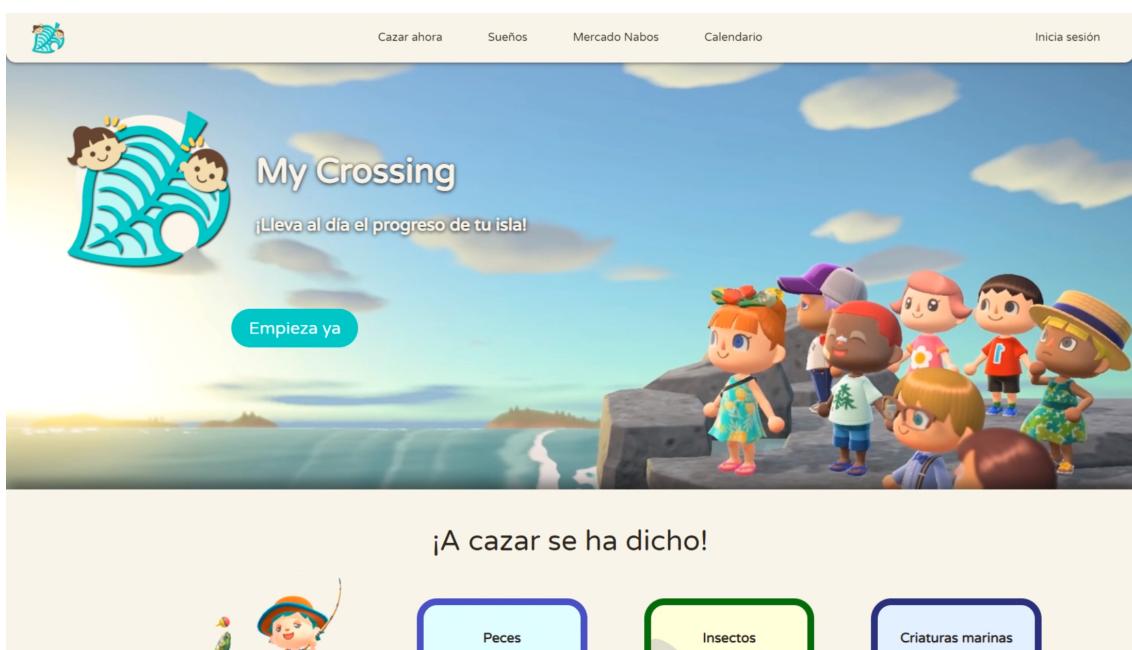


Figura 9.1: Inicio

Aunque muchas de las funciones de la aplicación son accesibles sin necesidad de disponer de una cuenta en la misma, primero procederemos al registro para poder así ver la funcionalidad completa de la aplicación.

Para registrarnos, hacemos clic en el botón de "Empieza ya", (véase la figura 9.1) lo que nos llevará a la pantalla de registro.

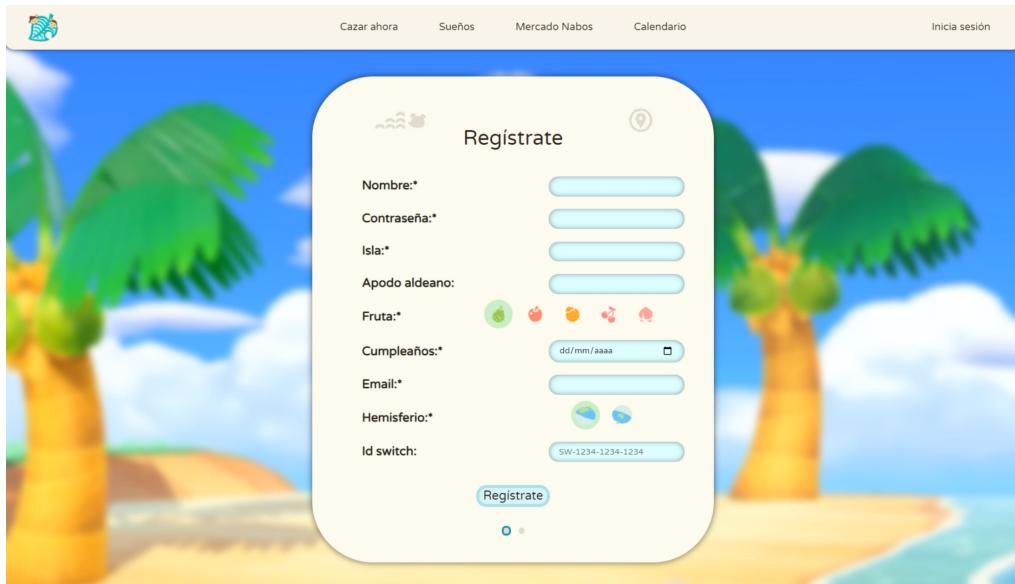


Figura 9.2: Registro

Una vez rellenamos los datos de manera correcta, hacemos clic en el botón “Regístrate”,(véase la figura 9.2) y si todo ha ido bien, nos devolverá a la página de inicio, pero esta vez con nuestra cuenta creada y la sesión iniciada(véase la figura 9.3).



Figura 9.3: Inicio con sesión iniciada

Desde el inicio tenemos acceso a la mayoría de catálogos de la aplicación. Si nos vamos, por ejemplo, al catálogo de peces, veremos la siguiente pantalla.(véase la figura 9.4)

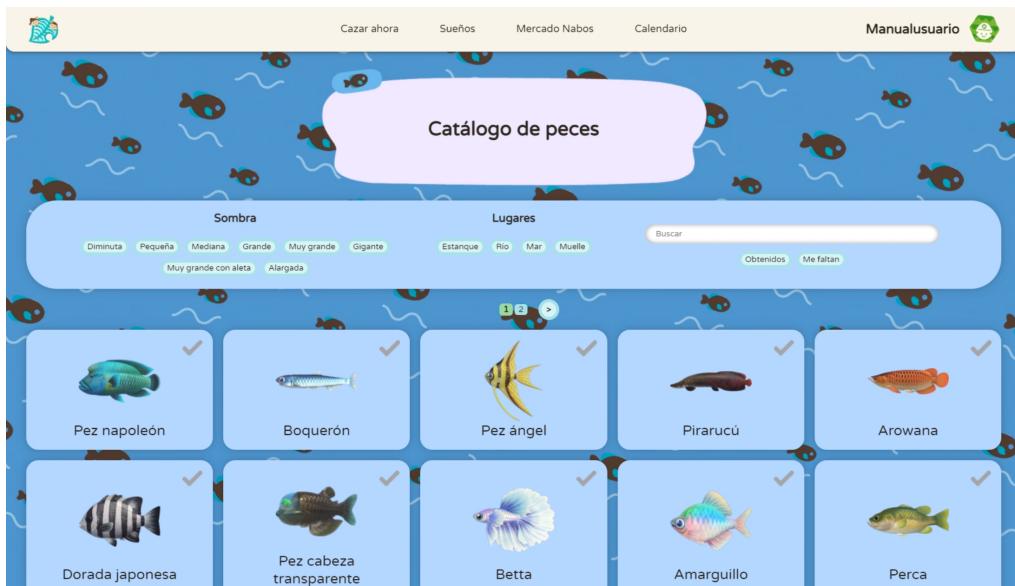


Figura 9.4: Catálogo peces

Aquí, al igual que en todos los catálogos, disponemos de unos filtros para poder realizar una búsqueda algo más precisa. En el caso de los peces, podemos filtrar por el tamaño de su sombra, por los lugares en los que podemos encontrarlos y por su nombre. En el caso de haber iniciado sesión podemos filtrar por aquellos peces que hemos obtenido ya o no.

En la gran mayoría de catálogos, hay un botón de check para que el usuario pueda ir indicando si ha obtenido el pez o no (véase la figura 9.5). De esta forma puede llevar un recuento de aquellos peces que ya ha obtenido para ir completando su colección.



Figura 9.5: Peces obtenidos

Además, podemos hacer clic en las fotos de los peces para obtener información sobre los mismos, como el lugar y la hora donde se puede obtener, así como otros datos. Adicionalmente, podemos ver los meses en los que el pez está disponible tanto para el hemisferio norte como para el hemisferio sur si hacemos clic en el botón superior izquierda (véanse las figuras 9.6 y 9.7).



Figura 9.6: Info hemisferio norte



Figura 9.7: Info hemisferio sur

Además, tenemos la opción de navegar por las distintas páginas de los resultados, bien avanzando o retrocediendo de uno en uno con los botones de las flechas, o bien haciendo clic sobre la página deseada (véase la figura 9.8).

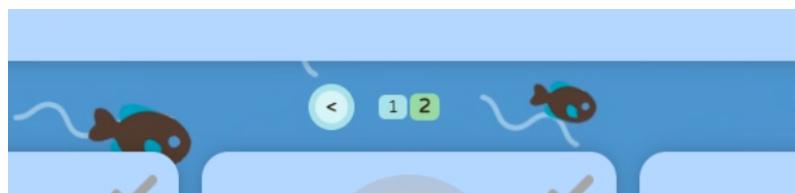


Figura 9.8: Paginación

De la misma forma, la gran mayoría de catálogos funcionan de manera idéntica con ciertas diferencias, como obviamente, la información que se muestra al hacer clic en un ítem. Hay otros cambios en algunos catálogos, como en el de obras de arte, en el cual podemos observar que hay ciertas piezas con un botón, que indica que esa obra tiene una falsificación (véase la figura 9.9).



Figura 9.9: Catálogo obras de arte

Si hacemos clic en ese botón, podemos ver un menú de información con las imágenes, tanto de la versión real como de la falsificación, para intentar ayudar al usuario a diferenciarlas (véase la figura 9.10).



Figura 9.10: Falsificación

Tanto en el catálogo de muebles como en el de ropa, existen ítems con variaciones (véase la figura 9.11). Estas variaciones podemos verlas desplazándonos con una barra de desplazamiento y si hacemos clic, veremos que cambia la imagen principal del ítem por el de la variante seleccionada (véase la figura 9.12).



Figura 9.11: Variaciones



Figura 9.12: Variación

En el catálogo de canciones, ademas de poder llevar recuento de aquellas canciones que hemos obtenido, podemos escucharlas. Si hacemos clic en la imagen de la foto, esta empezará a reproducirse, y además disponemos de algunos controles en el reproductor abajo a la izquierda, tanto para pausar/continuar la canción como para ajustar el volumen (véase la figura 9.13).

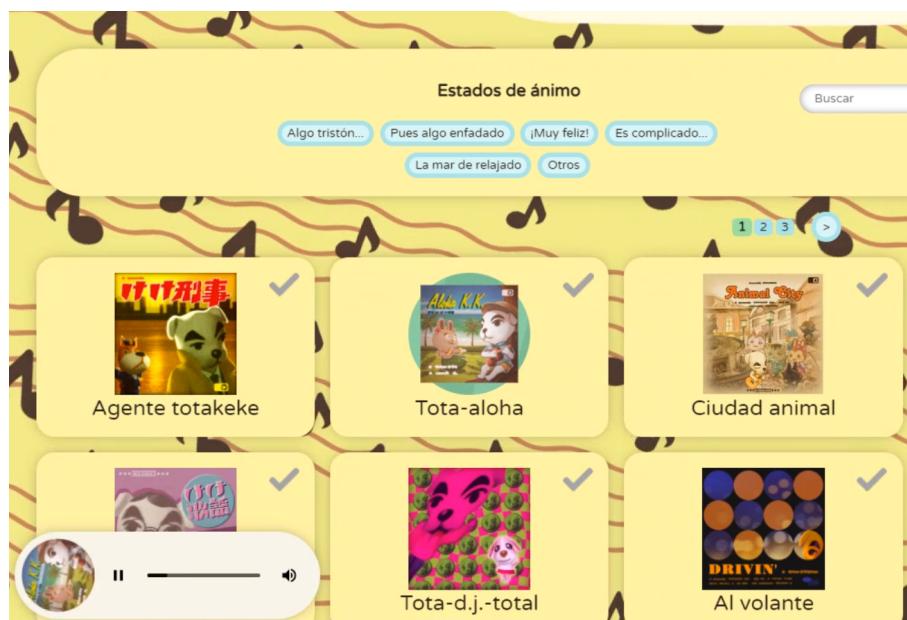


Figura 9.13: Catálogo canciones

Una vez terminado con los catálogos de la página de inicio, tenemos algunas otras herramientas accesibles desde la barra superior de navegación (véase la figura 9.14).



Figura 9.14: Barra de navegación

Si hacemos clic en la imagen de la izquierda, nos llevará a la página de inicio. Luego en la parte central disponemos de cuatro botones. El primero, “Cazar ahora”, nos lleva a un catálogo de los tres tipos de criaturas obtenibles, con la peculiaridad de que solo nos muestran aquellas criaturas capturables en la fecha y hora actuales (véase la figura 9.15).



Figura 9.15: Cazar ahora

Aquí, no solo disponemos de los filtros que tienen en su catálogo correspondiente, sino que además podemos cambiar el mes y hora, así como el hemisferio (de manera global) para poder ver aquellas criaturas en función del mes y hora, de forma que le sea más sencillo al usuario y no tenga que dar vueltas por la isla sin saber qué o dónde se puede encontrar. Si hemos modificado el mes y/o la hora y deseamos volver a la actual, es tan fácil como hacer clic en el botón de la flecha roja (véase la figura 9.16).



Figura 9.16: Filtros

Dentro de este catálogo, de base apareceremos en la pantalla de los insectos, por lo que si queremos ver los peces o las criaturas marinas es tan fácil como hacer clic en el botón correspondiente a la derecha de los filtros.

Volviendo a la barra de navegación superior, el siguiente botón sería el de “Sueños”, pero para explicar por completo su funcionalidad necesitamos antes pasar por el perfil, por lo que se hablará de este catálogo más adelante. Por lo que primeroaremos clic en el botón que pone “Mercado Nabos”, que nos llevará a la siguiente pantalla (véase la figura 9.17).

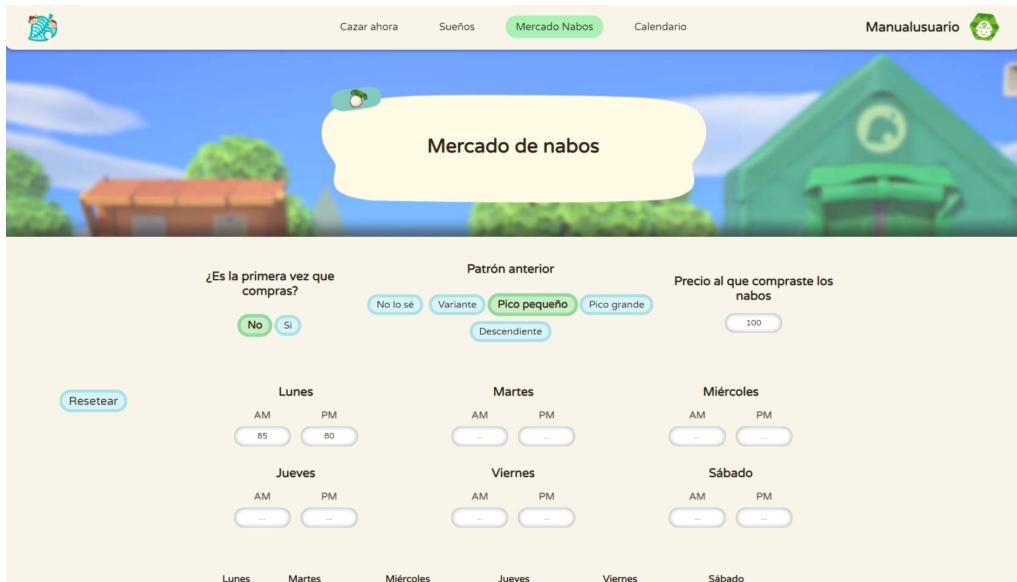


Figura 9.17: Mercado nabos

En esta página podemos predecir tanto el precio de los nabos en la isla esta semana, así como el patrón que seguirá, y los posibles máximos y mínimos, todo para intentar sacar el mayor beneficio posible.

Para empezar, debemos llenar tantos campos como podamos. El primero indica si es la primera vez que el usuario compra nabos o no, ya que la primera vez siempre se obtiene el mismo patrón. El segundo botón indica el patrón de la semana anterior, que también dispone de un botón en caso de que el usuario no lo sepa. Por último, hay que indicar el precio al que se compraron los nabos el domingo (véase la figura 9.18).



Figura 9.18: Datos

A partir de aquí, hay que indicar el precio al que Tendo y Nendo compran los nabos, tanto por la mañana como por la tarde. A mayor datos se introduzcan, mas precisa será la predicción. Como podemos ver, indicando los tres primeros campos, así como los precios del lunes, ya obtenemos una predicción bastante segura (véase la figura 9.19).

Patrón	Probabilidad(%)	Lunes		Martes		Miércoles		Jueves		Viernes		Sábado		Mínimo garantizado	Máximo potencial
		AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM		
Todos	—	—	85 80 74 a 140	69 a 200	64 a 600	59 a 600	54 a 600	40 a 600	35 a 600	30 a 600	25 a 200	20 a 199	75	600	
Pico pequeño	16.7%	85 80 90 a 140	90 a 140	139 a 199	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	25 a 81	20 a 78	139	200	
Pico pequeño	16.7%	85 80 74 a 78	90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	25 a 81	30 a 84	139	200	
Pico pequeño	16.7%	85 80 74 a 78	69 a 75	64 a 72	90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	139	200	
Pico pequeño	16.7%	85 80 74 a 78	69 a 75	64 a 72	59 a 69	90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	139	200
Pico pequeño	16.7%	85 80 74 a 78	69 a 75	64 a 72	59 a 69	90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	139	200
Pico grande	<0.01%	85 80 90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	40 a 90	40 a 90	40 a 90	40 a 90	40 a 90	200	600	
Pico grande	<0.01%	85 80 75 a 78	90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	40 a 90	40 a 90	40 a 90	40 a 90	200	600	
Pico grande	<0.01%	85 80 75 a 78	70 a 75	65 a 72	90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	40 a 90	40 a 90	200	600	
Pico grande	<0.01%	85 80 75 a 78	70 a 75	65 a 72	59 a 69	90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	40 a 90	200	600	
Pico grande	<0.01%	85 80 75 a 78	70 a 75	65 a 72	60 a 69	55 a 66	90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	200	600	
Descendiente	<0.01%	85 80 75 a 78	70 a 75	65 a 72	60 a 69	55 a 66	90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	200	600	

Figura 9.19: Tabla predicción

Como podemos observar, se nos indica una fila por cada posible “ruta” que puede suceder esa semana. A la izquierda vemos el patrón que podemos obtener, seguido de dos porcentajes que indican, el primero la probabilidad del patrón, y el segundo la probabilidad de la “ruta” dentro del patrón. Luego observamos los precios, tanto los indicados por el usuario como los posibles para el resto de la semana. Los distintos colores representan el beneficio que puede representar, siendo rojo el menor beneficio y el verde oscuro el mayor beneficio. Por último, las dos últimas columnas nos indican tanto el mínimo garantizado como el máximo potencial al que se puede llegar.

Por ejemplo, si nos fijamos en la tabla, vemos que hay un 100 % de probabilidades de que esta semana obtengamos un patrón “Pico pequeño”, dentro del cual, a falta de más datos, tenemos seis posibles “rutas” las cuales tienen todas la misma probabilidad de ocurrir. Sin embargo, todas tienen el mismo mínimo y máximo, siendo 139 y 200 bayas respectivamente.

Patrón	Probabilidad(%)	Lunes		Martes		Miércoles		Jueves		Viernes		Sábado		Mínimo garantizado	Máximo potencial
		AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM		
Todos	—	—	85 80 75 70 72 70	65 56 a 140	51 a 200	46 a 600	41 a 200	36 a 199	56	600					
Pico pequeño	100%	85 80 75 70 72 70	65 90 a 140	90 a 140	139 a 199	139 a 200	139 a 199	40 a 90	35 a 87	30 a 84	25 a 81	139	200		
Descendiente	<0.01%	85 80 75 70 72 70	65 56 a 59	51 a 56	46 a 53	41 a 50	36 a 47	56	59	59	59	56	59		
Pico grande	<0.01%	85 80 75 70 72 70	65 90 a 140	140 a 200	200 a 600	140 a 200	90 a 140	40 a 90	35 a 87	30 a 84	25 a 81	200	600		

Figura 9.20: Predicción precisa

Si rellenamos más datos a lo largo de la semana, vemos que se van reduciendo el número de filas ya que la predicción es mas precisa. Una vez llenado los datos hasta la mitad del jueves, observamos que efectivamente, el patrón de esta semana es el “Pico pequeño”, que el mínimo y máximo son los comentados anteriormente y que, el beneficio lo obtendremos entre la segunda mitad del jueves al sábado (véase la figura 9.20).

Obviamente, al ser datos ficticios no representa al 100% la precisión de la herramienta, pero con datos reales obtenidos del juego que resulten mas coherentes con el algoritmo establecido dentro del juego, se pueden obtener resultados bastante precisos.

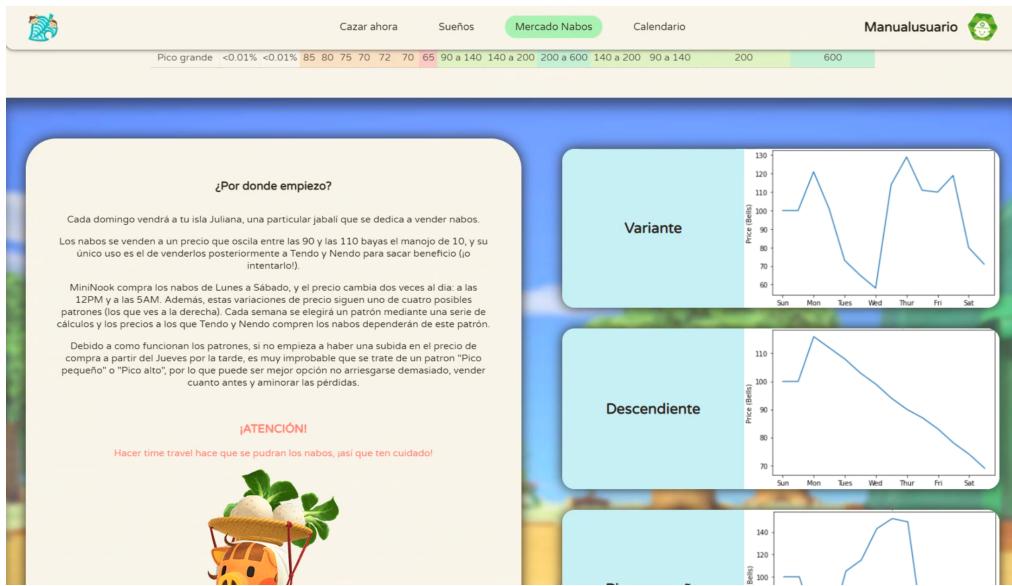


Figura 9.21: Información mercado

Si bajamos un poco, veremos que el usuario dispone de información para aprender cómo funciona el mercado de nabos dentro del juego, así como los posibles patrones que existen (véase la figura 9.21). Si coloca el cursor encima de cualquiera de las tarjetas de los patrones, podrá ver que aparece una breve descripción del funcionamiento del patrón en cuestión (véase la figura 9.22). De esta forma, el usuario puede acceder a esta información para intentar llenar los datos de la manera más precisa posible, obteniendo así un mejor resultado.



Figura 9.22: Información patrones

La siguiente parada dentro de la barra de navegación superior sería el “Calendario”, el cual nos llevará a la siguiente pantalla al hacer clic en él.

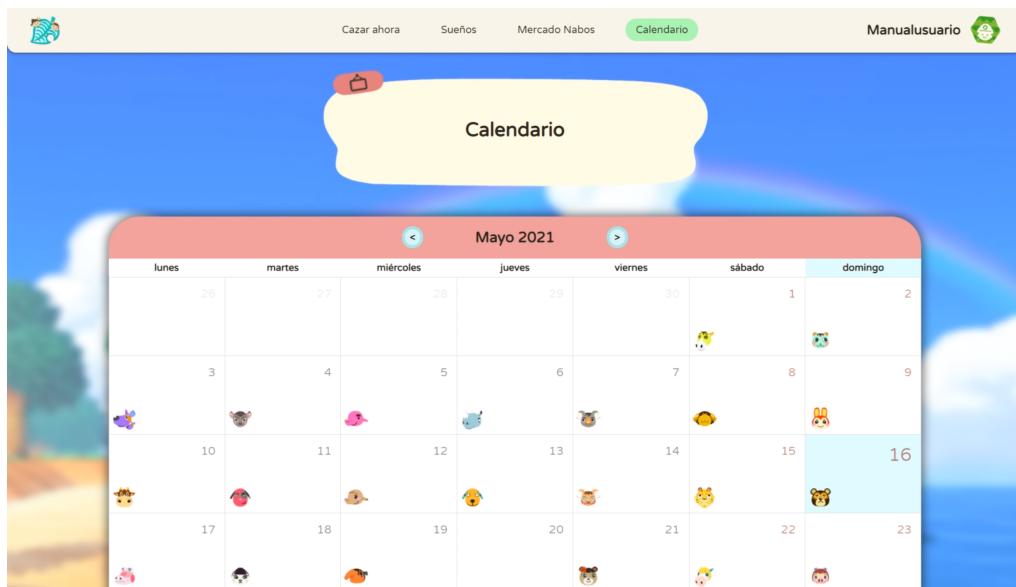


Figura 9.23: Calendario

En esta pantalla disponemos de un calendario por el que podemos navegar, en el cuál se nos indican los eventos disponibles, siendo estos los cumpleaños de los distintos vecinos, así como otros eventos más especiales (véase la figura 9.23). En azul claro se resalta el día en el que nos encontramos, y si colocamos el cursor sobre cualquiera de los iconos situados en la parte inferior izquierda de cada día, podremos ver el título del evento. Si se trata de un cumpleaños, la celda obtendrá un color verde (véase la figura 9.24), mientras que si se trata de otro tipo de evento, tendrá un color distinto.

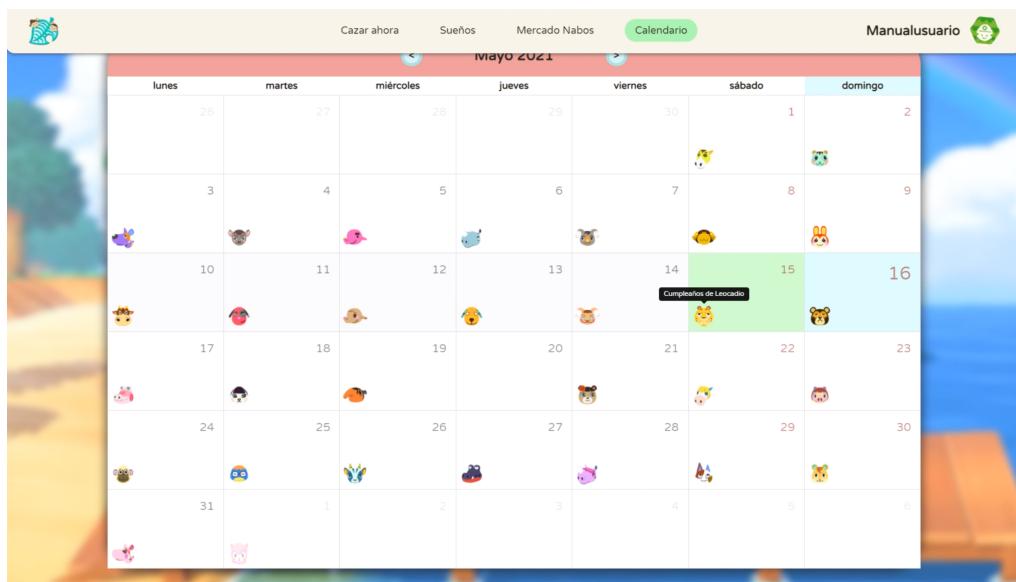


Figura 9.24: Cumpleaños

Si por ejemplo nos vamos al 31 de octubre, veremos que el ícono corresponde a uno de los NPCs del juego, y al colocar el cursor sobre el, no solo nos indica que se trata de Halloween, sino que la celda obtiene un color anaranjado (véase la figura 9.25).

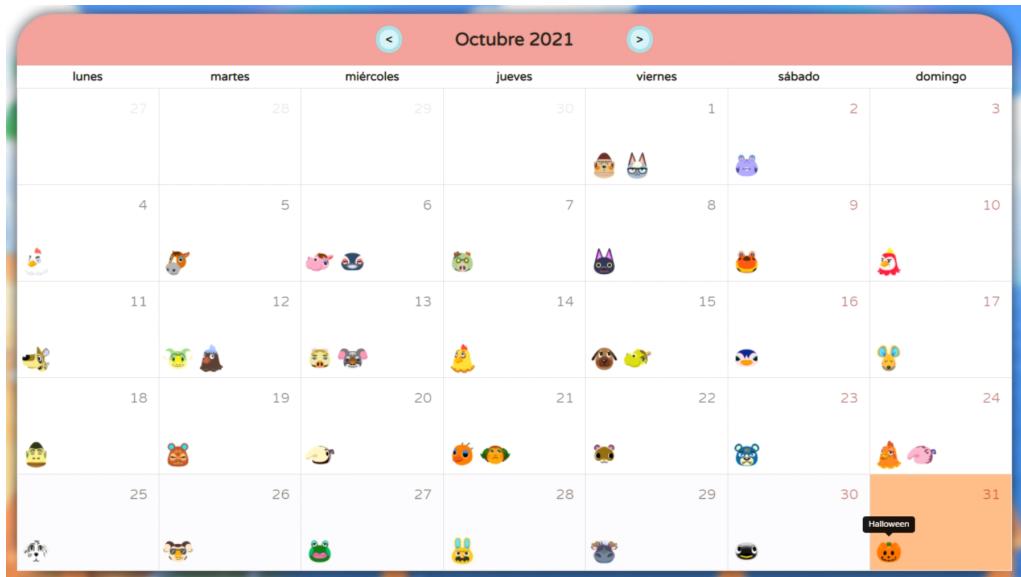


Figura 9.25: Evento

Si en este caso (tratándose de un evento especial), hacemos clic en el ícono, se nos abrirá un menú con información del evento (véase la figura 9.26).



Figura 9.26: Halloween

A continuación, como podemos ver en la barra de navegación, a la derecha del todo tenemos nuestro nombre de usuario seguido de una foto. Si hacemos clic en la foto se nos abrirá un menú desplegable que nos da la opción de, o ir a nuestro perfil, o cerrar sesión (véase la figura 9.27).



Figura 9.27: Desplegable perfil

En este caso haremos clic en el botón “Perfil”, que nos llevará tal y como indica su nombre, a nuestro perfil (véase la figura 9.28).

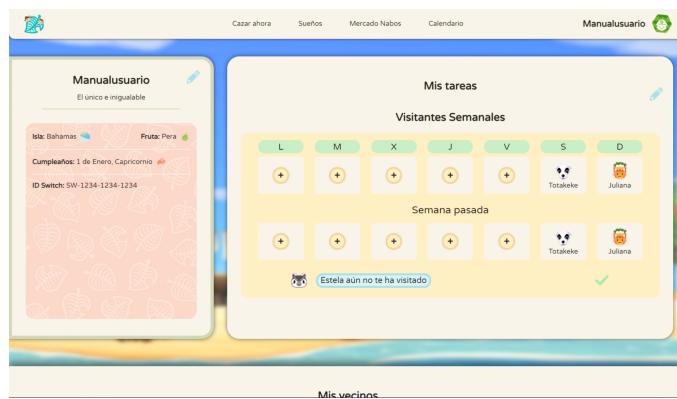


Figura 9.28: Perfil

Observemos primero el menú de la izquierda. Representa el pasaporte del usuario, donde podremos ver la información proporcionada en el registro, así como el signo del zodiaco al que pertenece. Si el usuario quisiera modificar los datos, sería tan sencillo como hacer clic en el lápiz azul del pasaporte, lo que nos llevaría al modo edición del mismo (véase la figura 9.29).

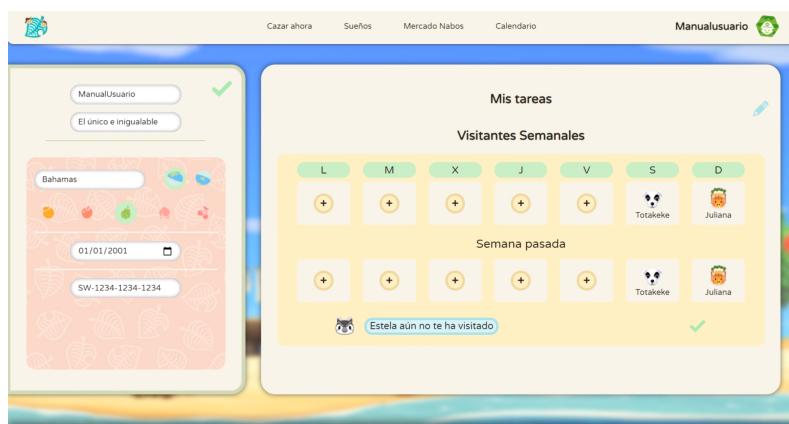


Figura 9.29: Editar perfil

Una vez en dicho modo edición, podemos cambiar los datos a nuestro antojo, siempre y cuando cumplan los criterios de validación. Una vez listo, hacemos clic en el botón del check verde para confirmar la operación. En caso de querer cancelar la operación, es tan fácil como volver a dejar los datos como estaban y hacer clic en el check verde.

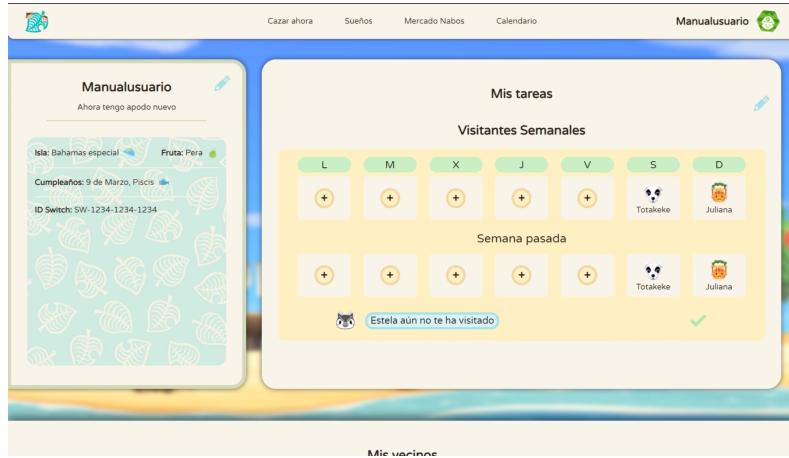


Figura 9.30: Perfil actualizado

Como podemos ver, se ha actualizado la información y además se ha cambiado el color de fondo del pasaporte (véase la figura 9.30). El fondo varía de forma aleatoria tanto al acceder al perfil como al actualizar la información.

Ahora fijémonos en el menú que se encuentra a la derecha. Ahí podemos encontrar dos de las herramientas más importantes y útiles para el usuario. En primer lugar, se trata de las tareas personales. Al acceder al perfil por primera vez se encontrará vacío, como es nuestro caso. Sin embargo, haciendo clic en el lápiz que encontramos a la derecha de dicho menú, entramos en el modo edición del mismo.



Figura 9.31: Modo edición

Como podemos observar, ha aparecido un botón amarillo con un '+', (véase la figura 9.31). Si hacemos clic en ese botón, se nos creará una tarea nueva. Podemos crear hasta un máximo de 10 tareas, de forma que cuando creemos la décima ya no aparecerá el botón de crear. A continuación vamos a crear tres tareas.



Figura 9.32: Nuevas tareas

Una vez creadas las tareas (véase la figura 9.32), estando en el modo edición, si hacemos clic en cualquiera de las tareas, se nos abre el menú para personalizar la tarea (véase la figura 9.33). Aquí podemos, o bien cambiarle el ícono a la tarea (ya que por defecto trae el de la hoja) haciendo clic en el botón "Actualizar" tras haber escogido otro ícono, o bien borrarla haciendo clic en el ícono de la papelera. Para cerrar el menú es tan fácil como hacer clic fuera del menú o deslizarnos en la página. Por ejemplo, vamos a cambiarle el ícono a las dos primeras tareas y vamos a borrar la última.

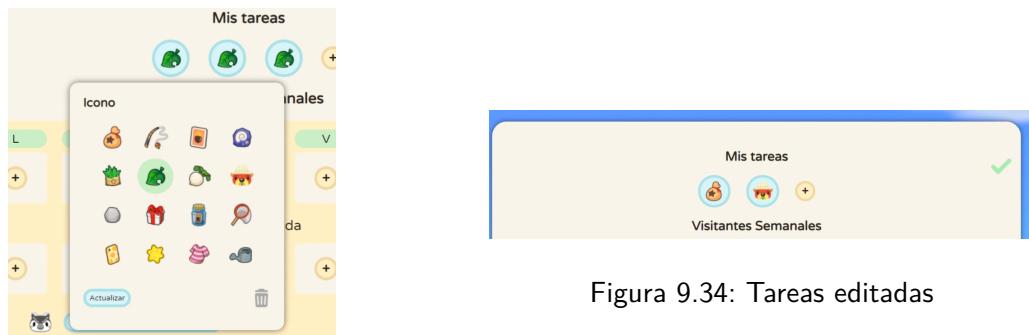


Figura 9.34: Tareas editadas

Figura 9.33: Menú tareas

Una vez hayamos terminado, hacemos clic en el check verde para salir del menú de edición (véase la figura 9.34) y ahora podemos activar o desactivar las tareas. Si hacemos clic en cualquier tarea, se activará, cambiando su color a verde (véase la figura 9.35). Si volvemos a hacer clic, se desactivará cambiando a azul. Si tenemos tareas activadas, al final del día se reiniciarán, ya que representan tareas diarias. De esta forma el usuario puede crear algunas tareas personalizables para representar su rutina en el juego, activarlas para recordar que ya la ha realizado y que se desactiven automáticamente para repetir el proceso cada día.



Figura 9.35: Tareas realizada

Ahora pasemos a la siguiente herramienta, los visitantes semanales. Aquí el usuario dispone de unos recuadros que representan tanto la semana actual como la anterior (véase la figura 9.36). Los recuadros correspondientes al Sábado y Domingo son inalterables ya que, con ciertas excepciones debido a torneos puntuales, siempre van a venir los mismos visitantes durante esos días.

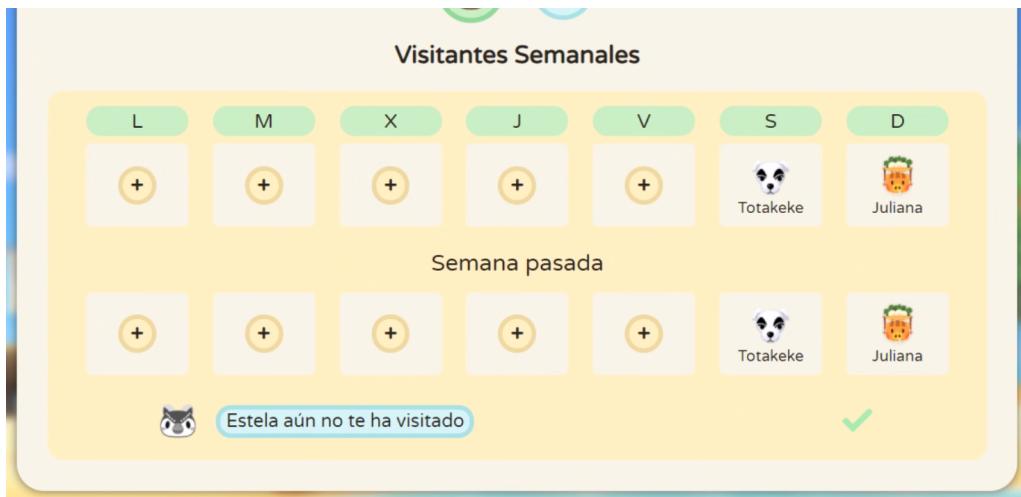


Figura 9.36: Visitantes semanales

En el resto de recuadros, si hacemos clic en el botón de añadir, se nos abre un menú en el que veremos una lista de los posibles visitantes. Dependiendo de los que ya nos hayan visitado, acorde con cómo funciona en el juego, habrá algunos iconos que no se podrán seleccionar ya que habrán alcanzado su límite. Como máximo, solo se puede repetir un visitante en dos semanas conjuntas. Para mostrarlo, procedemos a llenar los recuadros tal y como se ve en la imagen (véase la figura 9.37).



Figura 9.37: Semana rellanda

Si volvemos a hacer clic en cualquiera de los botones de añadir, veremos que los visitantes de la semana pasada, así como Alcatifa, que ya nos ha visitado en las dos semanas, serán menos visibles y no podrán seleccionarse (véase la figura 9.38).



Figura 9.38: Menú visitantes

En caso de querer modificar un visitante o eliminarlo, es tan fácil como hacer clic en su icono y, o bien cambiarlo por otro, o bien hacer clic en el ícono de la papelera. Además, en la parte inferior de la herramienta podemos ver que hay un botón con el icono de Estela. Si hacemos clic en él, podremos alternar entre dos estados, para indicar así si ya ha venido Estela esta semana a la isla.

Una vez hayamos modificado los datos, veremos que el ícono de guardar (el check verde) empieza a parpadear. Esto significa que tenemos cambios sin guardar, por lo que para guardar los datos hay que hacer clic. Si todo ha ido bien, veremos un mensaje confirmando que se ha guardado correctamente (véase la figura 9.39).

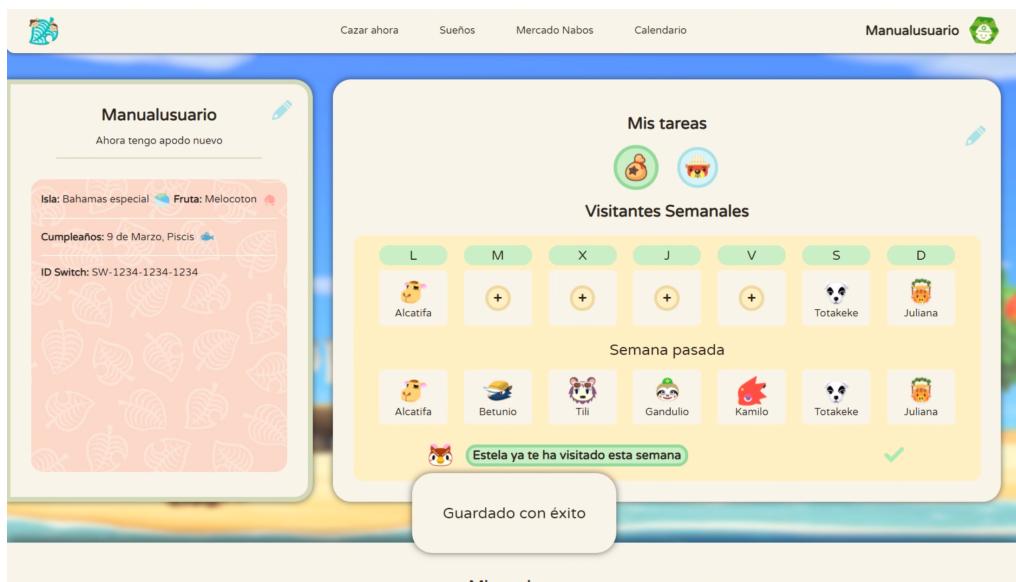


Figura 9.39: Confirmación guardado

A continuación, seguimos con el siguiente apartado. Si deslizamos la página hacia abajo, veremos la siguiente sección del perfil: Mis vecinos. En este apartado, el usuario puede añadir los vecinos que estén en su isla. Al principio, si es una cuenta nueva, veremos diez recuadros solo con los botones para añadir (véase la figura 9.40).

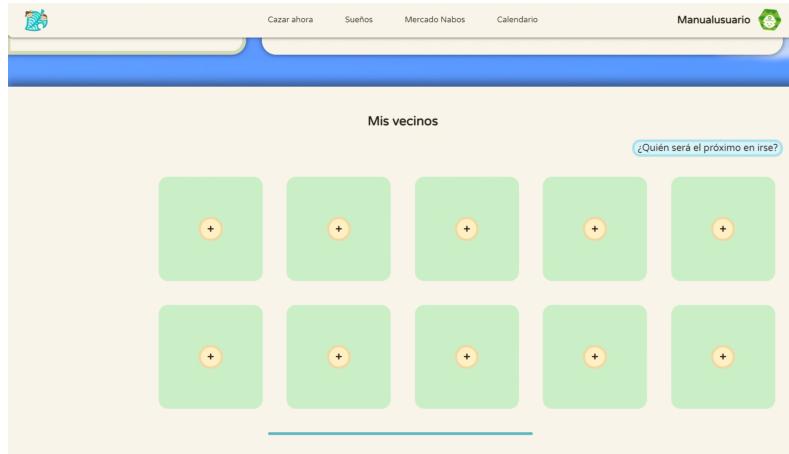


Figura 9.40: Mis vecinos

Para añadir un vecino, si hacemos clic en cualquiera de los botones, se abrirá un menú en el que podremos buscar y seleccionar el vecino que deseemos. Una vez seleccionado, le damos al check verde para guardar los datos. Podremos ver como ahora aparece nuestro primer vecino, así como un corazón al lado. Este corazón indica la amistad que se tiene con el vecino, que por defecto viene a uno.

Ahora, si colocamos el cursor sobre el recuadro del vecino, podremos observar que se nos muestra a la izquierda un recuadro con información del mismo, así como su foto (véase la figura 9.41).

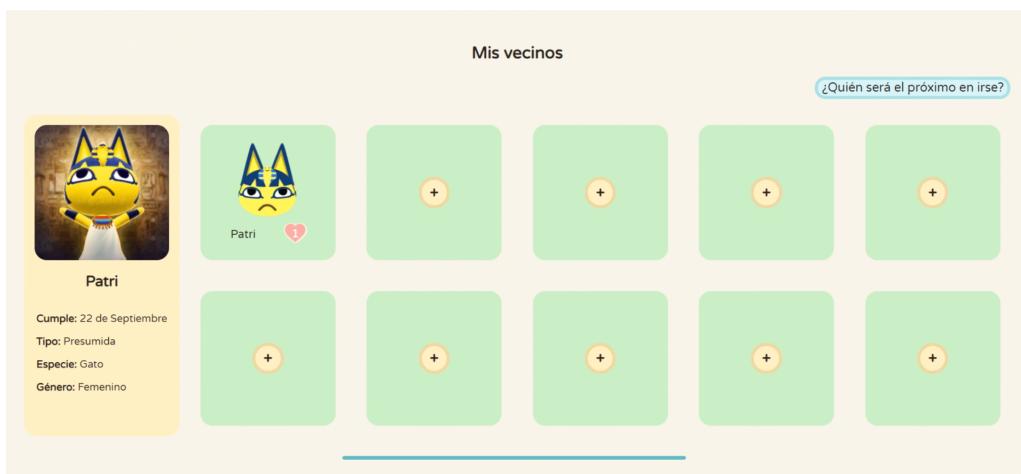


Figura 9.41: Información vecino

Si queremos cambiar o eliminar al vecino, es tan fácil como hacer clic en su ícono y volver a elegir otro y guardar, o a darle al botón de la papelera (véase la figura 9.42). A continuación, vamos a añadir más vecinos para explicar con mas claridad el siguiente apartado.



Figura 9.42: Menú vecinos



Figura 9.43: Menú amistad

Si hacemos clic en el corazón de cada vecino, veremos que se nos abre un menú con seis corazones (véase la figura 9.43). Aquí se puede cambiar el nivel de amistad del vecino haciendo clic sobre el corazón con el número deseado.

Si nos fijamos, en la parte superior derecha de la herramienta podemos ver un botón azul. Si hacemos clic en él, se nos abre un desplegable con información sobre el nivel de amistad de los vecinos, de forma que el usuario pueda saber qué nivel atribuir a cada vecino (véase la figura 9.44).

Figura 9.44: Información amistad

Además, en este desplegable vemos que hay un botón “Calcular porcentaje”. Si nos fijamos, en los recuadros han aparecido algunas cosas nuevas.

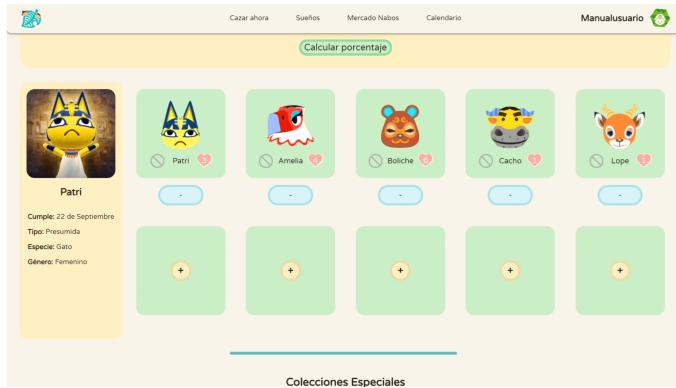


Figura 9.45: Mudanza

Como vemos, cada vecino tiene un ícono de bloquear, así como un pequeño recuadro azul (véase la figura 9.45). Esta sección sirve para calcular el porcentaje de mudanza de cada vecino, que tiene que ver con el nivel de amistad de cada uno. Siguiendo la información que podemos encontrar en el menú desplegable, podemos llenar los niveles de amistad de cada vecino, así como excluirlos del cálculo si fuera necesario haciendo clic en el botón de bloqueo. Si un vecino está siendo bloqueado, el ícono aparecerá en rojo, y por lo tanto no contará para el cálculo de la probabilidad.

Una vez tengamos listo tanto la amistad, como aquellos vecinos bloqueados (en caso de haberlos), le damos al botón de “Calcular porcentaje” que podemos encontrar más arriba. Podemos observar que han aparecido unos números en los recuadros azules (véase la figura 9.46), que indican la probabilidad que tiene cada vecino de mudarse de la isla. Como vemos, al haber bloqueado a Patri, esta no cuenta para el cálculo de la probabilidad y por lo tanto, no tiene número.

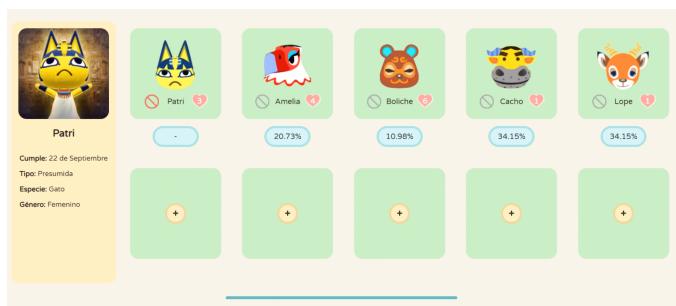


Figura 9.46: Cálculo porcentaje

Tal y como funciona el juego, aquellos vecinos con mayor amistad tienen menos probabilidad de mudarse, por lo que, como podemos ver, Boliche que tiene un nivel seis de amistad (el máximo), es el que tiene menos probabilidades de mudarse (casi un 11%), en comparación con Cacho y Lope, que al ser los que menos amistad tienen, hay más probabilidades de que sea uno de los dos el que decida mudarse (cada uno con casi un 34 % aproximadamente).

Una vez hayamos terminado, podemos volver a la vista original cerrando el menú desplegable haciendo clic en el botón azul que se encuentra en la zona superior derecha del menú desplegable. Sigamos con la próxima herramienta, las Colecciones Especiales.

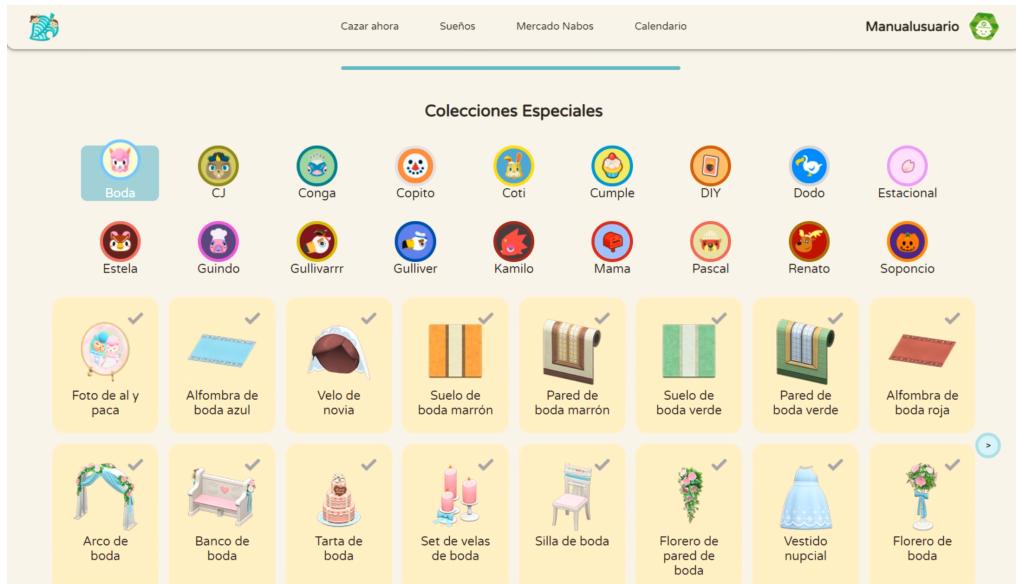


Figura 9.47: Colecciones especiales

Lo primero que nos encontramos es una serie de botones que indican la colección (véase la figura 9.47). Una de ellas estará resaltada para denotar que es la colección activa. Si hacemos clic en los distintos botones, vemos que los ítems de abajo van variando. Una vez hayamos seleccionado la colección deseada, podemos observar que hay varios ítems, cada uno con su imagen y nombre, así como un botón con forma de check gris. Si hacemos clic en este botón, se volverá rojo indicando así que ya disponemos de este ítem (véase la figura 9.48).

Además, en caso de que sea necesario, hay botones para navegar por las páginas de ítems a los lados de la herramienta.



Figura 9.48: Check obtenido

Por último, tenemos el álbum de fotos del usuario (véase la figura 9.49). Si hacemos clic en el botón del menú de la derecha, nos aparecerá un menú para introducir la url de la imagen que queramos añadir a nuestro álbum (véase la figura 9.50).



Figura 9.49: Álbum fotos



Figura 9.50: Añadir imagen

Una vez dispongamos de alguna imagen, aparecerá en grande una a la izquierda (véase la figura 9.51), y en caso de tener más de una, podemos seleccionar la imagen que queremos ver en grande, o bien de forma temporal colocando el cursor encima de la imagen, o bien haciendo clic para que se mantenga esa selección, que se mostrará algo más grande que las demás.



Figura 9.51: Álbum con fotos

Si deseamos borrar alguna foto, es tan sencillo como activar el modo borrado haciendo clic en la papelera. Sabremos que estamos en el modo borrado porque cambiará el fondo del menú de morado a rojo (véase la figura 9.52), y las imágenes temblarán si el cursor se sitúa sobre ellas. Una vez activado, si hacemos clic en las imágenes, las borraremos.



Figura 9.52: Borrar imagen

Ahora que hemos visto el funcionamiento del álbum de fotos, podemos pasar a explicar la última herramienta de la página, que sería el catálogo de sueños, al que podremos acceder si hacemos clic en el botón “Sueños”, situado en la barra de navegación superior. Esto nos llevará a un catálogo donde podremos ver los sueños que han subido los usuarios (véase la figura 9.53).



Figura 9.53: Catálogo sueños

En cada ítem podremos ver, el nombre de la isla del sueño, su código para poder viajar al mismo, el usuario que ha subido el sueño, así como algunas fotos de la isla que haya querido subir el usuario. En caso de que haya subido más de una (hasta un límite de tres), podemos navegar entre las mismas mediante las flechas situadas a los lados (véase la figura 9.54).



Figura 9.54: Sueño

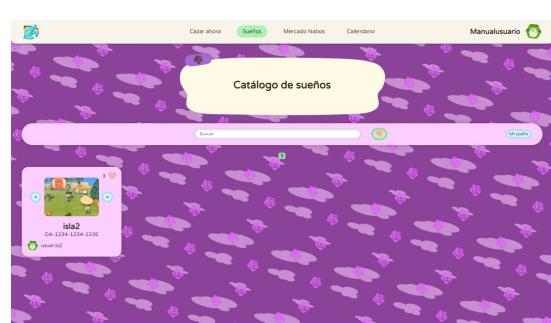


Figura 9.55: Catálogo filtrado

Además, cada sueño tiene un corazón en la parte superior derecha que indica el número de “Me gusta” que ha recibido, y haciendo clic sobre él podemos darle o quitarle el “Me gusta”. Con esto, no solo podemos indicar que nos ha gustado un sueño, sino que le sirve al usuario como una lista personal de favoritos, ya que en los filtros puede filtrar tanto por el nombre de la isla como por aquellos a los que le haya dado “Me gusta”, (véase la figura 9.55).

Si el usuario lo desea, puede subir un sueño haciendo clic en el botón “Mi sueño” que encontrará junto a los filtros a la derecha. Al hacer clic, se abrirá un menú tal y como se ve en la imagen (véase la figura 9.56). Aquí podremos introducir el código de nuestro sueño, así como de una a tres imágenes, las cuales podremos seleccionar o deseleccionar haciendo clic en ellas. Una vez hallamos llenado los datos, pulsamos en el botón del check verde para confirmar la creación de nuestro sueño.



Figura 9.56: Crear sueño

En caso de querer modificar tanto el código como las imágenes, es tan sencillo como volver a hacer clic en el botón de “Mi sueño”, y luego modificar los datos. En caso de querer eliminarlo, en el mismo menú podemos hacer clic en el ícono de la papelera y confirmar nuestra decisión (véase la figura 9.57).



Figura 9.57: Confirmación borrado

Conclusiones y desarrollo futuro

La realización del proyecto ha sido muy enriquecedora, ya que nos ha permitido comprender el esfuerzo que conlleva realizar un trabajo desde cero: pasar por cada una de sus fases, desde que solo es una idea, pasando por la planificación, el desarrollo, y hasta que finalmente es un producto completo y la idea ha pasado a ser realidad. Todo esto, sumado a la cantidad de preparación que requiere realizar un diseño y un análisis preciso, así como la dificultad de formar a un equipo en un lenguaje desconocido, conforma un reto bastante interesante del que nos sentimos bastante orgullosos de haber participado y llevado a cabo hasta el final.

Durante todo el trascurso del proyecto, como era de esperar, nos hemos visto en la necesidad de tomar varias decisiones y afrontar varios problemas (como se ha podido ver en el capítulo 6). Sin embargo, no podríamos estar más contentos con todos estos fallos y problemas que hemos tenido, ya que es así como de verdad podemos aprender para intentar evitar los mismos fallos en el futuro. Porque al final, este proyecto no se basaba exclusivamente en realizar una aplicación, sino que una de los principales objetivos era formarnos en el proceso y ampliar nuestro conocimientos, y es por eso por lo que decidimos usar un framework que no habíamos utilizado nunca, así como otros lenguajes en los que no teníamos tanta experiencia, todo para que pudiéramos aprovechar al máximo el tiempo empleado y salir del proyecto con nuevos conocimientos para abrirnos fronteras.

Y al final, eso es lo que hemos hecho, aprender. Aprender sobre el framework, sobre el lenguaje, sobre todo lo que necesitásemos para poder seguir adelante, y es algo que no hemos dejado de hacer durante toda la duración del proyecto. Sprint tras sprint descubríamos nuevas formas, mucho mas sencillas y eficientes, de abarcar los mismos problemas, y si a día de hoy (una vez finalizado el proyecto) echamos la vista hacia atrás, podemos ver todo lo que hemos aprendido desde que empezamos y el largo camino que hemos recorrido en algo menos de un año.

Cabe mencionar que consideramos que el proyecto no ha quedado tan completo como lo habríamos deseado, ni mucho menos, pero aspirar a más hubiera sido un grave error teniendo en cuenta el tiempo que teníamos asignado. Aún queda mucho margen de mejora, y a medida que pase el tiempo y vayan saliendo nuevas actualizaciones para el juego, más herramientas y más características se podrán añadir a nuestra aplicación. Sin embargo, seguimos pensando que el resultado obtenido ha sido bastante bueno (y bastante cercano en la gran mayoría de sus aspectos a lo que en un principio se ideó).

En resumen, no podríamos estar más satisfechos con el proyecto: tanto por el resultado que hemos acabado obteniendo, como por todo lo que hemos aprendido por el camino, sin duda ha sido una gran oportunidad para demostrar lo que hemos aprendido durante estos años de carrera, seguir formándonos y vernos cara a cara con un situación muy cercana a la realidad que sin duda nos será de gran ayuda para afrontar la siguiente etapa de nuestras vidas.

Desarrollo futuro

Aunque hemos realizado y centralizado muchas de las herramientas más importantes, un juego tan grande como es "Animal Crossing" nos ofrece mucho margen de mejora para nuestra aplicación. Hay bastantes herramientas y funciones que nos hubiese gustado implementar, pero que por cuestión de tiempo hemos optado por no hacerlo para centrarnos en otras.

A continuación listamos algunas de las funcionalidades más relevantes que pensamos que podríamos implementar en un futuro como mejoras para la aplicación:

- Modo anti-spoilers: una funcionalidad que consideramos bastante útil es un modo anti-spoilers para los listados de criaturas, ya que puede que haya ciertos jugadores que, a la llegada de un nuevo mes (y nuevas criaturas que capturar), quieran conocer pistas para saber sobre qué momento del día y lugar pueden capturar aquellas criaturas que aún no tengan pero no deseen saber cuáles son para llevarse la sorpresa, por lo que un modo que ocultase las imágenes y el nombre de las criaturas en el listado podría ser algo interesante para implementar.
- Al igual que los usuarios pueden subir su isla con su código de sueño, una funcionalidad muy interesante sería la de subir sus propios diseños. Dentro del juego el usuario puede crear diseños que puede usar tanto para vestimenta como para decoración, por lo que un catálogo de diseños donde los usuarios suban sus propias creaciones para compartirlas con el resto de jugadores, así como un sistema de favoritos y "Me gusta", es algo bastante útil para los jugadores.
- En el apartado de "Colecciones especiales" se podría añadir un filtro de búsqueda por texto para facilitarle la búsqueda al usuario en aquellas colecciones especialmente extensas, así como añadir algunas otras posibles colecciones que puedan ser de interés para el usuario, ya que el juego sigue actualizándose y añadiendo nuevo contenido.
- Un apartado sobre el que se ha hecho poco hincapié ha sido el de la jardinería. El juego posee un sistema de cultivo de flores que, a partir de unos colores base se pueden llegar a generar otras flores de colores distintos y menos comunes. Es un sistema complejo ya que dependiendo del color y origen de la flor, ésta posee un gen u otro que puede servir para cultivar un color en particular. Pensamos que una herramienta bastante útil podría ser una especie de matriz sobre la que el usuario pueda colocar las flores que desee en la forma que así vea conveniente y que la aplicación le muestre qué colores pueden ser generados en ciertas posiciones de la matriz siguiendo la disposición introducida.

Referencias

- Alexis8717 (2020). Acnh api. fecha de consulta: Febrero de 2021.
- Autores, V. (2020). Turnip prophet. fecha de consulta: Abril de 2021.
- KevinPayravi (2020a). Acnh.directory. fecha de consulta: Octubre de 2020.
- KevinPayravi (2020b). Nookiepdia api. fecha de consulta: Febrero de 2021.
- LeParadoxHD (2019). Angular 10 - curso completo - tutorial. fecha de consulta: Octubre de 2020.
- Lewis, M. (2018). Angular calendar. fecha de consulta: Abril de 2021.
- Ninji (2020). Turnip prices. fecha de consulta: Abril de 2021.
- Norviah (2020). Animal crossing: New horizons database for npm. fecha de consulta: Febrero de 2021.