ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de Grado

Ingeniería Informática - Ingeniería del Software

My Crossing

Realizado por
Moisés Pantión Loza
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 tardaron mucho en salir varias webs y páginas 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 muchas otras, sirve de ayuda para el jugador proporcionándole herramientas e información sobre el juego.

No obstante, buscábamos proporcionarle al usuario un sistema de información lo más completo posible y que le ayudase tanto en el día a día como en tareas más esporádicas, por lo que como usuarios y parte de la comunidad que somos, decidimos 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 Spring Boot 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

TODO==========

Índice general

Índice de cuadros

Índice de figuras

Índice de código

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 las recoja a todas. 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.

Nuestra aplicación web mostrará ciertos datos dependiendo de si pertenecemos al hemisferio norte o sur, de la hora actual, de la fecha en la que nos encontremos etc, por lo que podemos concluir con que el servicio se podrá utilizar de forma internacional, siempre y cuando se sepa la lengua española.

CAPÍTULO 2

Definición de objetivos

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

- **Objetivos académicos:**
  - Centralizar 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 las herramientas a las que uno debería de acceder para obtener toda 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 mas conocida y con una interfaz más cercana al usuario, siguiendo un diseño similar a la estética del HUD 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.

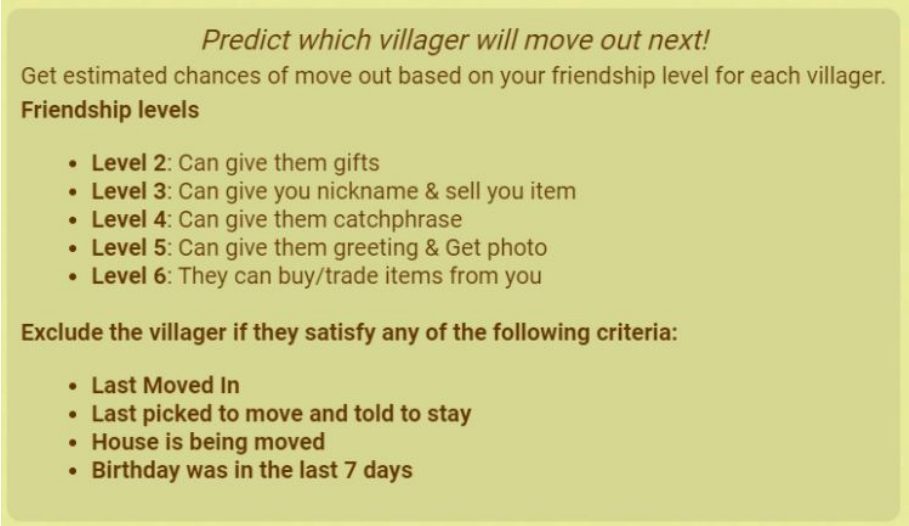


Figura 3.2: Mudanza

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 todas 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 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.

Por último, hemos añadido una herramienta original (que a día de hoy no hemos conseguido encontrar en Internet): El Vestidor. Esta herramienta simula un vestidor de cualquier tienda

de ropa, pero aplicada a las prendas del videojuego. Dispone de una interfaz que permite personalizar al personaje para que resulte lo más parecido al del usuario, o incluso probar nuevas combinaciones. De la misma forma, se puede vestir a dicho personaje con cualquier prenda que forme parte del videojuego.

Pensamos que esta herramienta es bastante útil ya que de este modo, un usuario puede probar distintos estilos y prendas para ver cual le convence más sin necesidad de disponer del objeto dentro del videojuego. De esta forma se puede centrar en conseguir aquella prenda que le interese en vez de realizar un proceso de prueba y error adquiriendo ropa que no utilizará y gastando tiempo y recursos. Además, cuenta con filtros para que no solo la búsqueda sea mas sencilla, sino que pueda idear conjuntos de ropa de manera más fácil.

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 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 API y los elementos relacionados con esta. Habría que hacer un repaso de lo hecho en la iteración anterior para aplicar las imagenes 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 realizan las pruebas (+ las del anterior?).

Iteración 5

Implementar 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.

Iteración 6

Terminar la documentación, realizar los apartados restantes de la documentación y cerrar el proyecto.

4.1.2. Estimación vs Realidad

Iteración	Horas estimadas	Horas reales	Diferencia (
1	15	12	-3
2	50		
3	85		
4	200		
5	85		
6	90		
7	80		
8	45		
Total	650		

//TODO cambiar tabla Estimacion - Reales - Diferencia 1: 15 - 12 2: 50 - 56 3: 170 - 4: 170 5: 200 6: 45 Total: 650

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:

TODO TABLA TOTAL

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 Pantión 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

//TODO Hacer requisito de info sobre usuario

IRQ-001	Información sobre Insectos

El sistema deberá almacenar y mostrar los siguientes datos sobre los insectos:

- Nombre
- Disponibilidad
- Rareza
- Imagen
- Precio

IRQ-002	Información sobre Peces

El sistema deberá almacenar y mostrar los siguientes datos sobre los peces:

- Nombre
- Disponibilidad
- Sombra
- Rareza
- Imagen
- Precio

IRQ-003	Información sobre Criaturas de mar

El sistema deberá almacenar y mostrar los siguientes datos sobre las criaturas de mar:

- Nombre
- Disponibilidad
- Sombra
- Velocidad
- Imagen
- Precio

IRQ-004	Información sobre Fósiles

El sistema deberá almacenar y mostrar los siguientes datos sobre los fósiles:

- Nombre
- Imagen
- Precio

IRQ-005	Información sobre Arte

El sistema deberá almacenar y mostrar los siguientes datos sobre las obras de arte:

- Nombre
- Imagen
- Existencia de copia falsa

IRQ-006	Información sobre Aldeanos

El sistema deberá almacenar y mostrar los siguientes datos sobre los aldeanos:

- Nombre
- Imagen
- Icono
- Personalidad
- Cumpleaños
- Especie
- Género

IRQ-007	Información sobre Canciones

El sistema deberá almacenar y mostrar los siguientes datos sobre las canciones:

- Nombre
- Imagen
- Música

IRQ-008	Información sobre Ropa

El sistema deberá almacenar y mostrar los siguientes datos sobre las prendas de ropa:

- Nombre
- Tipo de prenda
- Fuente
- Imagen
- Variaciones
- Precio

IRQ-009	Información sobre Muebles

El sistema deberá almacenar y mostrar los siguientes datos sobre los muebles:

- Nombre
- Tipo de mueble
- Fuente
- Tamaño
- Si se puede pedir
- Imagen
- Variaciones
- Precio
- Precio Millas

IRQ-010	Información sobre Eventos
El sistema deberá almacenar y mostrar los siguientes datos sobre los eventos: • Nombre • Horas • Descripción • FechaHN • FechaHS	

5.2.2. Requisitos Funcionales

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

FRQ-002	Catálogo Peces
El sistema deberá permitir el acceso a un catálogo de Peces	

FRQ-003	Catálogo Bichos
El sistema deberá permitir el acceso a un catálogo de Bichos	

FRQ-004	Catálogo Criaturas marinas
El sistema deberá permitir el acceso a un catálogo de Criaturas marinas	

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

FRQ-006	Catálogo Arte
El sistema deberá permitir el acceso a un catálogo de Obras de Arte	

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

FRQ-008	Catálogo Ropa
El sistema deberá permitir el acceso a un catálogo de Ropa	

FRQ-009	Catálogo Vecinos
El sistema deberá permitir el acceso a un catálogo de Vecinos	

FRQ-010	Calendario de Eventos
El sistema deberá permitir el acceso a un calendario de Eventos	

FRQ-011	Catálogo Canciones
El sistema deberá permitir el acceso a un catálogo de Canciones	

FRQ-012	Criaturas en tiempo real
El sistema deberá permitir ver las criaturas disponibles para atrapar en tiempo real	

FRQ-013	Calculador de nabos
El sistema deberá permitir el cálculo del patrón de venta de nabos	

FRQ-014	Diseños de la comunidad
El sistema deberá permitir la búsqueda de diseños hechos por la comunidad	

FRQ-015	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-016	Acceso Perfil
El sistema deberá permitir el acceso a un perfil propio	

FRQ-017	Edición Perfil
El sistema deberá permitir la edición del perfil propio	

FRQ-018	Visitantes semanales
El sistema deberá permitir ver y editar los visitantes semanales de mi isla	

FRQ-019	Tareas
El sistema deberá permitir ver, añadir, editar y borrar tareas diarias a elección del usuario	

FRQ-020	Multimedia
El sistema deberá permitir subir y borrar fotos y vídeos de la isla del usuario	

FRQ-021	Avance colección Peces
El sistema deberá permitir ver el avance personal de la colección de Peces, así como los que faltan por obtener	

FRQ-022	Avance colección Bichos
El sistema deberá permitir ver el avance personal de la colección de Bichos, así como los que faltan por obtener	

FRQ-023	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-024	Avance colección Canciones
El sistema deberá permitir ver el avance personal de la colección de Canciones, así como las que faltan por obtener	

FRQ-025	Avance colección Fósiles
El sistema deberá permitir ver el avance personal de la colección de Fósiles, así como los que faltan por obtener	

FRQ-026	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-027	Avance colección personal
El sistema deberá permitir registrar el avance personal de las colecciones	

FRQ-028	Probabilidad mudanza
El sistema deberá permitir calcular la probabilidad de que mis vecinos se muden de la isla	

FRQ-029	Restricción avance colecciones
El sistema no deberá permitir que un usuario modifique el avance de las colecciones de otro usuario	

FRQ-030	Restricción perfil
El sistema no deberá permitir que un usuario modifique el perfil de otro usuario	

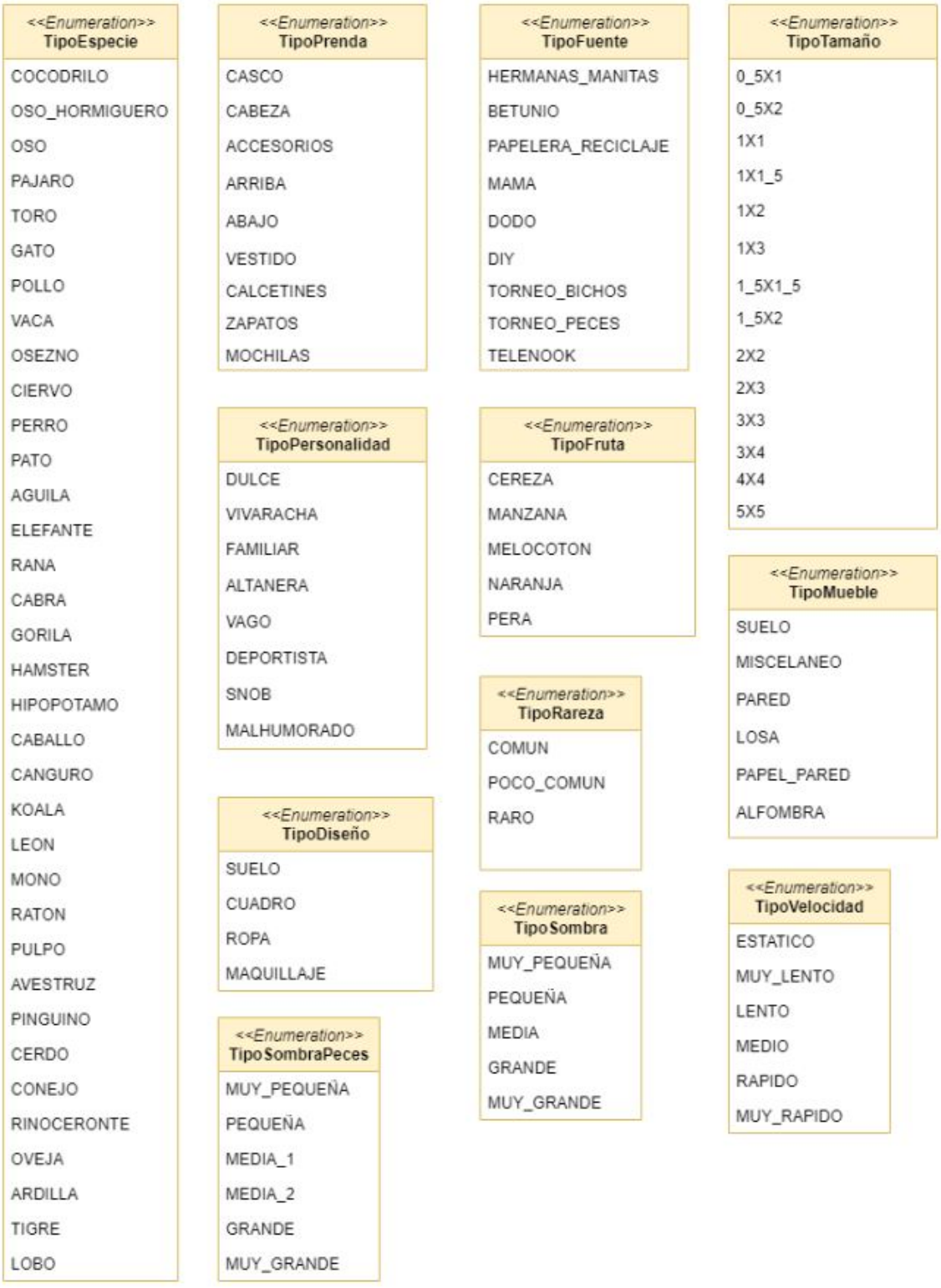
5.2.3. Requisitos no Funcionales

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	

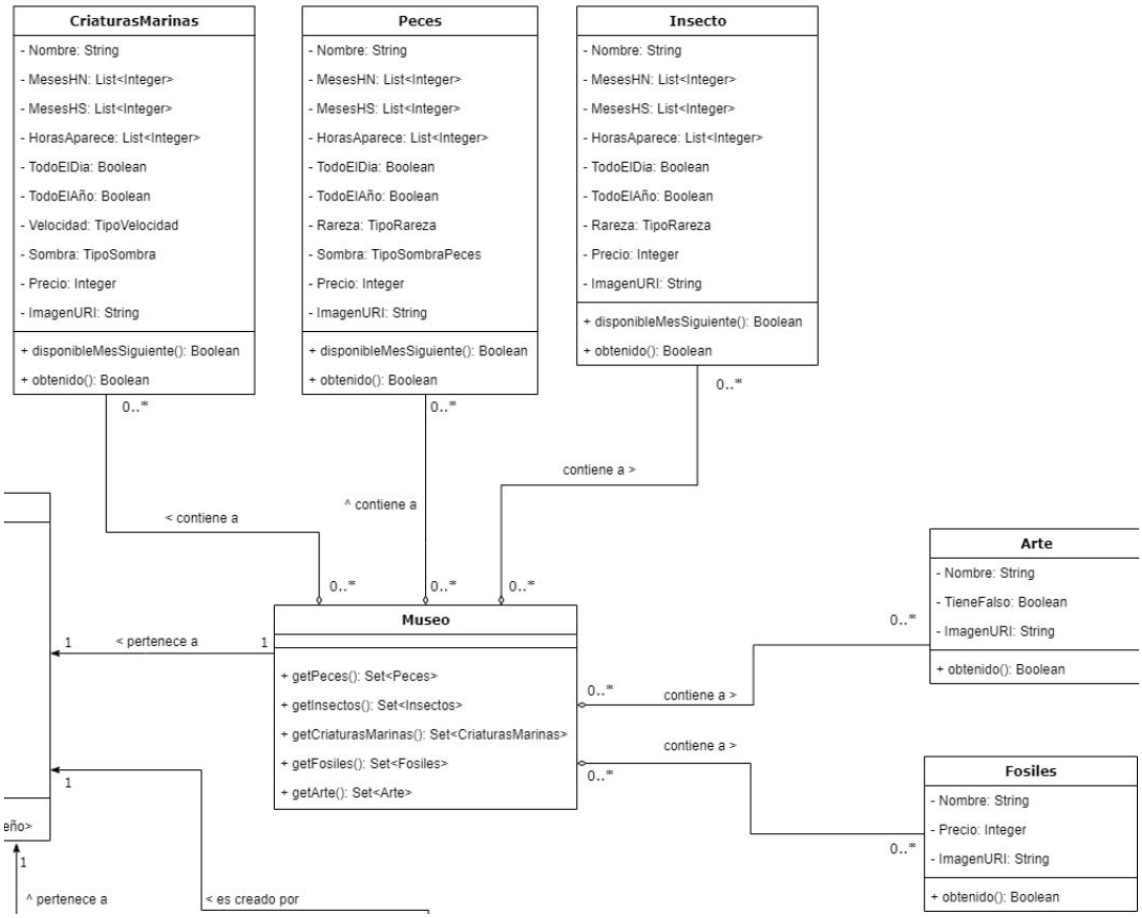
5.3– Diagrama de Clases

Dado que tratamos con grandes cantidades de información, gran parte de ella la tenemos que obtener a través de APIs. Así mismo, como sed trata de información de un videojuego, muchos datos han de aparecer en forma de enumeraciones.

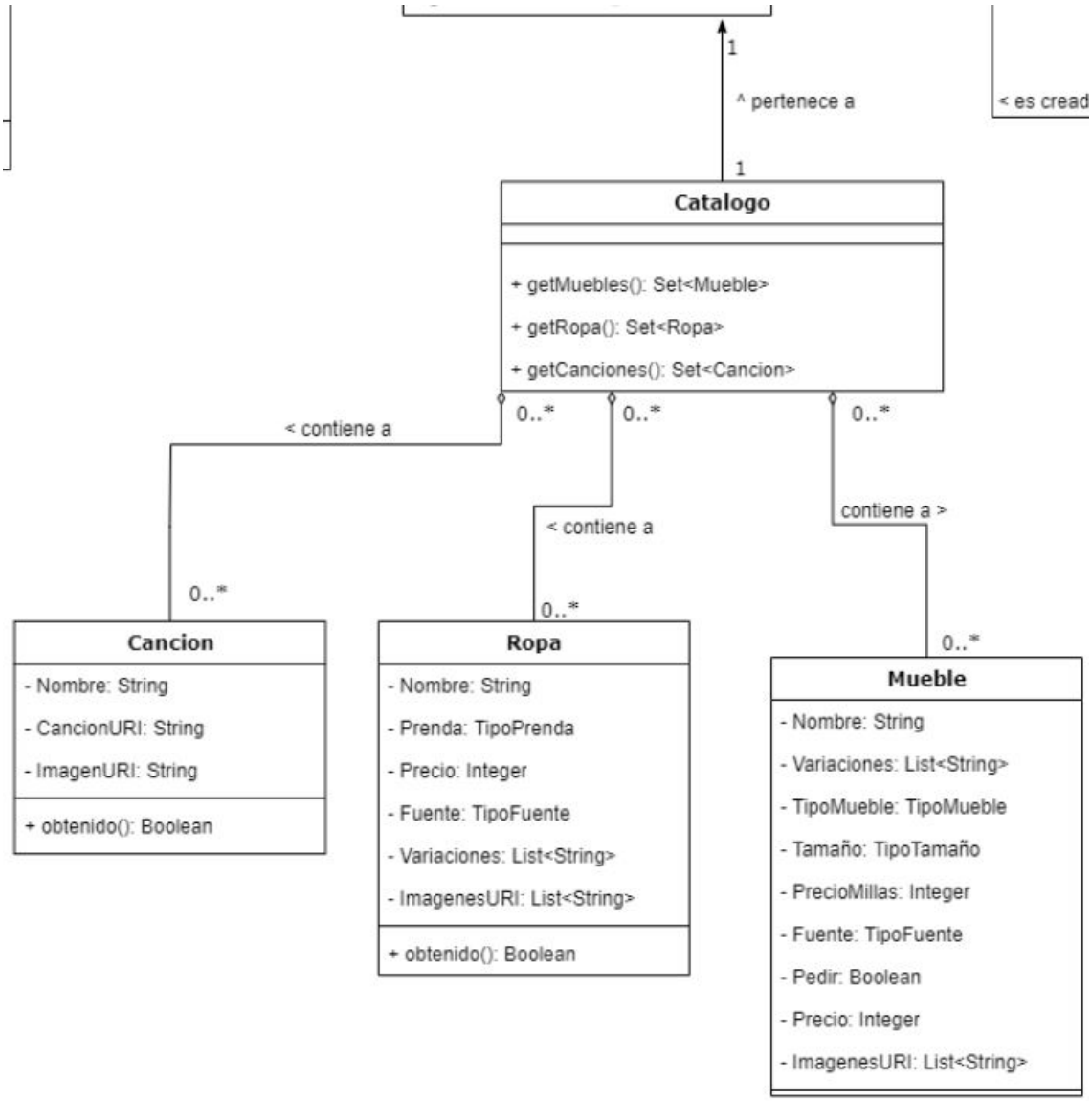


Debido al gran tamaño del diagrama de clases, vamos a verla dividida en partes más pequeñas de forma que muestren distintas relaciones y se pueda ver de forma más clara.

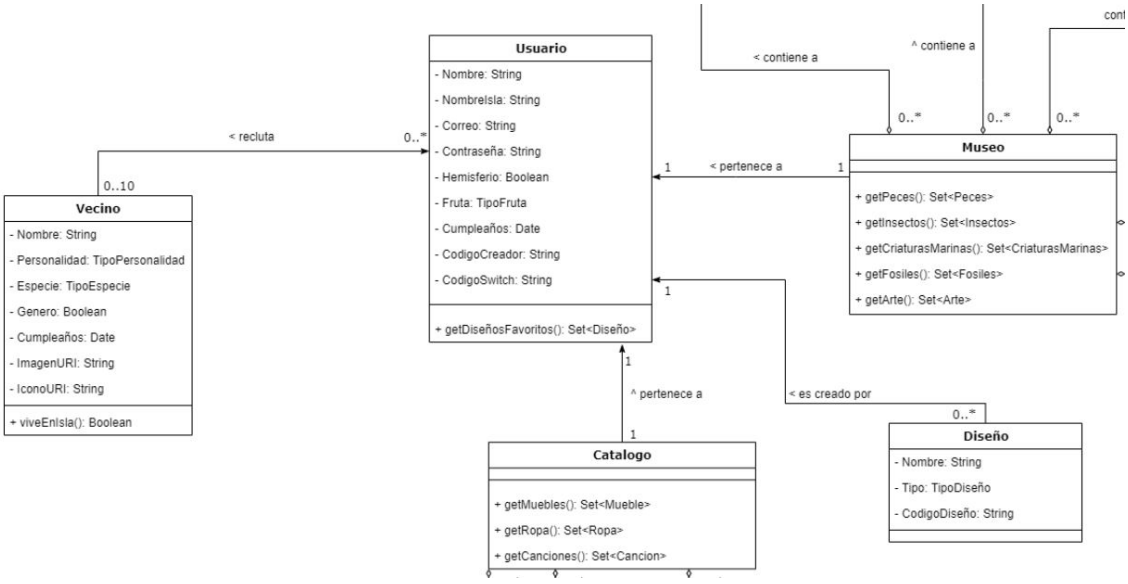
A continuación se puede ver la clase Museo, la cual recoge la colección personal de Insectos, Peces, Criaturas Marinas, Fósiles y Obras de Arte.



Luego tenemos la clase Catálogo que de manera análoga hace lo mismo con la Ropa, los Muebles y las Canciones.



Tanto Museo como Catálogo van asociados a Usuario, así como Vecinos y Diseños.



Por último está la clase Evento que no está relacionada con ninguna otra, ya que solo se

usarán los datos para el Calendario de Eventos el cual es meramente informativo.

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.

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 (véase la figura 5.1), 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, cambiado 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 item 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

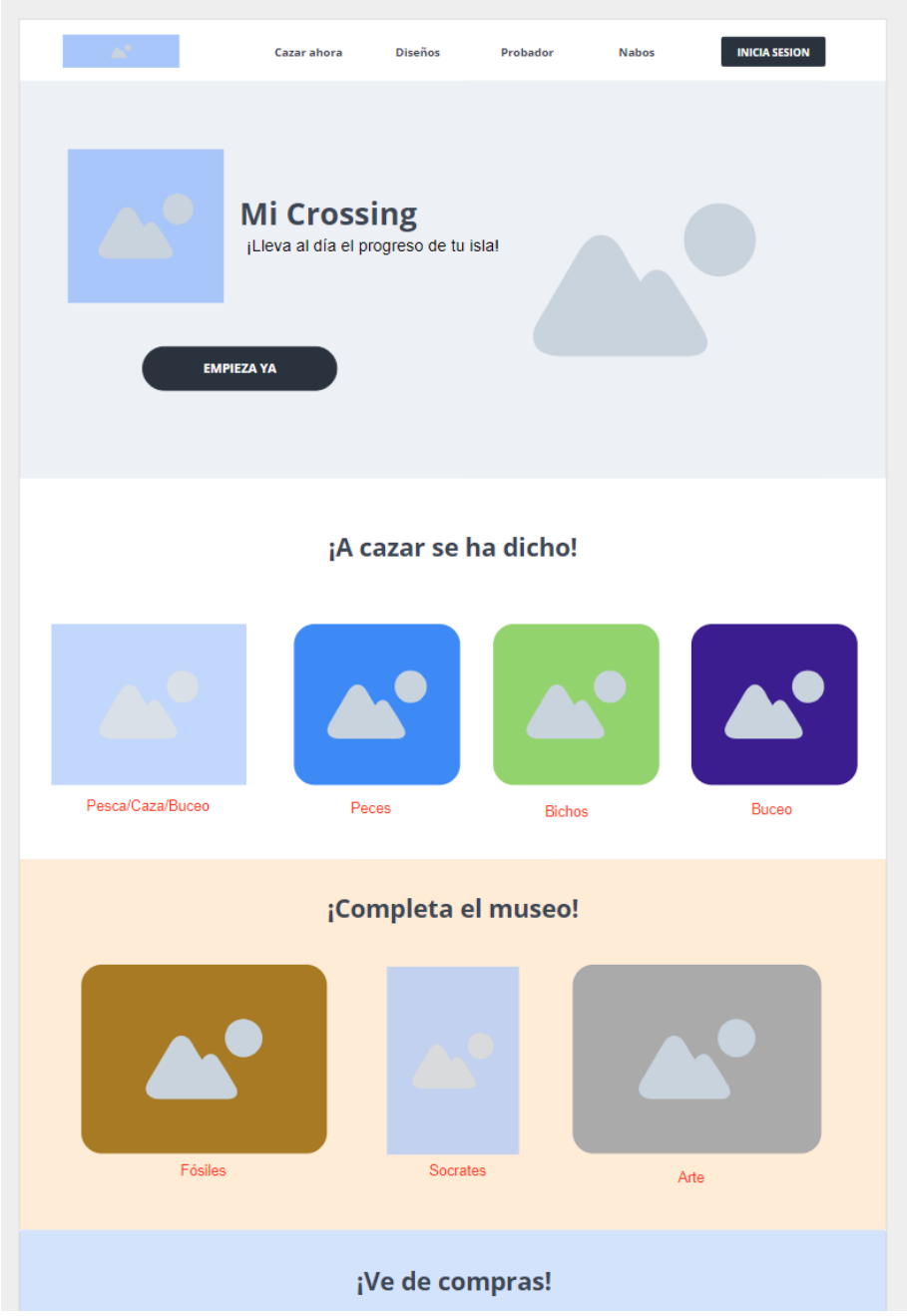


Figura 5.1: Inicio 1

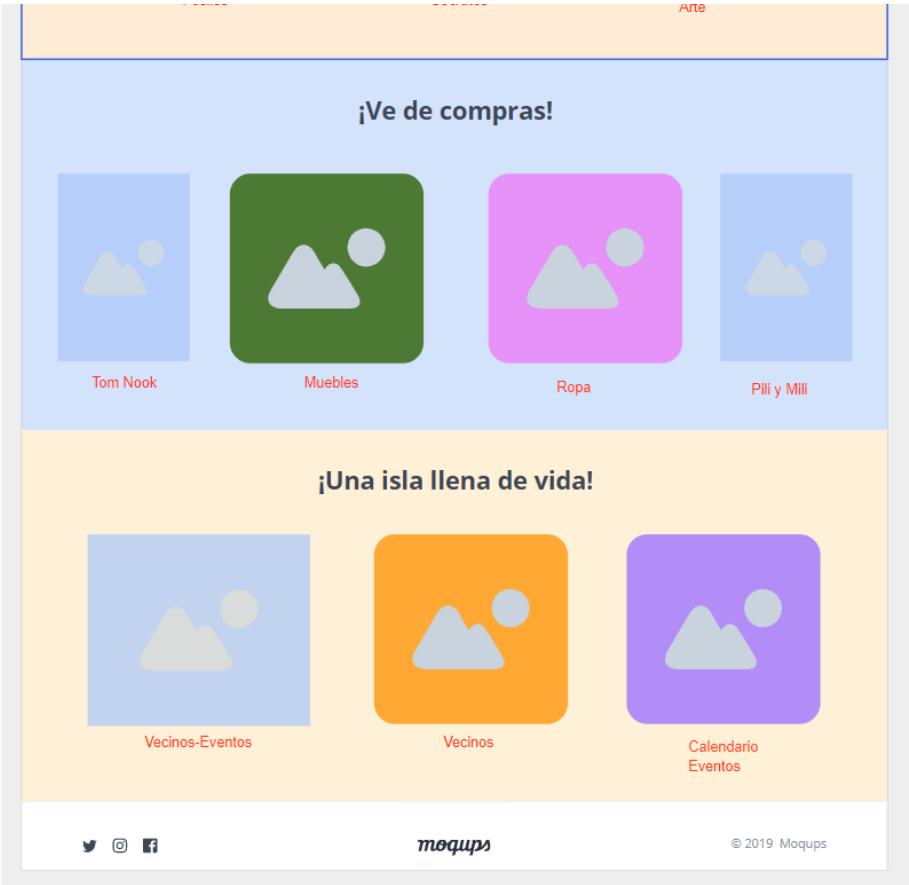


Figura 5.2: Inicio 2

de los que ya ha capturado marcándolos con el tick superior derecho en cada item, cambiándose así el fondo a un color distinto para diferenciarlos a simple vista (véase la figura 5.3).

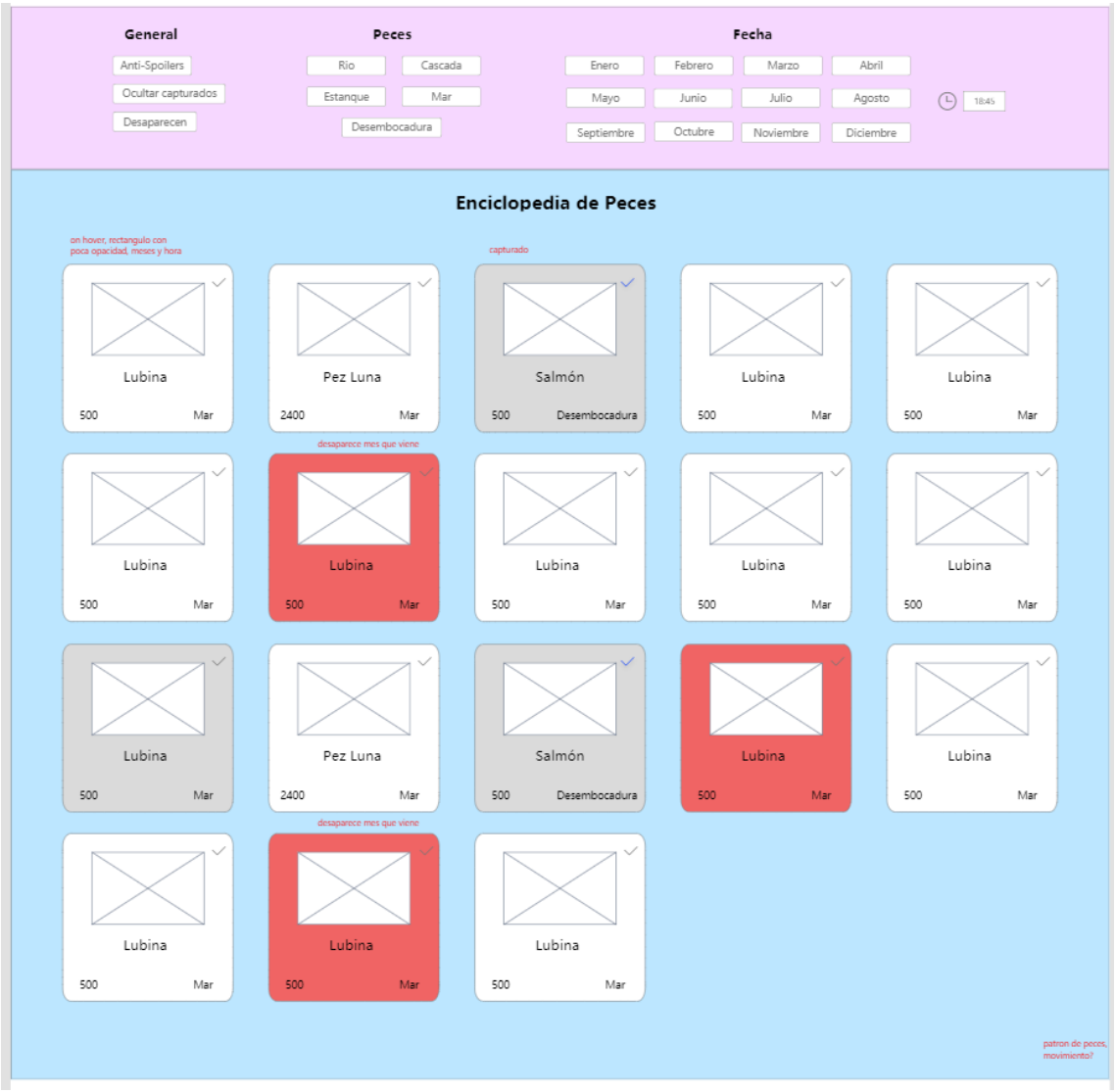


Figura 5.3: Listado genérico

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).

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.

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

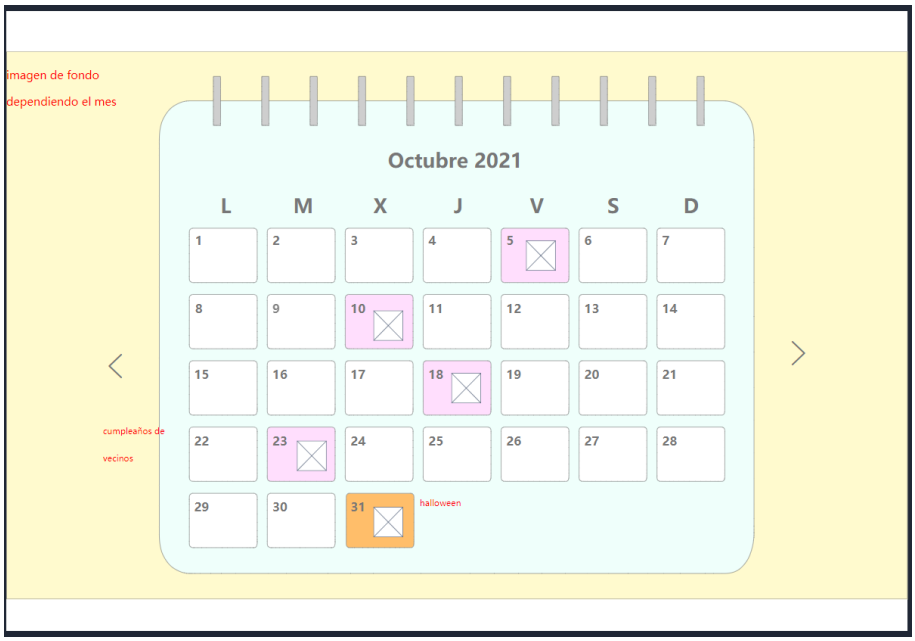


Figura 5.4: Eventos

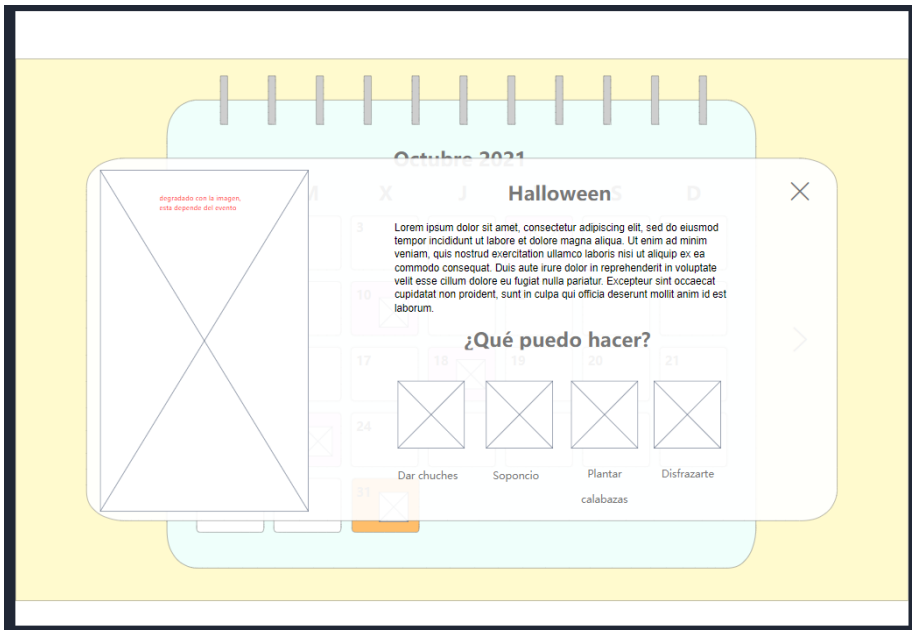


Figura 5.5: Eventos - Detallado

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.

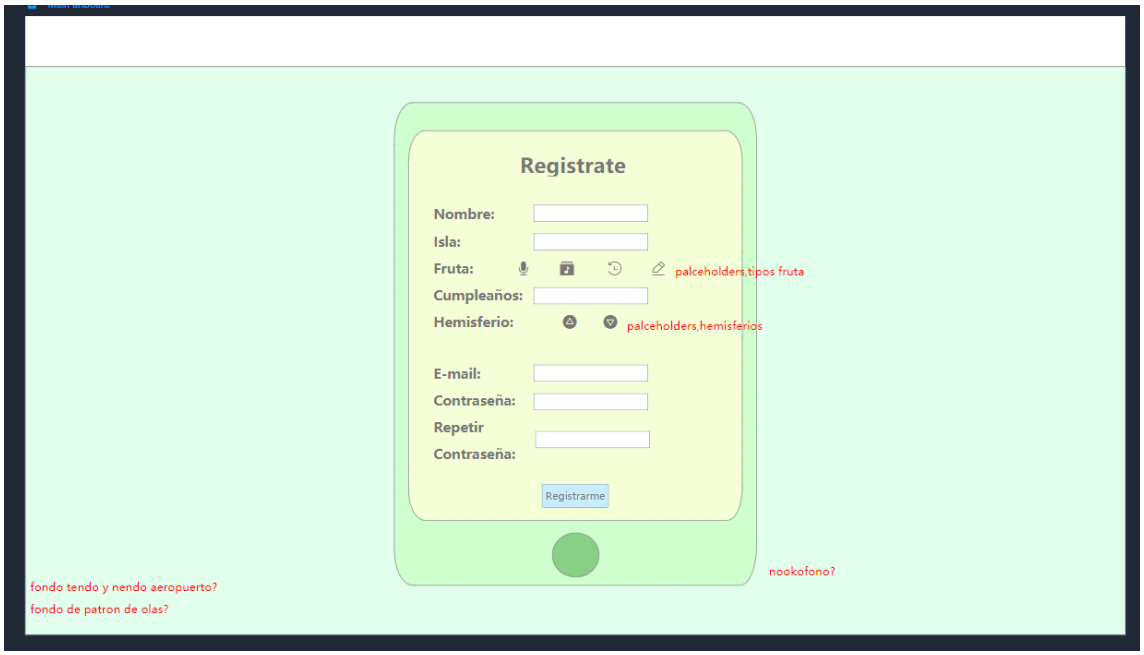


Figura 5.6: Registro

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 icono 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 rellenando 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.

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 diferen-

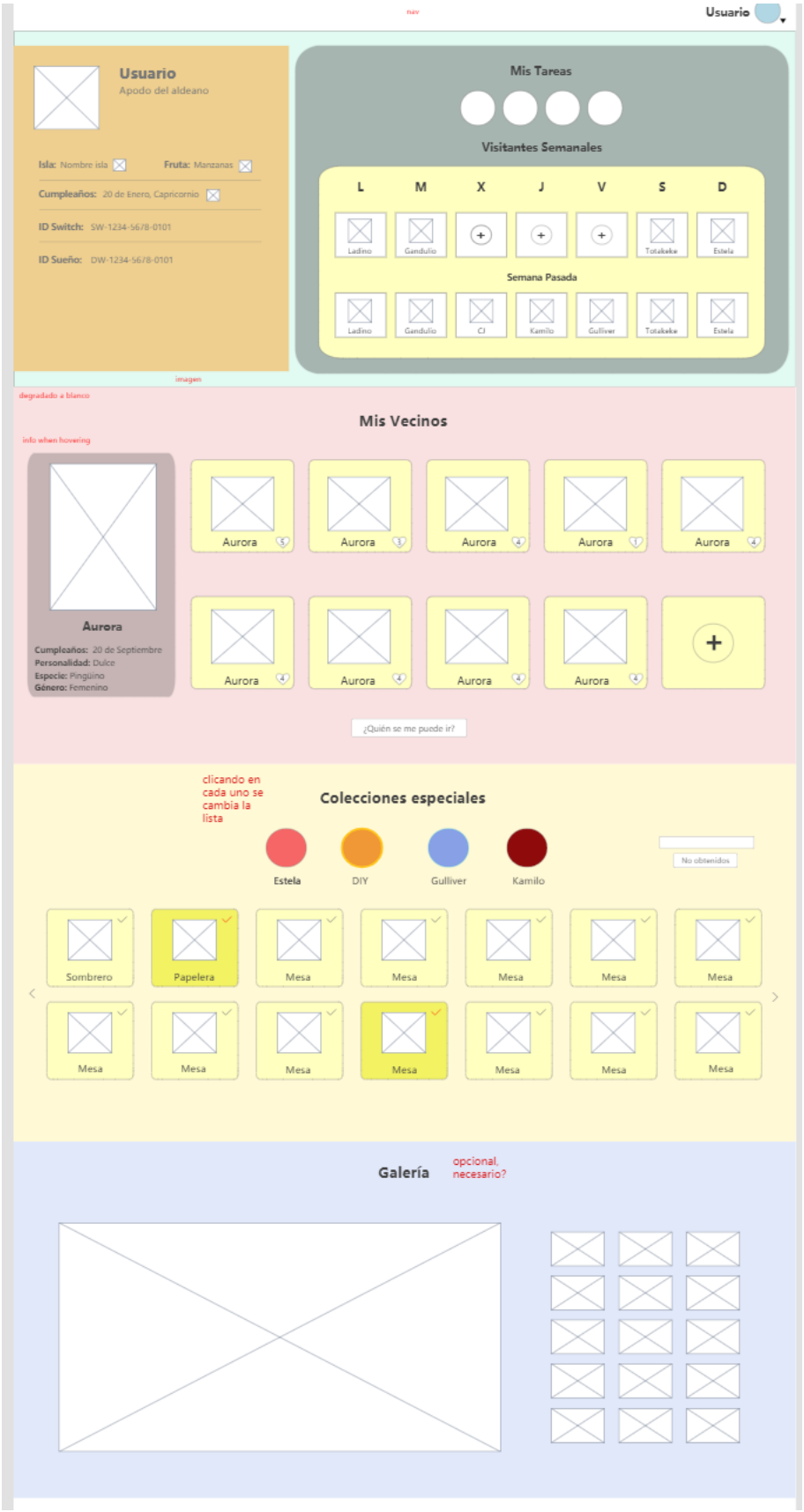


Figura 5.7: Perfil

cia, 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 que es lo que le aguarda.

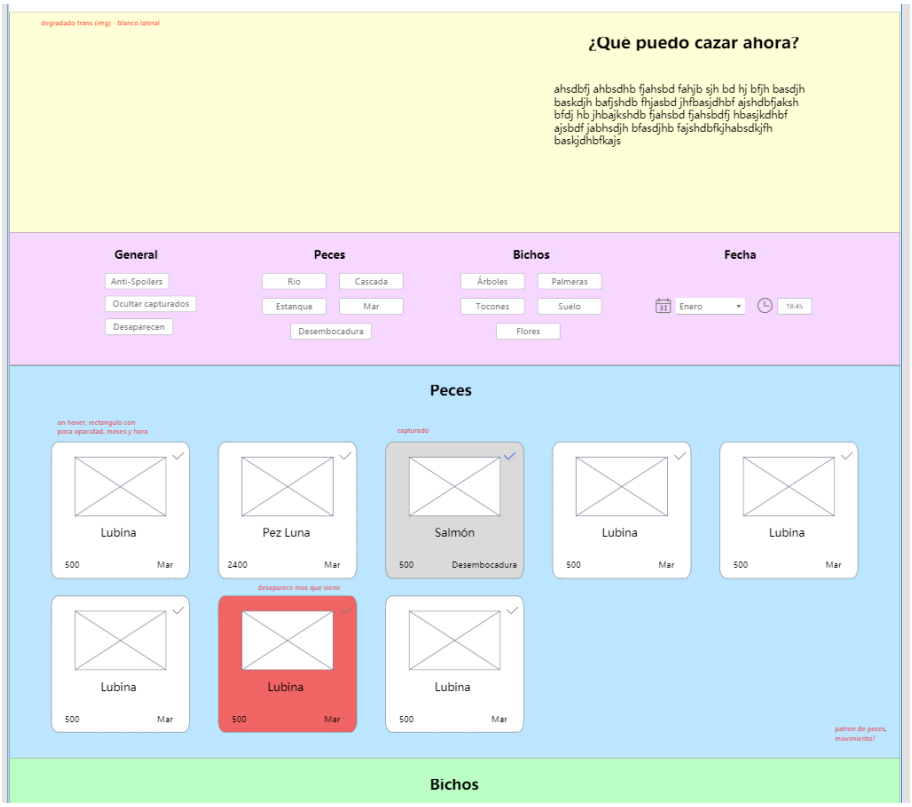


Figura 5.8: Cazar ahora 1

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.

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).

Figura 5.9: Cazar ahora 2

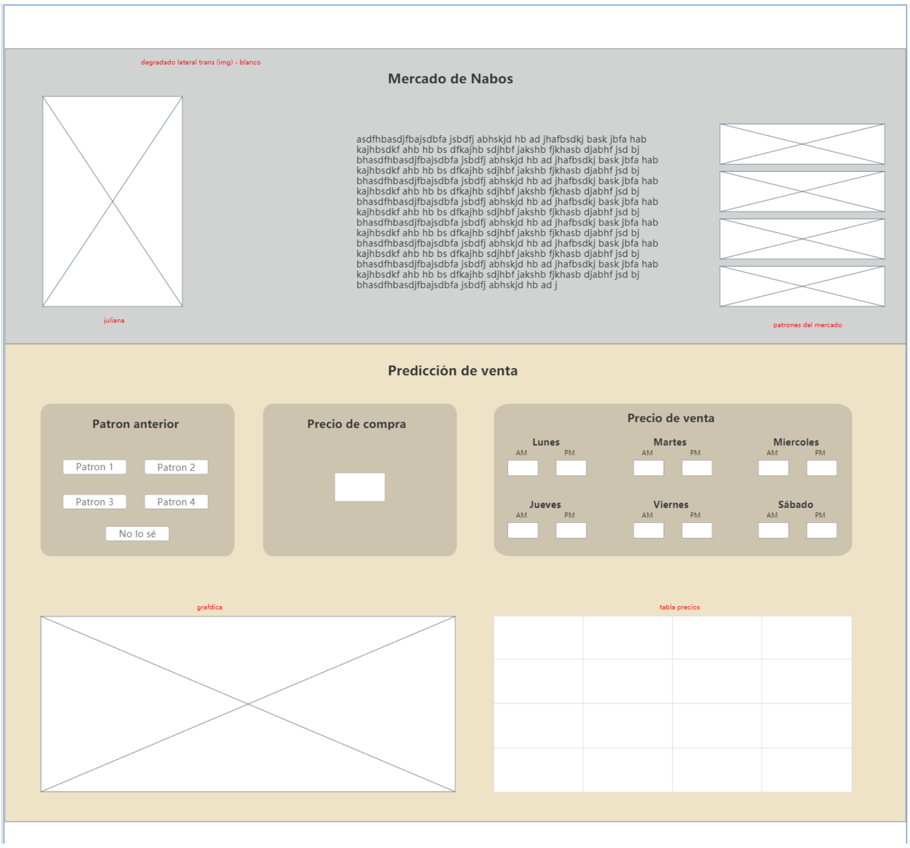


Figura 5.10: Calculadora de Nabos

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.

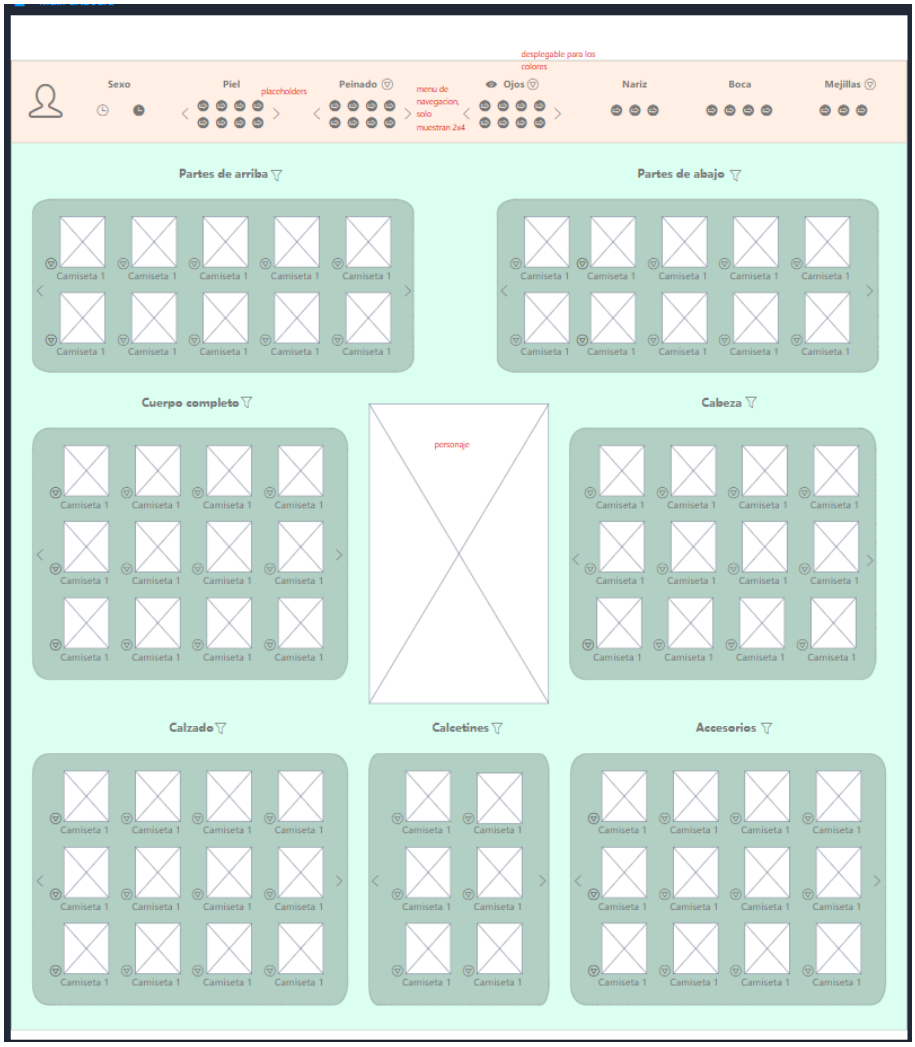


Figura 5.11: Probador

CAPÍTULO 6

Implementación

6.1– Entorno de desarrollo

Figura 6.1: Logotipo Visual Studio Code

El entorno de desarrollo utilizado para estra 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 de 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 lo 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 algun 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 fueramos a usar no iban a tener tanto peso para nosotros como lo iba a tener esta, por lo que ibamos 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 que hemos visto en la carrera y además hemos adquirido experiencia en Java durante todo 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 si 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 mas 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 raiz 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 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 (asi como la realización de los tests, ya que no tiene sentido desarrollar unos tests para features que no estan 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 sprint, 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.

Dado que no se había publicado el algoritmo, 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).

Respecto a la feature de "Colecciones Especiales", dado que no nos habíamos informado tanto como debíamos sobre la API que íbamos a usar, descubrimos que dicha API 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 catalogo de colecciones, de forma que con una sola petición por parte del administrador, se actualizan los items 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 ahorramos 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.

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 para acceder tanto a imágenes como a datos del juego.

Sin embargo, una vez nos pusimos a implementar 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 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 API, 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 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 ademas su uso era extremadamente sencillo.

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.

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 (Turnip Prophet) 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 limitado la personalización. 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 0.

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 muestan 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 fotográfica 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.

CAPÍTULO 7

Pruebas

CAPÍTULO 8

Comparación con otras alternativas

CAPÍTULO 9

Manual

Conclusiones y desarrollos futuros

Apéndices

Referencias

AUTORES, Varios. «Escuela Técnica Superior de Ingeniería Informática.», 2014. Fecha de consulta: 24 de Noviembre de 2014, URL `http://www.informatica.us.es`.

BEZOS, Javier. «The titlesec and titletoc Packages.» *TexEmplares*, 8, (2007), 283–298. CervanTEX, 2007.

DE SOUSA, José Martínez. *Ortografía y ortotipografía del español actual*. Trea, 2004.