PROYECTO FIN DE GRADO

ELDEN WIKI

MIGUEL ROBLA TOURIS

2º CFGS DESARROLLO DE APLICACIONES WEB

INDICE:

DESCRIPCIÓN 3
INTRODUCCIÓN AL JUEGO 3
Elden Ring 3
Conceptos del videojuego 4
ANÁLISIS DAFO 5
Inversion en implementación 6
ELABORACIÓN DEL PMV 7
DIAGRAMA GANTT 8
TECNOLOGÍAS UTILIZADAS 9
REQUISITOS 9
Usuarios 10
Objetos o información 10
CASOS DE USO 11
DESARROLLO BACK-END 16
Modelos de datos 17
Modelos de Usuarios 19
API (CONTROLADORES) 20
DESARROLLO FRONT-END 23
Wireframe 23
Diseño 31
POSIBLES AMPLIACIONES 33
Sección de comentarios 33
Barra de búsqueda 33
Ampliación de la base de datos 34
MANUAL DE USO 35
Corrección del sistema de puntuaciones de Windows 35
Popular la base de datos con información de ejemplo 37

INDICE DE IMAGENES:

Diagrama de Gantt 9
Diagrama de caso de uso 1 11
Diagrama de caso de uso 2 13
Diagrama de caso de uso 3 14
Diagrama de caso de uso 4 15
Diagrama de base de datos 17
Ejemplo de cláusulas de control de datos en los modelos 19
API, Index de Armas 20
API, ViewData de Armas 21
Clase RolesAutomaticos.cs 22
Adición de la cláusula AddRoles en los servicios del programa. 23
Adición de autorizaciones en la app. 23
Vista de Inicio formato escritorio 24
Vista de Inicio formato móvil 24
Vista de índice formato escritorio 25
Vista de índice formato móvil 25
Vista de índice de administrador formato escritorio 26
Vista de índice de administrador formato móvil 26
Vista de detalles formato escritorio 27
Vista de detalles formato movil 27
Vista de borrar formato escritorio 28
Vista de borrar formato móvil 28
Vista de crear/editar formato escritorio 29
Vista de crear/editar formato escritorio 29
Vista de login/registrarse formato escritorio 30
Vista de login/registrarse formato móvil 30
Ejemplo de menú en el videojuego 31
Ejemplo de vista en la web para comparación con el menú del juego 32
Panel de control, reloj y región 36
Cambio de los Símbolos realizado 36
Pantalla previa a la configuración adicional 36
Vista general de VisualStudio2019 37
Pantalla de selección y confirmación para publicar una aplicación de capa de datos 38
Publicación de los datos exitosa 38

DESCRIPCIÓN:

Página web que ejerce de compendio de objetos varios (armas, armaduras, consumibles, objetos, etc.) del videojuego Elden Ring, con sus datos pertinentes.

En este caso y para mayor facilidad de uso y testeo, hemos simplificado las propiedades de algunos objetos, reduciendo así la cantidad de entrada de datos para cada objeto, o al menos reduciendo la cantidad de datos obligatorios.

El proyecto explicado de manera genérica y simple es una plataforma de subida, modificación, visualización y borrado de información, con alguna de estas acciones restringidas según los permisos de cada usuario.

INTRODUCCIÓN AL JUEGO:

Elden Ring:

Elden Ring es el título del juego reciente sobre el que se basa este proyecto. Aunque reciente, se ha vuelo muy popular en poco tiempo. Se trata de un juego de temática medieval fantástica del estilo “Souls”, caracterizado por ser complicado, desafiante y por no llevar de la mano al jugador.

Este videojuego, además de lo anterior, trata de emular a los clásicos antiguos en algunos de sus aspectos. En concreto y la razón por la que esta página puede ser viable y rentable económicamente, no tiene ningún sistema de seguimiento de misiones, ni guía o compendio dentro del juego que permita conocer localizaciones de personajes, objetos, consumibles y demás. Incluso su mapa tiene una leyenda muy escasa y muy poco interactiva.

El juego quiere que el jugador tome notas cuando se le presenta con información crucial para llevar a cabo una misión, o direcciones para encontrar un objeto importante. El problema reside en que en el mercado del videojuego de hoy en día es muy común que se incluya alguna interfaz donde se puedan ver todos estos datos, por lo que la inmensa mayoría de jugadores no tomarán notas.

Además, hay que tener en cuenta que este juego, tratando de emular a los clásicos, es mucho más grande y denso que éstos, por lo que tienen demasiado contenido como

para que el jugador promedio se acuerde de todos los detalles necesarios Es por ello que muchos de estos jugadores acuden a guías online, video guías o Wikipedias específicas para su juego.

Conceptos del videojuego:

Para las posteriores explicaciones puede que haya ciertos conceptos que mucha gente que no acostumbre a jugar o no haya probado videojuegos actuales pueda no conocer, aquí ofreceremos ciertas definiciones útiles para la mejor comprensión del resto del documento:

- NPC (Non Playable Character): O en castellano, “Personaje No Jugable (PNJ)”. Es aquel personaje sobre el cual el jugador no tiene control. Suelen referirse a aquellos personajes controlados por la máquina que por defecto no son hostiles hacia el jugador. Hay que tener en cuenta que en Elden Ring, la inmensa mayoría de los NPCs se volverán hostiles hacia el jugador si éste los ataca de manera repetida, y son mortales, lo que quiere decir que pueden desaparecer del juego permanentemente, cerrando de manera abrupta posibles misiones y objetos ligados a dichas misiones.

- Estadísticas: Son propiedades que determinan las facultades del personaje. Por defecto y sin objetos, armas ni armaduras equipadas el personaje tendrá unas estadísticas base que el jugador podrá ir mejorando a medida que transcurre la historia principal. Esto es esencial para explicar el siguiente punto y entender algunas de las propiedades que tienen los objetos de tipo Arma y Armadura.

- Requisitos: Son las estadísticas mínimas que el jugador necesita poseer para poder equiparse una pieza de Armadura o Arma. De manera general a mejor sea el quipo, tendrá requisitos más altos. Elden Ring tiene como mecánica peculiaridades respecto a los requisitos de los objetos equipables;

    - Si un arma tiene requisitos de Fuerza, Destreza o ambas, se puede empuñar con las dos manos para reducir dicho requisito un 50%.
    - No hay una restricción total a la hora de equiparse armaduras, pero la velocidad a la hora de esquivar se verá altamente perjudicada si se sobrepasan ciertos requisitos: El jugador tiene un peso de carga máximo

que se puede sobrepasar y está determinado por el atributo de Resistencia. Si sobrepasa el 75% de esa carga, su velocidad de esquive y Estamina gastada al hacerlo se verá perjudicada, si sobrepasa el 100% de esa carga el jugador se verá aún más perjudicado.

- Escalado: Dependiendo de nuevo de las estadísticas, algunos atributos que le ofrecen las piezas de equipo al jugador se pueden ver multiplicadas si el escalado de éstas es de un alto nivel. En Elden Ring el rango de escalado viene determinado por las letras:

Rango	Escalado de daño
S	141% en adelante
A	101% - 141%
B	81% - 100%
C	51% - 80%
D	25% - 50%
E	1% - 24%

- Vitalidad / Maná / Estamina: Son los recursos del jugador, Los cuales pueden aumentar o disminuir según el equipo que se lleve. El único de estos recursos que se recupera de forma automática es la Estamina o Aguante. El resto se debe de recuperar mediante el uso de ciertos consumibles

ANÁLISIS DAFO:

Debilidades:

- Funcionalidad limitada.
- Proyecto muy especializado en un único videojuego.
- Necesario un buen posicionamiento SEO para poder generar unos ingresos notables por publicidad.

Amenazas:

- Gran competencia debido a la popularidad del videojuego.
- Relativamente débil contra ataques informáticos.
- Necesidad de actualizar las tecnologías una vez queden obsoletas.

Fortalezas:

- Tecnología muy eficiente y ágil.
- Diseño modular y creado con la temática del videojuego en mente.
- El diseño ha sido planteado de tal manera que los jugadores identifiquen que la página es sobre el videojuego tan solo con mirar su disposición y esquema de colores, esperando mejorar el número de clics y tiempo de navegación en la página.

Oportunidades:

- La popularidad creciente del videojuego atrae cada vez más visitas.
- El futuro lanzamiento de contenido descargable para el videojuego aumentará tanto el contenido de la página como el tráfico de ésta.
- Debido a la naturaleza anti intuitiva del juego, muchos de sus jugadores acudirán a servicios de información como es esta web.

INVERSION EN IMPLEMENTACIÓN:

Web a medida, sin uso de CRM		500€
Diseño a medida, sin uso de CRM		380€
Tamaño de más de 4 páginas	(descontando las 4 iniciales)	80€
Elaboración de bade de datos propia		410€
Registro de usuarios y control por roles		430€
Proyecto de cero, sin bocetos previos de cliente		180€
No incluye posicionamiento SEO		-
No incluye hosting		-
No incluye multi idioma		-

Total	2380€

Dado el caso de este proyecto, teniendo en cuenta el peso de sus fortalezas en comparación con sus debilidades, se puede ver que, por el precio establecido, éste es el momento de invertir aprovechando la popularidad del videojuego y el consecuente aumento en visitas y por tanto el aumento en rentabilidad.

ELABORACIÓN DEL PMV:

El único coste del PMV en este caso sería el alquiler de un dominio o hosting para el acceso a la web. Un ordenador personal serviría como servidor para esta prueba y el resto es trabajo realizado por el desarrollador, sin necesidad de contratar a terceros.

El coste más barato de alquiler de dominio encontrado es para los dominios ".com", costando durante todo el primer año 9,99€. Aunque el servicio de Microsoft Azure nos proporciona un hosting temporal gratuito durante 12 meses.

El beneficio del PMV sería bajo ya que, al ser probado con una pequeña cantidad de clientes, el tráfico de la página web no generaría apenas beneficios por anuncios. El proyecto en ese momento se sostendría de mejor o peor manera en función de la popularidad del videojuego sobre el que se está realizando la wiki.

El diseño sería más esquemático y será menos agradable visualmente que el producto final, pero su funcionalidad y usabilidad será total. La intención es que la fiabilidad del producto sea completa, pero un proyecto web siempre se puede ir puliendo, corrigiendo y mejorando cuantas más personas lo prueben, por lo que la mejora de la fiabilidad es totalmente posible durante la fase de pruebas del PMV.

Se tratará de realizar encuestas a aquellos que visiten la página, para conocer el público que está interesado en este tipo de servicios y poder adecuar la página lo máximo posible a ese tipo de usuarios.

En la encuesta se preguntará al usuario si conoce de otras páginas que ofrezcan servicios similares, con el fin de estudiar dichas páginas y obtener ideas para realizar una competencia a estas (Ejemplo: Fandom Wiki).

DIAGRAMA GANTT:

En este apartado se muestra visualmente el progreso del proyecto en cuestión al tiempo invertido para alcanzar cada hito propuesto en el desarrollo de la aplicación.

El margen de tiempo total ha sido de aproximadamente cuatro meses, debido al cambio total de proyecto en medio del desarrollo de éstos un total de tres veces, lo que obligó a comenzar de nuevo todo el proceso.

La precisión del eje correspondiente al tiempo en este diagrama es con un margen de media semana, es decir cada mes es divisible aproximadamente en 8 partes. Y como se puede apreciar algunas líneas en este eje se han solapado, esto quiere decir que se ha estado trabajando simultáneamente en varias tareas para cumplir diferentes hitos del proyecto.

El trabajo se ha dividido en ocho hitos principales, con diferentes sub hitos dentro de cada uno, que no se ven reflejados en este diagrama. Una vez finalizado un hito, si se tuviese que retocar en algún momento, ese tiempo invertido caería bajo la clasificación “detección y depuración de problemas”. Este apartado o hito se extiende en la mayoría de la línea temporal ya que como requisito para cumplir algunos hitos era necesario retocar, modificar o refactorizar el contenido de otros hitos ya finalizados previamente.

A su vez, el apartado de “Documentación” también se extiende por la mayoría de la línea ya que el categorizar y plasmar el progreso de la aplicación en la documentación, así como modificar la información ya existente, ha sido un trabajo prácticamente diario, aquí también se incluye el tiempo usado para preparar la presentación para la defensa del proyecto.

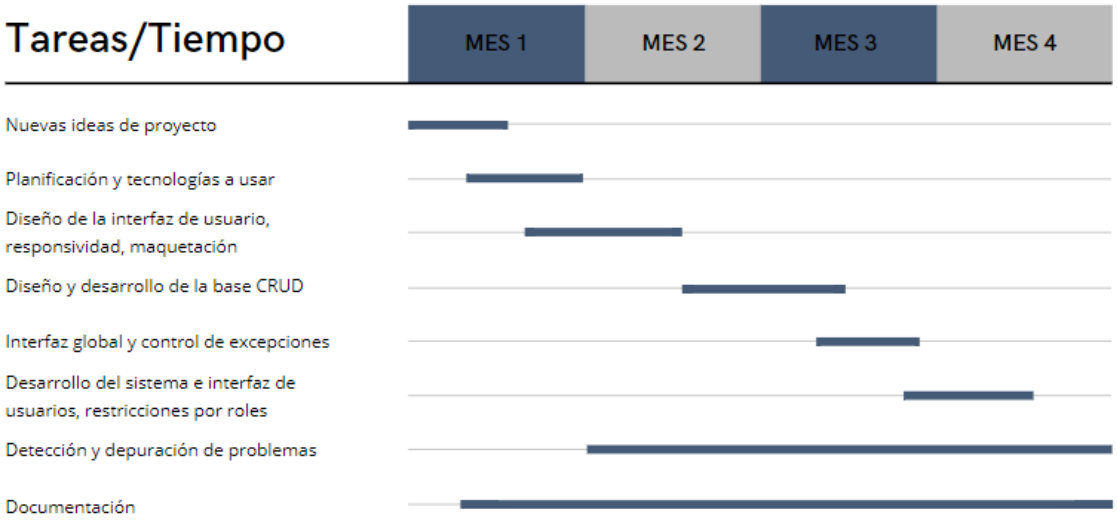

Ilustración 1 Diagrama de Gantt

TECNOLOGÍAS UTILIZADAS:

Al ser un servicio principalmente informativo, en el que se requiere de la posibilidad de creación, edición y vista en detalle de diversos elementos, además de necesitar algún tipo de autenticación para que el sistema determine quién tiene permisos para realizar las acciones previamente expuestas, nos decantamos por:

- Tecnologías ASP.NET Core con Modelo-Vista-Controlador.
- Para la base de datos se usa SQL Server Express.
- Para la vista y front-end se usará HTML, CSS y JavaScript nativos, además de algunas propiedades de la librería de Bootstrap y algunos pedazos de código incrustado en C#.
- El proyecto entregado no estará desplegado en un hosting, pero en caso de hacerlo en un futuro, se haría uso de Google AdSense y AmazonAds para los espacios publicitarios. Que permiten mostrar anuncios personalizados según la información del usuario.

REQUISITOS:

Usuarios:

Se registrarán usuarios:

- El usuario podrá registrarse con una cuenta.
- El usuario podrá cancelar su cuenta, eliminándola de la base de datos.
- El usuario deberá confirmar su cuenta accediendo al correo.

Para registrarse, el usuario deberá introducir:

- Correo electrónico.
- Contraseña.
- Confirmación de contraseña.

El usuario podrá tener los roles de:

- Administrador: Puede realizar todas las acciones permitidas en la web.
- Usuario registrado: Puede visualizar y modificar la información y sus detalles.
- Usuario anónimo: Sólo puede visualizar la información y sus detalles.

Objetos o información:

Existirá un índice de objetos para cada clase de objeto. Donde se podrán visualizar todas las entradas de objetos para dicha clase. Dichas clases serán a su vez las opciones de acceso del menú de la página.

Cada objeto tendrá las siguientes vistas:

- Creación: Exclusiva para el usuario de tipo Administrador, permite generar un nuevo objeto de la clase donde nos encontremos. Los campos introducidos serán analizados para cumplir los modelos establecidos y no generar conflictos en la base de datos.
- Detalles: De libre acceso para todos los tipos de usuarios, permite visualizar más detalles del objeto de los que aparecen en su índice correspondiente.
- Edición: Exclusiva para el usuario Administrador y Registrado, permite la edición de las características de los objetos ya existentes. Los campos modificados se analizarán como en el apartado de Creación.

- Borrado: Exclusivo para el usuario de tipo Administrador, permite visualizar toda la información detallada y posibles consecuencias antes de borrar de manera permanente un objeto de la página.

CASOS DE USO:

diagramas

CASO 1; El administrador crea o modifica un objeto en la página:

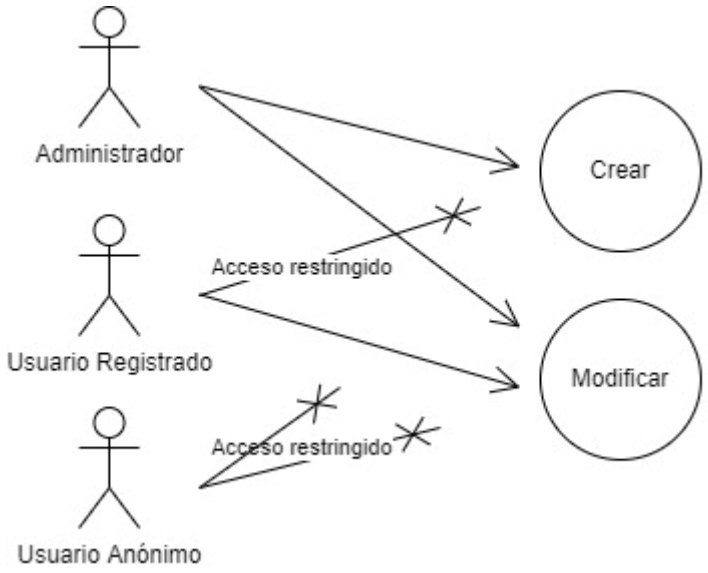

Ilustración 2 Diagrama de caso de uso 1

Precondiciones:

- El usuario Administrador ya tiene la cuenta creada y sesión iniciada.
- En caso de crear un objeto de tipo “Arma”, el usuario debe haber creado previamente al menos un objeto del tipo “TipoArma” y un objeto de tipo “Ceniza”.

Flujo principal:

- El usuario se situará en el índice del tipo de objeto donde quiere añadir una nueva entrada. En caso de modificarlo se situará en la página de detalles del objeto a modificar.

- Clicará en el botón “Crear nuevo/a X” situado debajo del título de la página, que a su vez le indica el objeto a crear. En caso de modificarlo clicara en el botón “Editar” que aparece justo encima de la imagen en la página de detalles.
- Accederá a una nueva página, exclusiva para Administradores, donde introducirá los datos pertinentes y necesarios para la creación o modificación del objeto.
- Para añadir la imagen, previamente se debe de alojar dicha imagen dentro de los archivos del proyecto, siendo su ruta:
    - NombreProyecto\wwwroot\img\Tipoarma\imagen.jpg
- El nombre, descripción, imagen y ciertos atributos del objeto serán de inserción obligatoria.
- Para finalizar, el usuario clica en “Crear nuevo” o “Guardar cambios”, guardando el objeto en la base de datos y mostrándolo al resto de usuarios.

Flujos alternativos:

- El usuario no completa los campos obligatorios o algún campo completado no es del tipo de dato necesario:
    - aparecerá un mensaje de error en las casillas de entrada que no estén completadas, o en las que estén completadas con el tipo de dato equivocado o fuera de su rango.

CASO 2; El administrador borra un elemento:

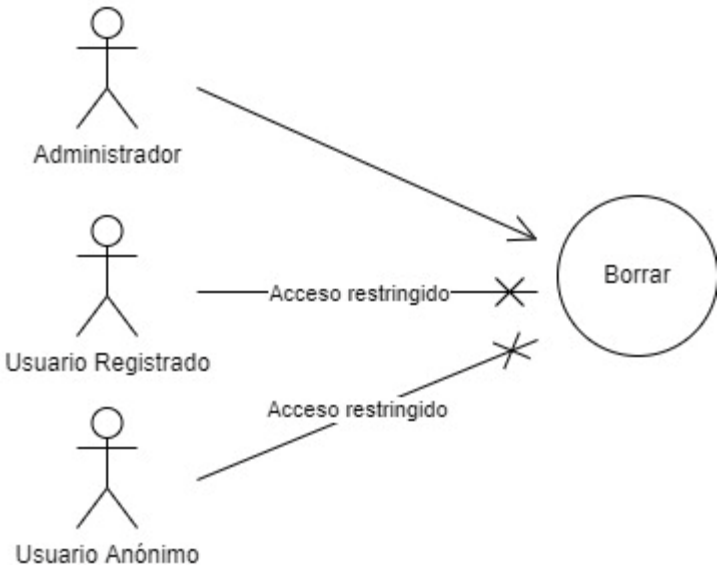

Ilustración 3 Diagrama de caso de uso 2

Precondiciones:

- El usuario está con la sesión de administrador iniciada.
- Ya hay objetos creados y visibles en el índice de objetos a borrar.

Flujo principal:

- El usuario clicará en el botón “Borrar” que aparecerá en el índice, situado debajo del elemento que se quiera borrar.
- Podrá visualizar los detalles del objeto antes de confirmar el borrado, junto con un mensaje de alerta que advierte de que el borrado es permanente e irreversible.
- El usuario clica en “Confirmar borrado”, borrando el objeto y siendo redirigido de nuevo al índice de objetos, que ya no mostrará el ítem borrado.

Flujos alternativos:

- El usuario cambia de opinión y quiere volver al índice sin borrar nada:
    - Clicará en el botón “Cancelar” y será redirigido al índice con el objeto intacto.

- El usuario decide borrar un elemento del tipo Ceniza o TipoArma:

- El proceso será el mismo que el flujo principal con la salvedad de que el mensaje de alerta también le advertirá sobre un posible borrado de objetos en cascade debido a las dependencias del modelo.
- Si decide confirmar el borrado, se eliminarán también todos los objetos asociados al elemento recién borrado.

CASO 3; El usuario cambia la contraseña de su cuenta:

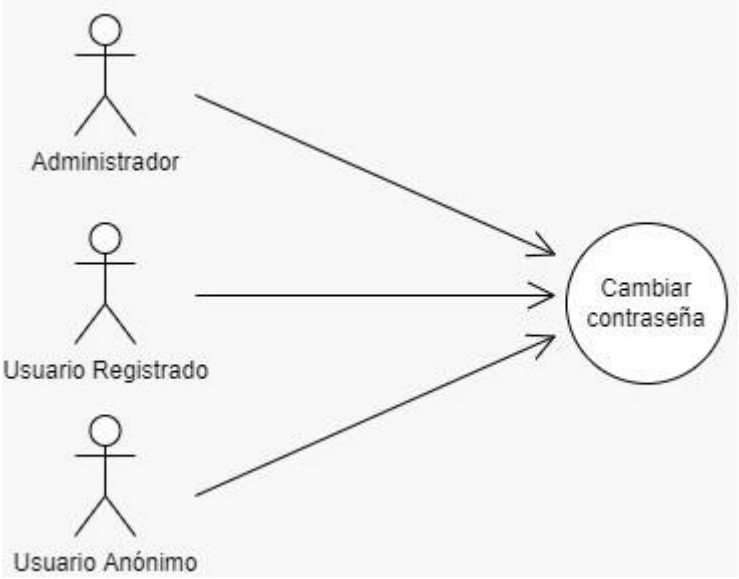

Ilustración 4 Diagrama de caso de uso 3

Precondiciones:

- El usuario tiene una sesión iniciada con su cuenta.

Flujo principal:

- El usuario clica en su nombre de cuenta situado en la parte derecha del menú, redirigiéndolo a su pantalla de ajustes de cuenta.
- Escogerá la opción “Cambiar contraseña” del submenú de la pantalla.
- Se carga un formulario donde se pedirá la antigua contraseña por seguridad, así como la nueva, que tendrá que repetir dos veces.
- El usuario clica en “Cambiar contraseña”, en ese momento, si todo ha ido bien, se le redirige a inicio y la contraseña se actualiza.

Flujos alternativos:

- El usuario no escribe la antigua contraseña correctamente, no rellena los campos obligatorios o no escribe la nueva contraseña igual en los dos campos en los que se solicita:
    - Al clicar en el botón de cambio de contraseña aparecerán mensajes de error debajo de los campos correspondientes, indicando qué ha ido mal y previniendo el cambio de la contraseña.

- El usuario cambia de opinión y decide no cambiar su contraseña:
    - El usuario es libre de clicar en cualquier opción del menú para salir del formulario de cambio de contraseña en cualquier momento.

CASO 4; El usuario decide borrar su cuenta:

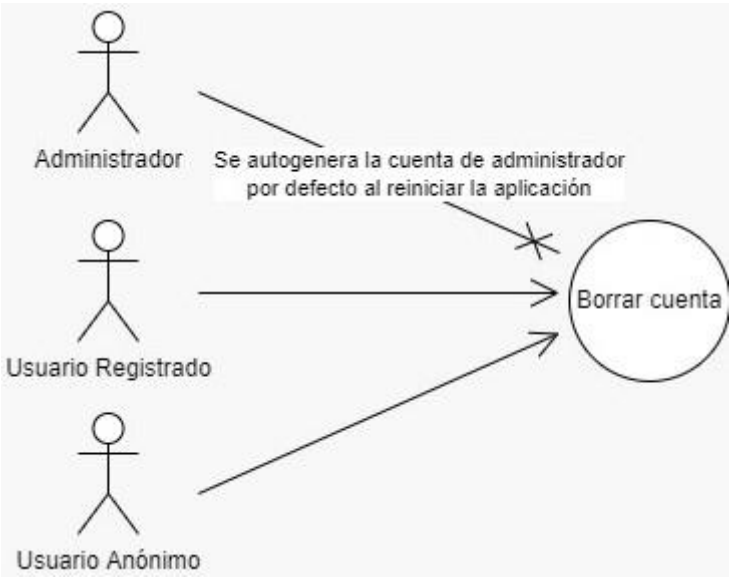

Ilustración 5 Diagrama de caso de uso 4

Precondiciones:

- El usuario tiene una sesión iniciada con su cuenta.

Flujo principal:

- El usuario clica en su nombre de cuenta situado en la parte derecha del menú, redirigiéndolo a su pantalla de ajustes de cuenta.
- Escogerá la opción “Borrar cuenta” del submenú de la pantalla.
- Se cargará una nueva pantalla que le recuerda que está a punto de borrar su cuenta y datos asociados permanentemente, para confirmar el borrado se le pedirá que escriba su contraseña como medida extra de seguridad.
- El usuario clica en “Confirmar borrado”, en ese momento su cuenta ya no se encuentra en la base de datos y se le redirige a la pantalla de inicio como usuario anónimo.

Flujos alternativos:

- El usuario escribe mal su contraseña:
    - Aparecerá un mensaje de error debajo del cambo rellenado y se el impedirá continuar con el borrado hasta acertar la contraseña.
- El usuario cambia de idea y decide no eliminar su cuenta:
    - El usuario es libre de clicar en cualquier opción del menú para salir de la pantalla de confirmación de borrado en cualquier momento.

DESARROLLO BACK-END:

En este proyecto creamos cinco tablas de datos; Armas, Armaduras, Consumibles, Cenizas y TipoArmas, la última tabla es una tabla relacional, TipoArmaViewModel, que nos permite crear una relación entre las Armas, las Cenizas que usan y el tipo de arma que es cada objeto de la tabla Armas. Además hacemos uso de otras tres tablas de AspNet Core Identity. A continuación explicamos cada tabla y sus relaciones basándonos en el siguiente diagrama:

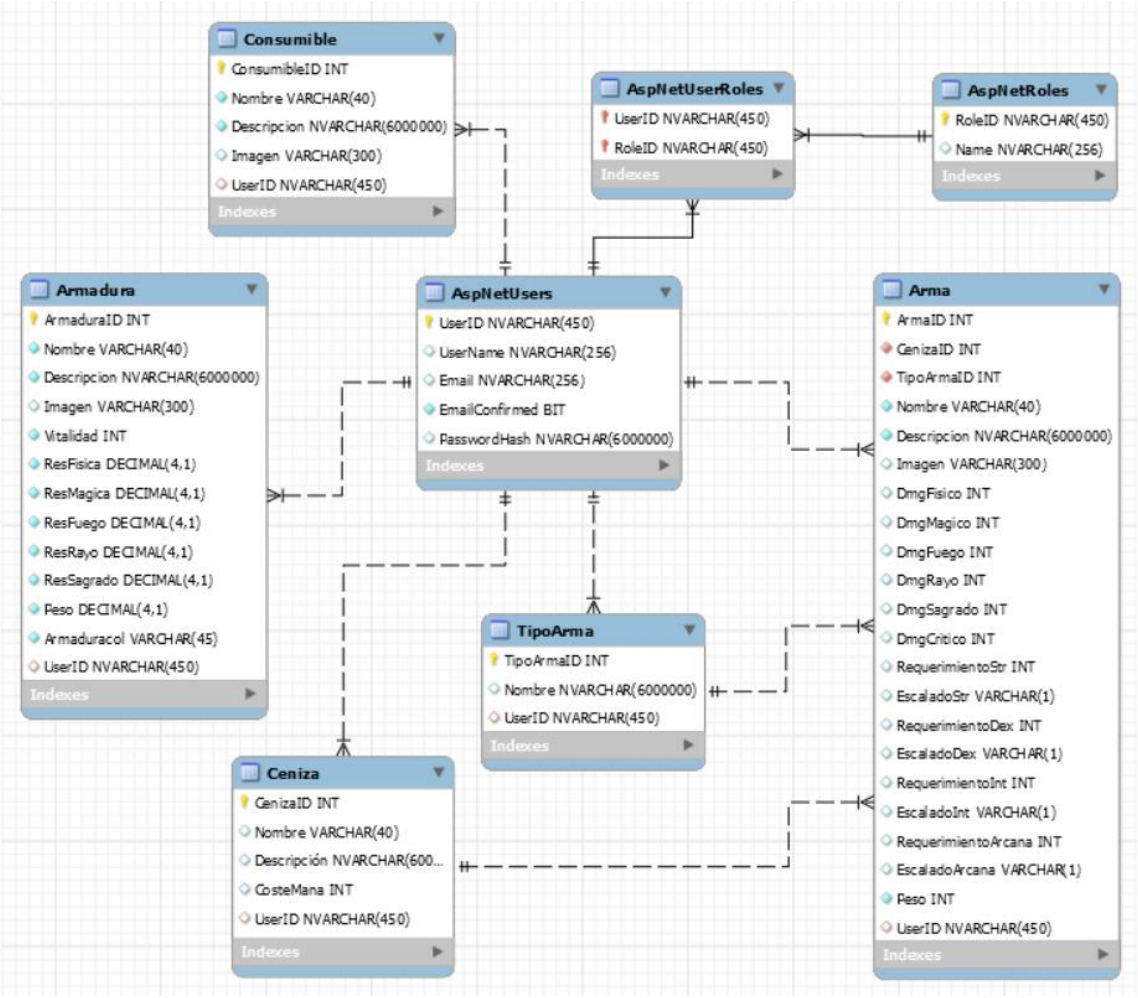

Ilustración 6 Diagrama de base de datos

Modelos de datos:

Cada tabla tiene por supuesto un campo ID que hace función de clave primaria, pero no todas las tablas tienen claves foráneas ni relaciones entre sí, salvando a la relación necesaria con la de usuarios para que éstos puedan interactuar con la página. El control de tipos de datos y sus valores se realizan en su mayoría en los modelos en sí, mediante instrucciones propias de .Net. Comenzaremos por las tablas de datos del contenido a visualizar.

La tabla de Armas es la que más campos tiene, esto es debido a la naturaleza y características de las armas en este videojuego, hemos simplificado la información real del videojuego para no añadir demasiada información y agilizar tanto la creación del modelo como de los objetos a mostrar desde la interfaz a posteriori. Esta tabla a su vez

se relaciona con las tablas TipoArmas y Cenizas, cuyos datos se usan como campos obligatorios en esta tabla.

La relación está creada pensando en que cada arma debe tener obligatoriamente un tipo de arma y una ceniza asignadas. Un elemento de TipoArma puede tener varias armas asignadas, pero un arma solo puede tener un tipo de arma asignado, lo mismo ocurre con las cenizas.

Para ello se crea la tabla relacional, donde se almacena un listado de los elementos almacenados en las tablas TipoArma y Cenizas, además del ID de la Arma asociada que hace función de clave primaria, para su posterior uso a la hora de crear un nuevo objeto de tipo Arma. Veremos más adelante como hemos restringido en el front-end la selección de estas dos propiedades.

Respecto a las tablas de Armaduras y Consumibles, éstas no precisan de las propiedades Cenizas ni TipoArma, por lo que son independientes y no tienen una conexión relacional.

La tabla de Armaduras gana el segundo puesto en densidad, pero es más simple que la anterior ya que tal como hemos dicho no requiere de relaciones con otras tablas.

lo mismo ocurre con la tabla de Consumibles, que a su vez tiene aún menos campos a rellenar.

Para todas estas tablas, el control de inserción de los datos de forma obligatoria o del control de campos obligatorios se ha hecho desde el modelo para mayor seguridad y personalización. A continuación se puede ver una captura de uno de estos modelos como ejemplo, para mostrar las cláusulas utilizadas para dicho control de datos:

```
[Display(Name = "Requerimiento de Inteligencia")]
[Range(0, 99, ErrorMessage = "El valor debe estar entre 0 y 99")]
12 referencias
public int? RequerimientoInt { get; set; }
[Display(Name = "Escalado de Inteligencia")]
[RegularExpression(@"^[SABCDE]$", ErrorMessage = "Las opciones son: S,A,B,C,D y E.")]
12 referencias
public char? EscaladoInt { get; set; }


[Display(Name = "Requerimiento de Fe")]
[Range(0, 99, ErrorMessage = "El valor debe estar entre 0 y 99")]
12 referencias
public int? RequerimientoFe { get; set; }
[Display(Name = "Escalado de Fe")]
[RegularExpression(@"^[SABCDE]$", ErrorMessage = "Las opciones son: S,A,B,C,D y E.")]
12 referencias
public char? EscaladoFe { get; set; }


[Display(Name = "Requerimiento de Arcana")]
[Range(0, 99, ErrorMessage = "El valor debe estar entre 0 y 99")]
12 referencias
public int? RequerimientoArcana { get; set; }
[Display(Name = "Escalado de Arcana")]
[RegularExpression(@"^[SABCDE]$", ErrorMessage = "Las opciones son: S,A,B,C,D y E.")]
12 referencias
public char? EscaladoArcana { get; set; }

[Column(TypeName = "decimal(4,1)")]
[Range(0, 200, ErrorMessage = "El valor debe estar entre 0 y 200")]
[Required(ErrorMessage = "Campo obligatorio")]
10 referencias
public decimal Peso { get; set; }
```

Ilustración 7 Ejemplo de cláusulas de control de datos en los modelos

Como se puede observar, se ha hecho uso principalmente de las cláusulas Required, RegularExpression y Range, con sus respectivos ErrorMessage para sobrescribir los mensajes de alerta predeterminados en inglés.

Modelos de Usuarios:

Para el control de usuarios, se ha utilizado la tecnología de .net Identity, que nos facilita muchas medidas de control y seguridad, entre ellas se destaca:

- Control de usuario persistente.
- Prevención de inyección SQL.
- Protección contra DDoS.
- Confirmación de cuentas mediante correo electrónico.
- Protección de datos sensibles mediante el uso de claves hash.

- Uso de inicios de sesión externos, permite el uso de cuentas de Facebook, Google, y demás cuentas para iniciar sesión de manera directa.
- Control de acceso a partes de la página por parte del usuario mediante la asignación de roles.

En nuestro caso nos centraremos sobre todo en este último punto. Para la creación y asignación de roles a usuarios usaremos principalmente tres tablas; AspNetUsers, AspNetRoles y la tabla relacional AspNetUserRoles.

No haremos cambios directos en los modelos de estas tablas, si no que las modificaremos y automatizaremos tareas mediante la modificación de controladores y adición de algunas clases y funciones creadas desde cero, lo cual veremos en el siguiente apartado.

API (CONTROLADORES):

La mayoría de los métodos generados aquí son los autogenerados para index, get, post, edit y details (CRUD) por MVC mediante el uso de scaffolding. La mayor cantidad de modificación se ha realizado en el controlador de Armas, ya que necesita de cambios para poder utilizar los datos de Cenizas y TipoArma en sus objetos:

```
public async Task<IActionResult> Index()
{
    //SE GENERAN TANTAS LISTAS COMO PROPIEDADES FORANEAS TENGAMOS EN EL MODELO
    IQueryable<String> ListaTipos = from t in _context.TipoArma
                                    orderby t.Nombre
                                    select t.Nombre;

    IQueryable<String> ListaCenizas = from c in _context.Ceniza
                                      orderby c.Nombre
                                      select c.Nombre;

    //SE CREA UNA LISTA CON TODOS LOS OBJETOS DEL MODELO
    var ListaArmas = from a in _context.Arma
                     select a;

    //SE INCLUYEN EN DICHA LISTA LAS PROPIEDADES FORÁNEAS
    ListaArmas = ListaArmas.Include(a => a.Ceniza).Include(a => a.TipoArma);

    //SE CREA UNA VARIABLE DEL TIPO MODELO IGUAL QUE EL QUE SE USA EN LA VISTA ASOCIADA, Y SE INTRODUCEN LAS TRES LISTAS ANT
    var ArmaViewModel = new ArmaViewModel
    {
        TipoArma = new SelectList(await ListaTipos.Distinct().ToListAsync()),
        Ceniza = new SelectList(await ListaCenizas.Distinct().ToListAsync()),
        Arma = await ListaArmas.ToListAsync()
    };

    //SE RETORNA DICHA VARIABLE
    return View(ArmaViewModel);

    //var applicationDbContext = _context.Arma.Include(a => a.Ceniza).Include(a => a.TipoArma);
    //return View(await applicationDbContext.ToListAsync());
}
```

Ilustración 8 API, Index de Armas

En el Index debemos almacenar en una lista todos los objetos del modelo junto a sus propiedades foráneas, para poder usar y mostrar todo ello en la vista, ya que en ésta no se usará el modelo de Armas, si no el ArmaViewModel. Esto ocurre también en las vistas de creación y modificación.

```
// POST: Armas/Create
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles="Administrador")]
0 referencias
public async Task<IActionResult> Create([Bind("ArmaID,Nombre,Descripcion,Imagen,TipoArmaID,CenizaID,DmgFisico,DmgMagico,DmgFuego,Dm
{
    if (ModelState.IsValid)
    {
        _context.Add(arma);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["CenizaID"] = new SelectList(_context.Ceniza, "CenizaID", "CenizaID", arma.CenizaID);
    ViewData["TipoArmaID"] = new SelectList(_context.TipoArma, "TipoArmaID", "TipoArmaID", arma.TipoArmaID);
    return View(arma);
}
```

Ilustración 9 API, ViewData de Armas

También, a la hora de crear y editar objetos correspondientes a este modelo, necesitamos unos ViewData de las dos propiedades foráneas, para poder hacer la selección de éstas en sus vistas correspondientes.

Aprovechando la captura anterior, se puede ver que utilizamos la cláusula “Authorize” para restringir la entrada a la vista desde el código del controlador según el rol del usuario. Esto se hace ya que, aun no mostrándole el enlace de acceso a estas vistas desde la interfaz mediante el uso de código que veremos en el siguiente apartado, si el usuario escribiese la URL correcta de acceso directamente desde la barra de búsqueda del navegador, podría acceder sin problemas. Estos “Authorize” en el controlador hacen que dichos usuarios sean redirigidos a una página de alerta.

Otra modificación significativa se ha realizado para la creación y asignación de roles, así como la creación automática del usuario administrador al iniciar la página. De este modo, aún si se reinicia la página de cero, se pierde la base datos por alguna razón, o se migra a otro servidor sin los datos de los usuarios, este usuario administrador persistirá con el mismo nombre y contraseña.

Para conseguir todo esto, primero se necesita modificar el archivo Startup.cs del proyecto, pero antes crearemos en un archivo aparte las funciones necesarias para crear tanto roles como usuarios, y para asignar los roles a los usuarios. Este archivo es CreacionRolesUsuarios.cs, y lo generamos fuera del Startup.cs con la idea de dejar el código de éste lo más limpio posible, ya que es un archivo fundamental para la correcta ejecución del proyecto.

```
public static class RolesAutomaticos
{
    1 referencia
    public static void EjecutarRoles(UserManager<IdentityUser> userManager, RoleManager<IdentityRole> roleManager)
    {
        InicializarRoles(roleManager);
        InicializarUsuarios(userManager);
    }

    1 referencia
    private static void InicializarRoles(RoleManager<IdentityRole> roleManager)
    {
        //Si no existe aún el rol de Administrador
        if (!roleManager.RoleExistsAsync("Administrador").Result)
        {
            //Creamos el objeto rol y lo añadimos a la listas de roles
            var rol = new IdentityRole
            {
                Name = "Administrador"
            };
            //Es necesario almacenar el rol en una variable .Result
            var result = roleManager.CreateAsync(rol).Result;
        }
        //Hacemos lo mismo con el rol Usuario
        if (!roleManager.RoleExistsAsync("Usuario").Result)
        {
            var rol = new IdentityRole
            {
                Name = "Usuario"
            };
            var result = roleManager.CreateAsync(rol).Result;
        }
    }

    1 referencia
    private static void InicializarUsuarios(UserManager<IdentityUser> userManager)
    {
        //Si aún no existe la cuenta administrador
        if(userManager.FindByNameAsync("admin1@administrador.admn").Result == null)
        {
            //Creamos el objeto cuenta
            var admin = new IdentityUser
            {
                UserName = "admin1@administrador.admn",
                Email = "admin1@administrador.admn",
                EmailConfirmed = true
            };
            //Lo añadimos a la lista de usuarios junto a una contraseña
            var result = userManager.CreateAsync(admin, "C@ntrasenia_1").Result;
            //Si se crea correctamente, asignamos a la cuenta nueva el rol de administrador
            if (result.Succeeded)
            {
                userManager.AddToRoleAsync(admin, "Administrador").Wait();
            }
        }
    }
}
```

Ilustración 10 Clase RolesAutomaticos.cs

En este archivo creamos una clase estática para poder llamarla en otras clases sin haberla instanciado previamente. Dentro de esta clase creamos tres funciones, la función InicializarUsuarios crean los usuarios y los roles cuando éstos sean necesarios y asigna los roles pertinentes, la función InicializarRoles Crea los objetos necesarios en la lista de roles cuando se detecta que aún no existen. La otra función, EjecutarRoles, simplemente llama a las otras dos asignándoles los parámetros pertinentes.

Ahora bien, para que estas funciones se ejecuten correctamente, debemos importar algunas librerías y modificar funciones en el Startup.cs para agregar los roles, las modificaciones a realizar son la adición de AddRoles en services, para poder crear

nuevos roles nosotros mismos, y UseAuthorization en la app, para poder restringir el acceso a vistas según dichos roles:

```
services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

Ilustración 11 Adición de la cláusula AddRoles en los servicios del programa.

```
app.UseAuthentication();
app.UseAuthorization();
```

Ilustración 12 Adición de autorizaciones en la app.

Esto en conjunto hace que se cree un usuario administrador por defecto con el rol de Administrador, y que a todos los demás usuarios creados mediante la interfaz se les asigne el rol de Usuario. Estos roles permiten y restringen el acceso a diferentes recursos y vistas de la página, veremos cómo se ha conseguido en el siguiente apartado.

DESARROLLO FRONT-END:

Wireframe:

Se ha tratado de mantener una disposición similar a la que muestra el videojuego para que la interfaz sea familiar para los jugadores:

- Inicio:

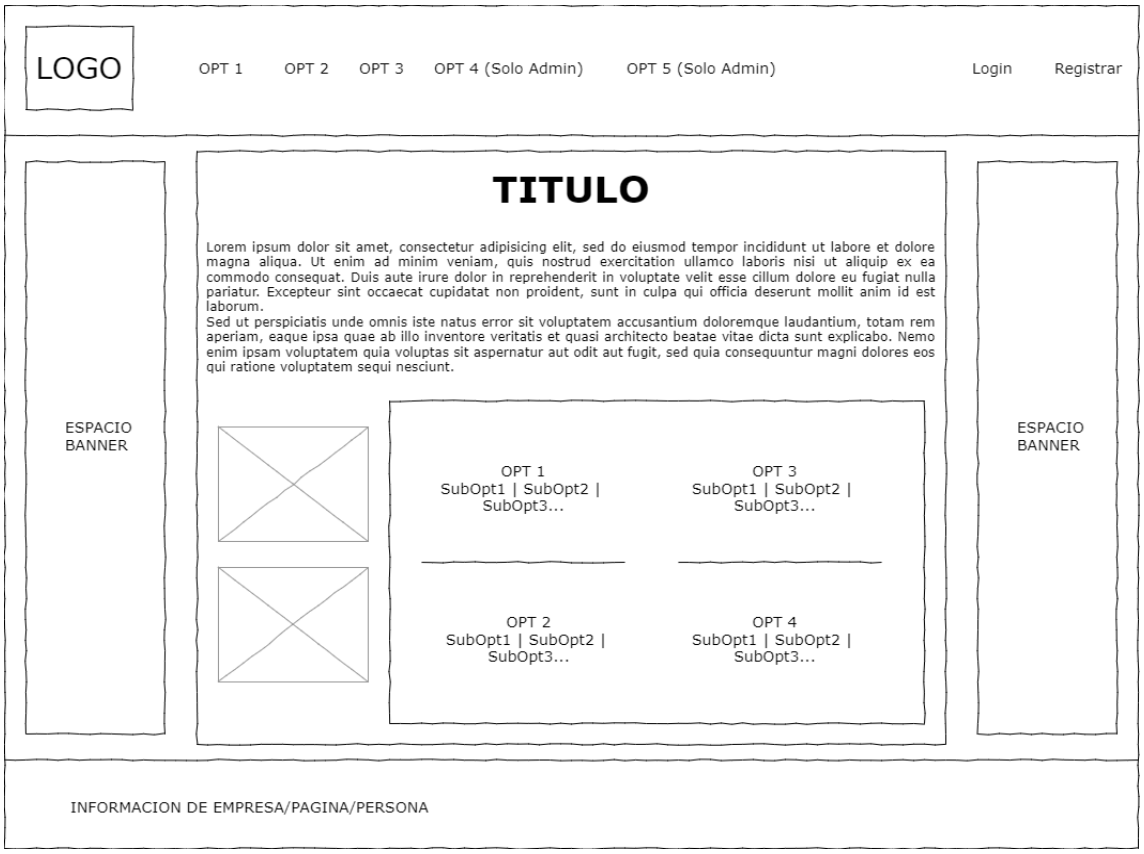

Ilustración 13 Vista de Inicio formato escritorio

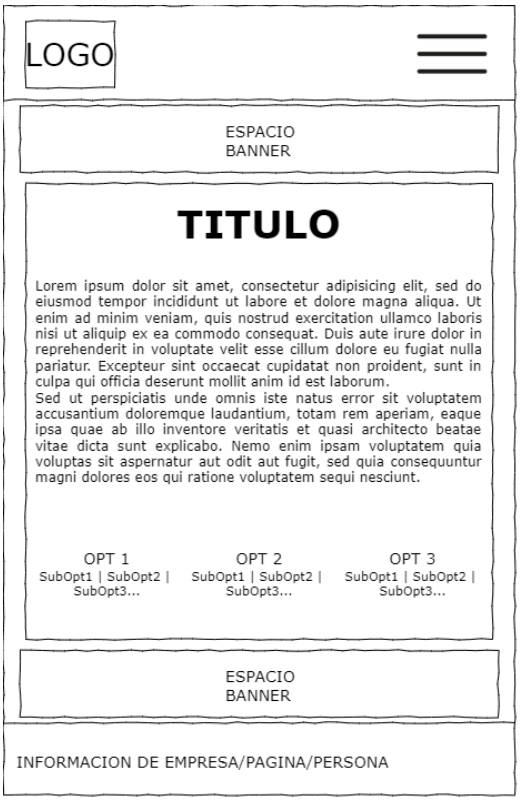

Ilustración 14 Vista de Inicio formato móvil

La página de inicio es sencilla, con el nombre de la página de título, seguido de unos párrafos de texto descriptivos que explican tanto el videojuego como la función que realiza la página.

La idea inicial era añadir unas imágenes del videojuego a un lateral y añadir accesos directos a los apartados del sitio web en forma de bloque, pero esto acabó descartándose ya que las imágenes no se visualizaban bien al ser demasiado pequeñas y confundirse con la imagen de fondo.

Acabamos decantándonos por modificar la vista de móvil y aplicarla también al modo escritorio.

- Index general:

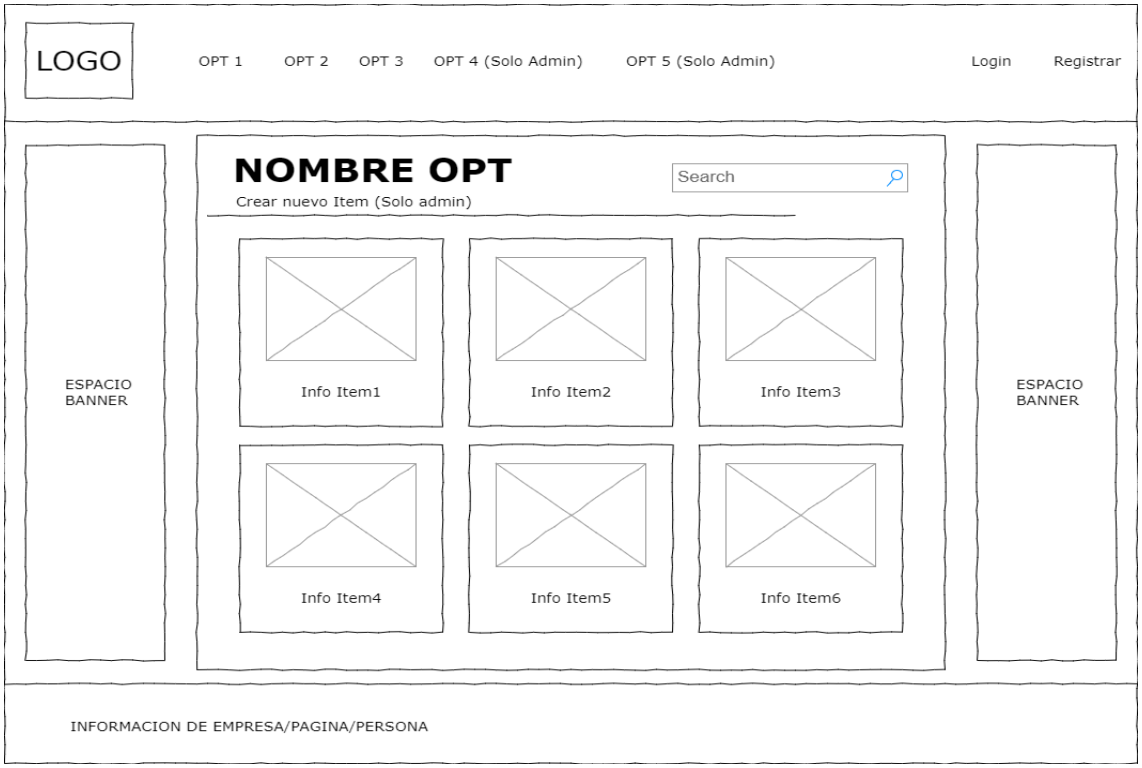

Ilustración 15 Vista de índice formato escritorio

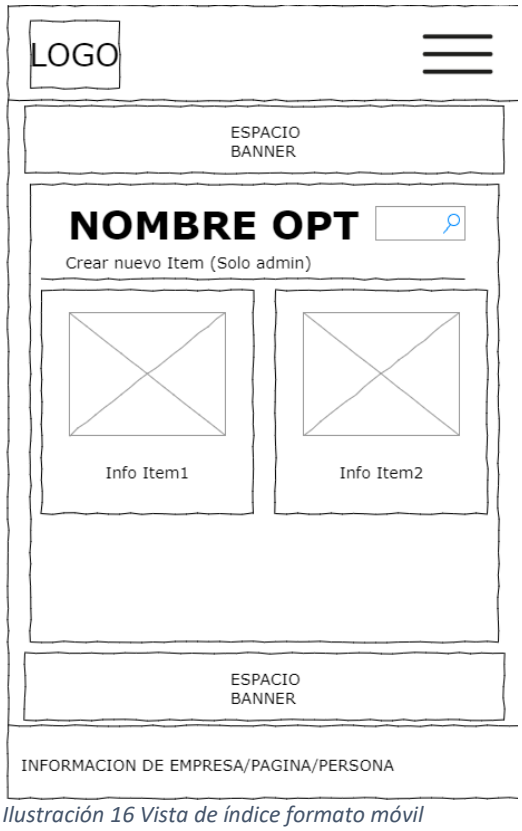

Ilustración 16 Vista de índice formato móvil

La página de los índices, aplicable a todos los apartados visibles por cualquier usuario, tiene un formato de lista por bloques, donde cada imagen irá acompañada de un nombre e información general si es necesaria.

Esta información, además de descriptiva, hace función de enlace para acceder a los detalles del objeto. Clicar en la imagen redireccionará al usuario al mismo lugar que clicando en el nombre, pero puede que clicar en otra información nos lleve a otras secciones.

La vista índice de Armas sufre un ligero cambio, en esta vista separamos las armas por tipos de armas, por lo que antes de comenzar con cada bloque de ítems primero aparece un titulo anunciando la categoría a la que pertenecen.

- Index administradores:

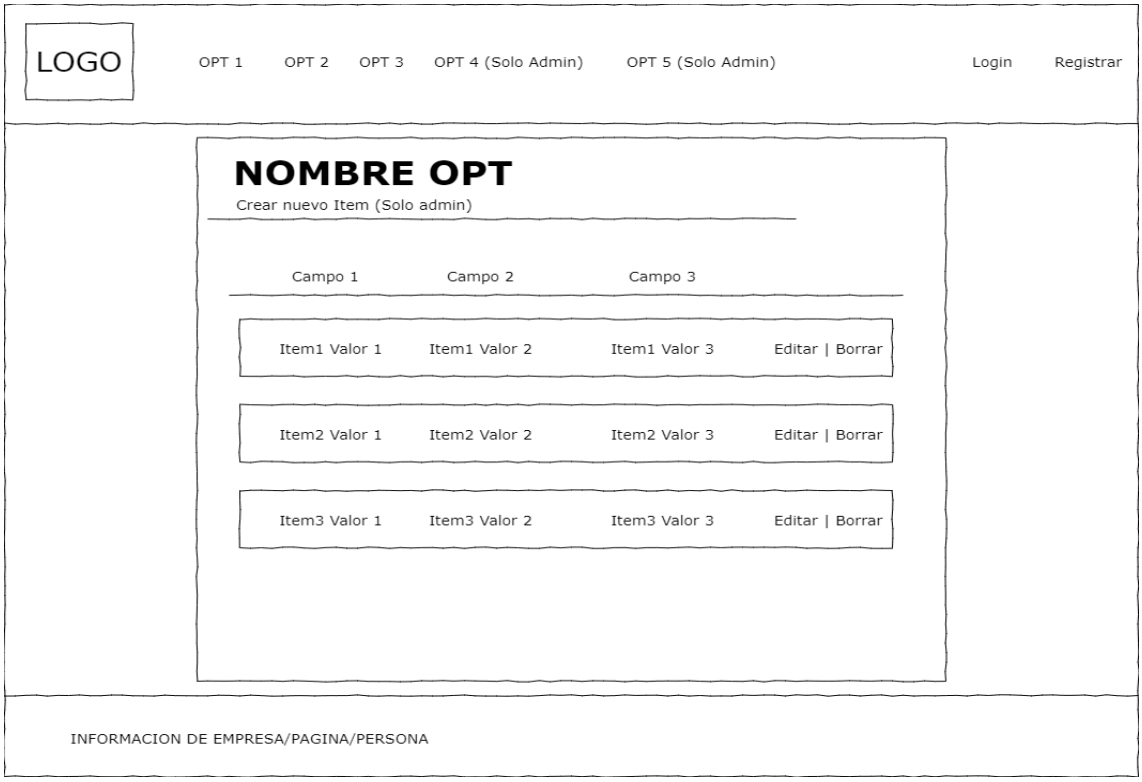

Ilustración 17 Vista de índice de administrador formato escritorio

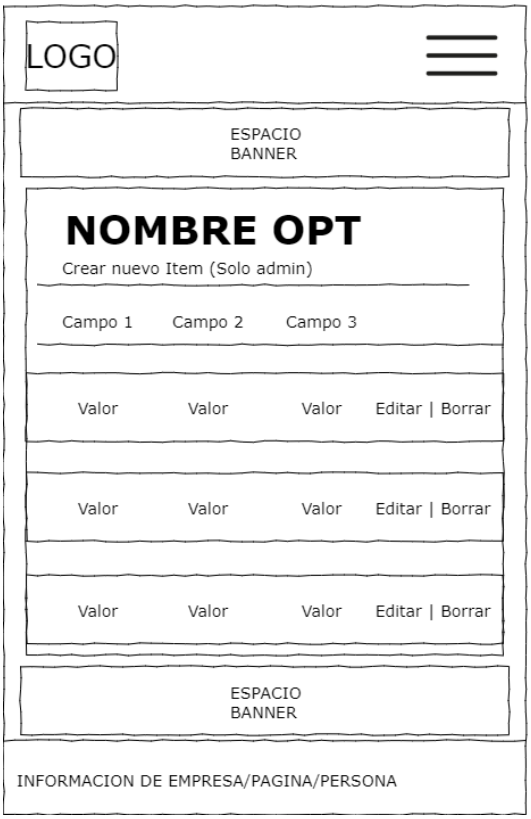

Ilustración 18 Vista de índice de administrador formato móvil

Esta página de índice especial es exclusiva para los Administradores.

Se aplica para aquellos modelos que no tienen imágenes, y por tanto se pueden mostrar los datos en formato tabla para comprimir y ordenar todos los datos administrativos necesarios para que el resto de la web funcione correctamente.

- Details:

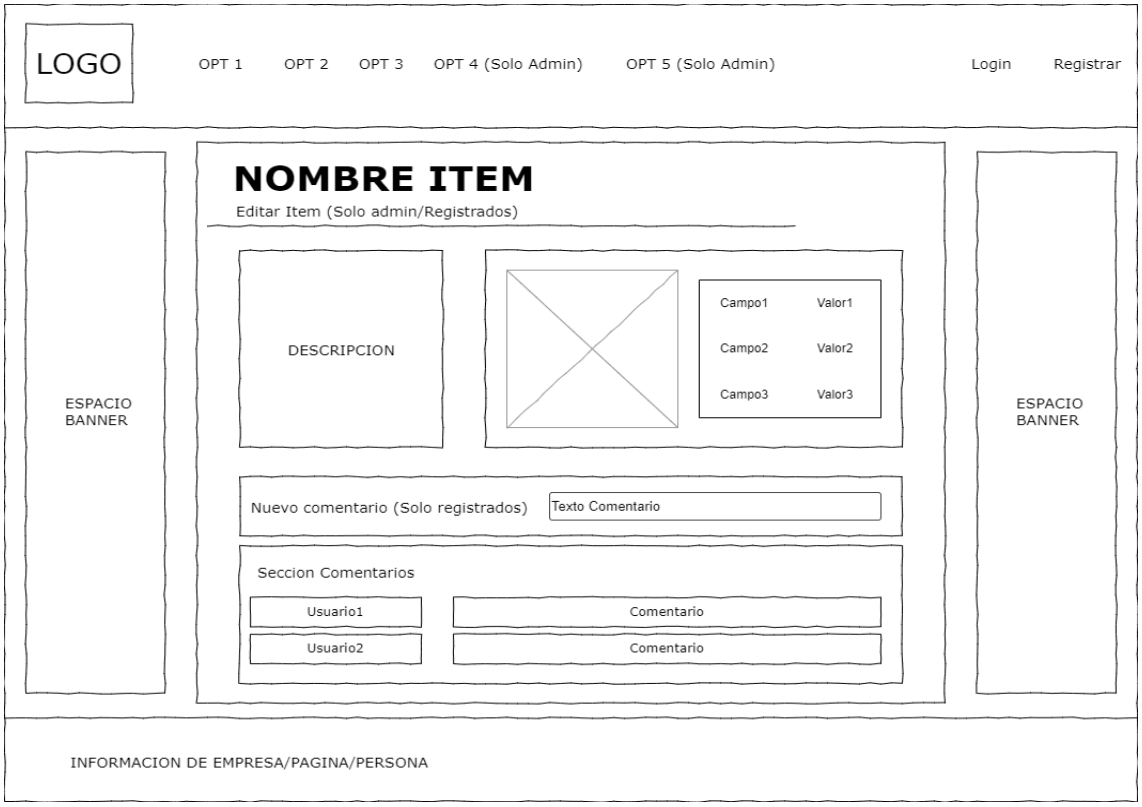

Ilustración 19 Vista de detalles formato escritorio

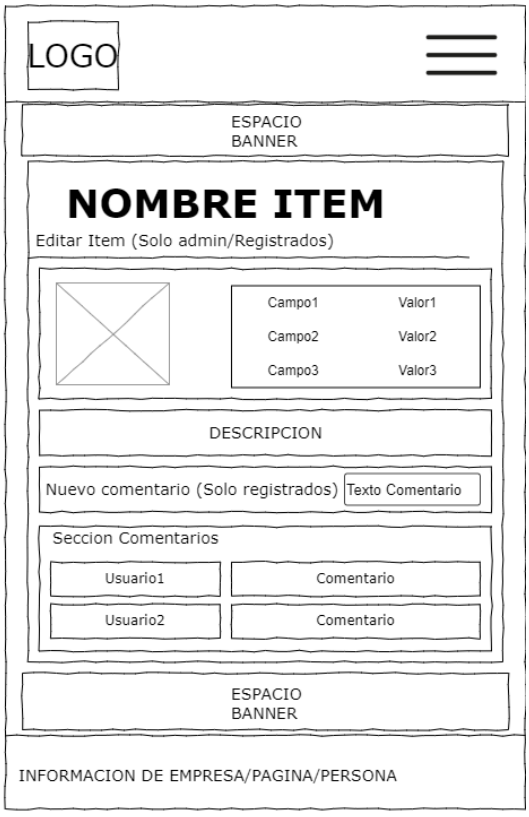

Ilustración 20 Vista de detalles formato movil

La página de detalles es la que más variedad de contenido va a mostrar. Originalmente se iba a dividir la información del objeto en dos tarjetas principales, siendo la subdivisión de la tarjeta correspondiente a los datos técnicos horizontal.

Con los detalles de los objetos de tipo Arma esto no se mostraba visualmente atractivo, por lo que la subdivisión de esta tarjeta de datos se acabó realizando de manera vertical.

- Delete:

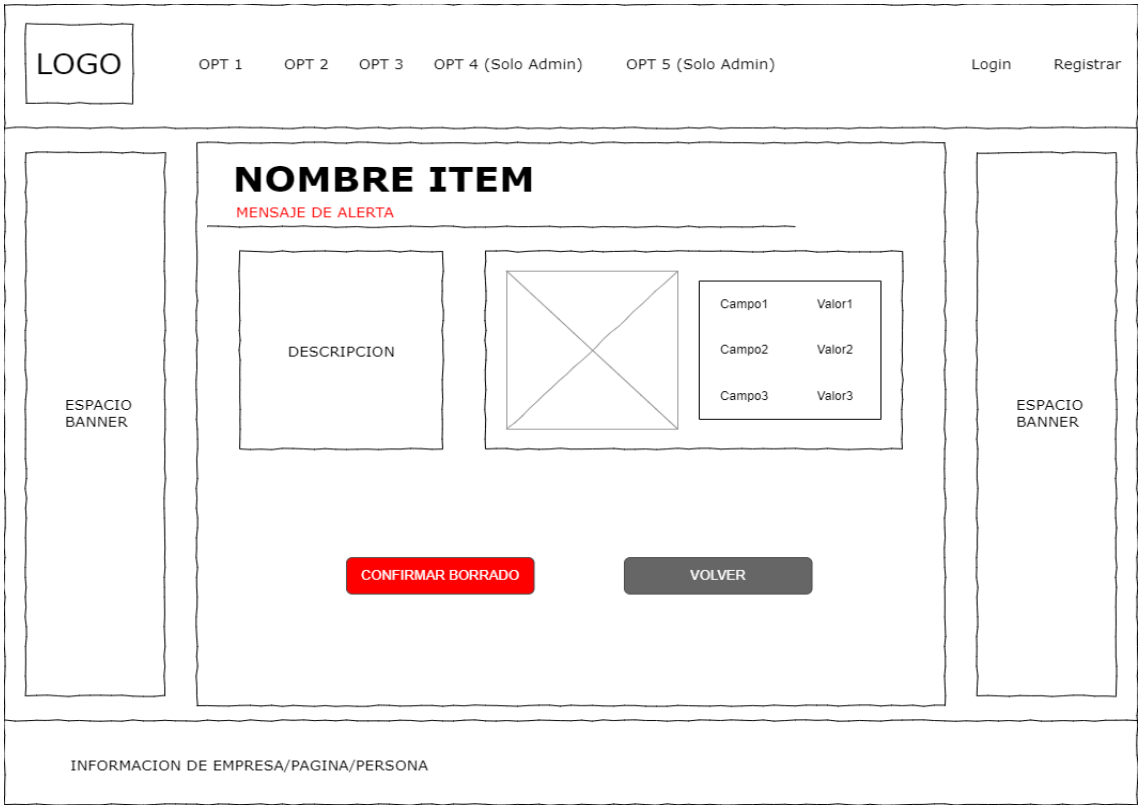

Ilustración 21 Vista de borrar formato escritorio

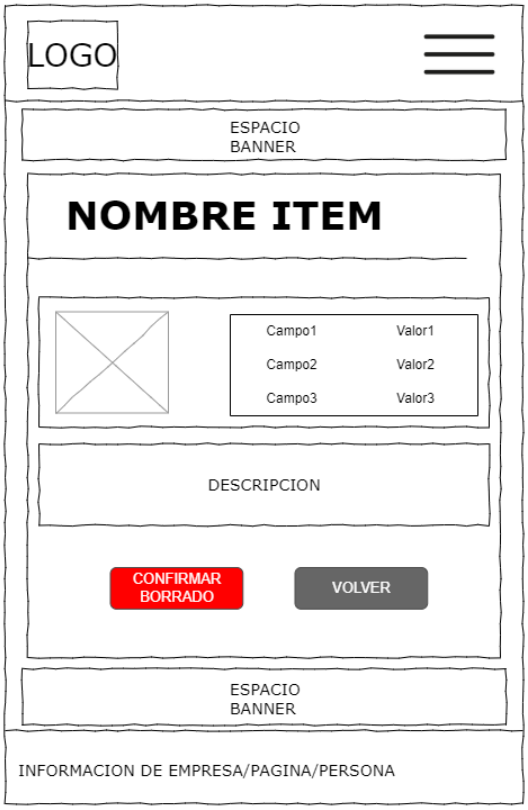

Ilustración 22 Vista de borrar formato móvil

La página de borrado es exclusiva para administradores y muestra información muy similar a la de detalles, la única diferencia es que, en vez de mostrar los comentarios, mostrará las opciones de borra o volver al índice de objetos.

.

- Create/Edit:

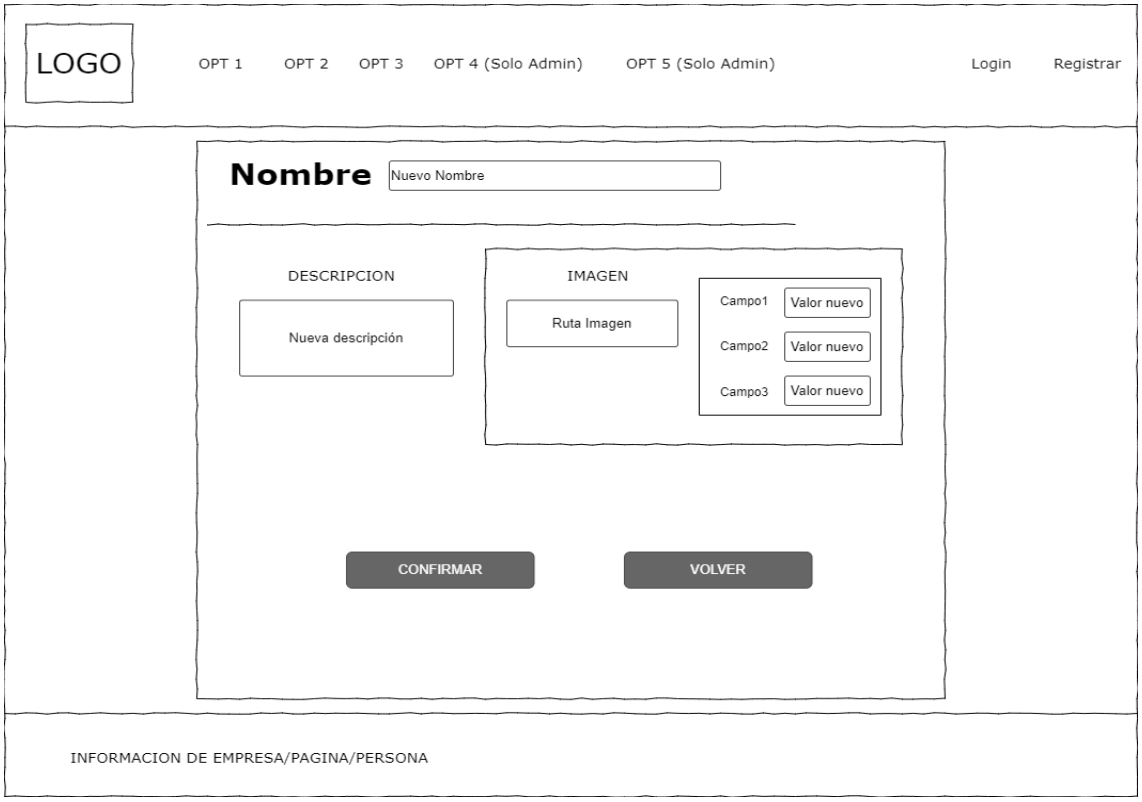

Ilustración 23 Vista de crear/editar formato escritorio

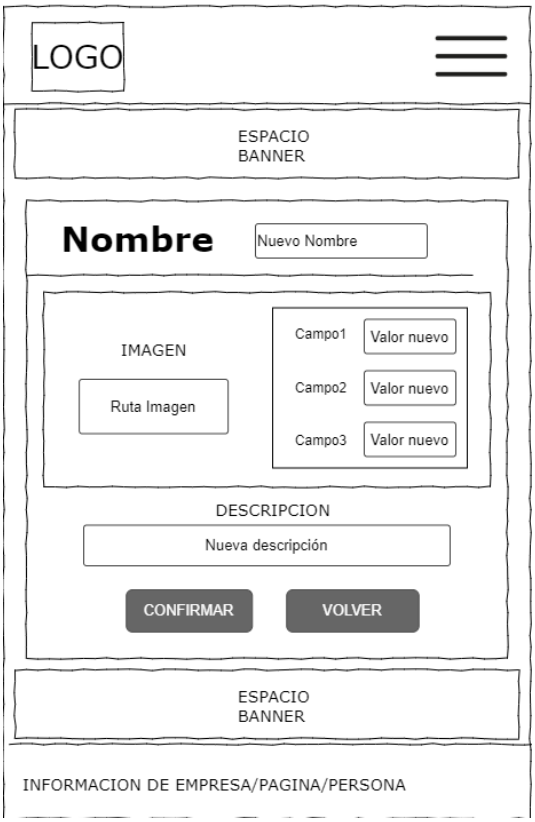

Ilustración 24 Vista de crear/editar formato escritorio

Estas páginas han recibido cambios significativos en la vista final.

Inicialmente se había propuesto mantener una disposición y formato de la información similar a como se muestra en la vista de detalles. Esto hace surgir un problema; la interfaz se vuelve poco intuitiva y el usuario se pierde fácilmente, en muchas ocasiones olvidándose de rellenar campos ya que no los encuentra a simple vista.

Por ello, esta vista se ha modificado para que se vea como un sencillo formulario lineal, en el que el usuario puede leer la información que se necesita proporcionar de forma más sencilla e intuitiva.

- Login/Registrar:

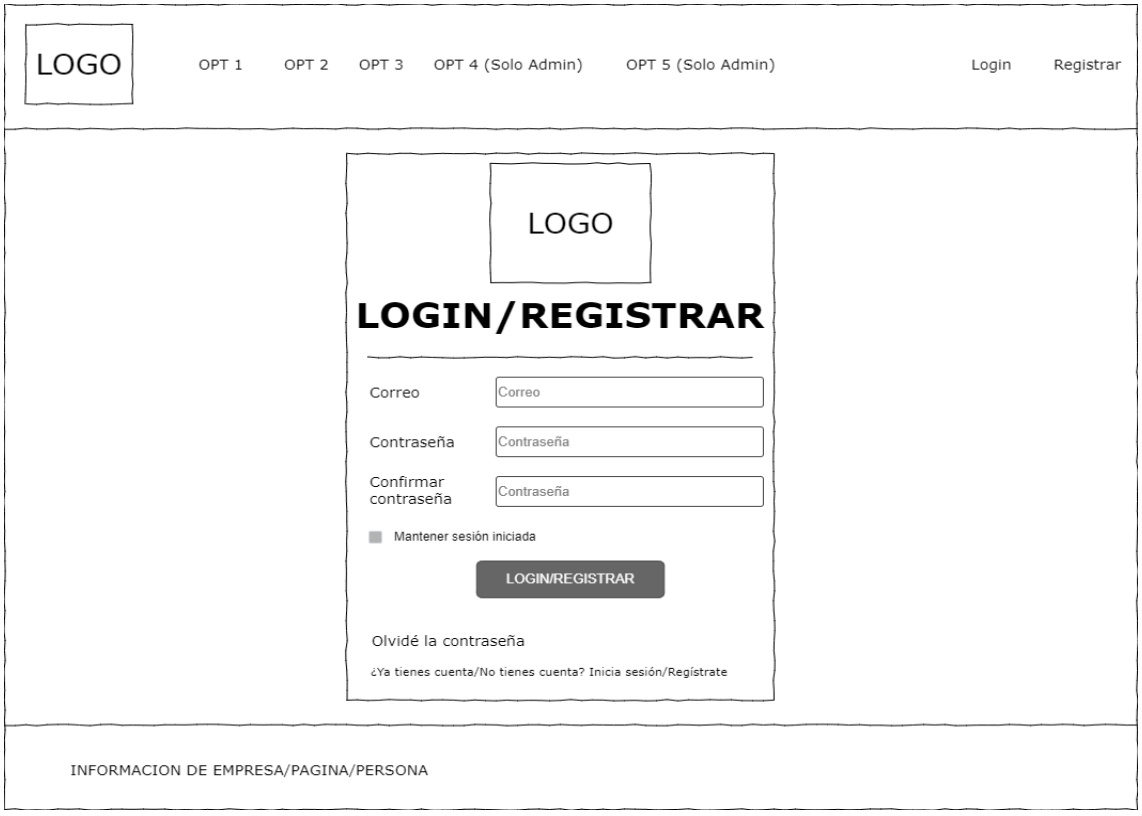

Ilustración 25 Vista de login/registrarse formato escritorio

Ilustración 26 Vista de login/registrarse formato móvil

La página de inicio de sesión y registrarse comparten el mismo formato, al ser poca la información necesaria a aportas, y como ocurre con la mayoría de los formularios en esta página, el contenedor del formulario es estrecho para permitir una lectura vertical de los datos a aportar.

Esto hace que la vista en escritorio y la de móvil sean virtualmente iguales.

Un formato similar se puede observar a la hora de acceder a la administración de la cuenta del usuario con sesión iniciada en ese momento.

Todas las vistas han sido diseñadas con las ampliaciones en mente, por lo que en los mockups se visualizan los elementos correspondientes a éstas. También observaremos que, a la hora de navegar en la web, en las vistas definitivas hay zonas libres de contenido, dejadas deliberadamente en caso de poder hacer alguna de las ampliaciones descritas en este documento.

Como se puede observar en las comparativas entre mockups y vistas definitivas, se han realizado algunos cambios para mejorar la experiencia de navegación, accesibilidad y responsividad del producto final.

Diseño:

Se ha decidido la disposición del contenido, el esquema de colores y tipo de letra basándonos en una captura del menú de selección de equipo. A continuación se muestra una comparación entre el menú de interfaz del juego y la página web:

Ilustración 27 Ejemplo de menú en el videojuego

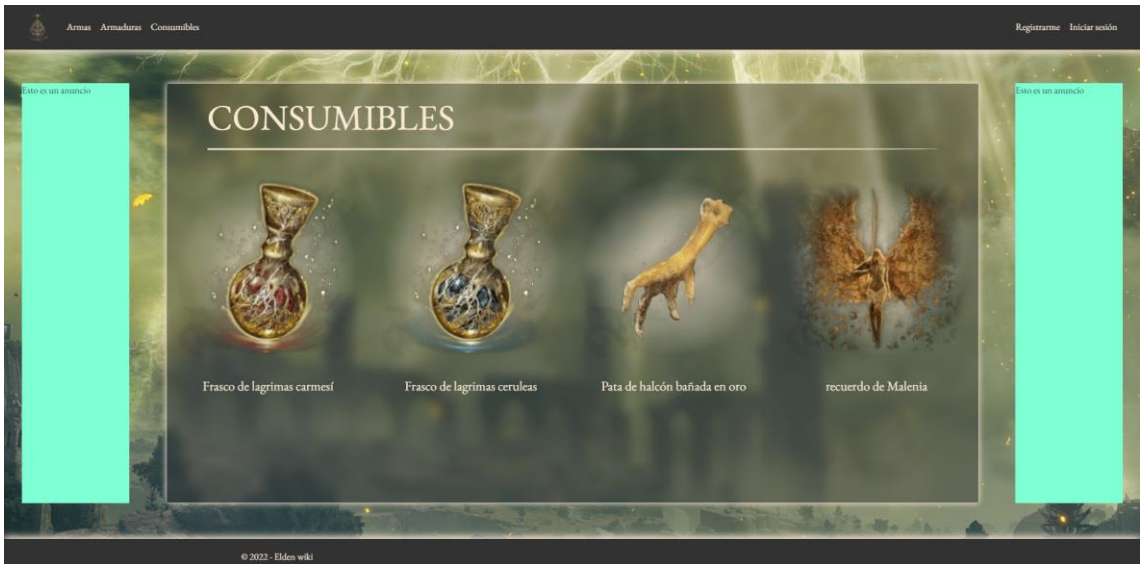

Ilustración 28 Ejemplo de vista en la web para comparación con el menú del juego

Como se puede observar, la caja de contenido principal es de un gris translúcido, muy similar a como se muestra el fondo del menú del juego, dejando ver el fondo, la diferencia reside en que la caja de contenido de la web tiene un efecto difuminado para el fondo, esto es para poder leer el contenido de manera más sencilla, sin que interfiera la imagen de fondo.

De la misma manera, la caja, el menú y el pie de página de la web tienen un ligero efecto de brillo en color beige claro, lo mismo ocurre en la interfaz del juego, donde se puede apreciar un efecto de brillo degradado similar en la zona inferior, aunque este ocupa una zona mucho más extensa.

Las barras de separación tienen un efecto de degradado u ofrecen un efecto de brillo o incandescencia, lo mismo ocurre con las barras de separación de los títulos del contenido de la página y en las divisiones verticales entre las opciones de la página de inicio de la web.

En cuanto a la letra, mediante una larga investigación y búsqueda se ha conseguido encontrar la fuente usada en el videojuego en Google Fonts (EB Garamond), dicha fuente es la usada en todo el texto del proyecto. Respecto a su color, a pesar de que el videojuego usa una combinación de beige y blanco, para la web se ha decidido mantener el beige y, en caso de ser necesario añadir contraste con el fondo, el color de letra pasa a ser de color grisáceo. A pesar de estas elecciones, el color de los mensajes de alerta u opciones que suponen cambios irreversibles se han mantenido en rojo.

A la hora de añadir cierto contraste para las imágenes de los objetos respecto al fondo, ya que muchos de estos elementos son de tonos apagados y grisáceos por la naturaleza

y ambientación del videojuego, se les ha añadido en el índice un degradado radial de color beige claro, simulando una vela alumbrándolos desde atrás, manteniendo así la temática medieval. Este efecto de retroiluminación también se utiliza al pasar el ratón por encima de las opciones del menú.

Respecto a la imagen de fondo, en una primera instancia dicha imagen cambiaba cada X segundos para hacer efecto de slider o presentación y hacer el estilo de la página más dinámico. Esto supuso un problema a la hora de controlar el contraste de los elementos respecto del fondo, ya que este cambiaba constantemente, por lo que finalmente se optó por dejar una imagen estática para mejorar la accesibilidad del sitio.

La tarjeta que contiene toda la información técnica respecto al objeto también está representada de forma similar a la que aparece en el juego, posicionando primero la imagen del objeto y justo debajo sus características técnicas.

POSIBLES AMPLIACIONES:

Sección de comentarios:

Se generaría una sección de comentarios en la vista de detalles de cada objeto, donde los comentarios presentados serán los referidos únicamente a dicho objeto.

Constará de una casilla de introducción de texto, de un máximo de 300 caracteres para evitar la carga excesiva de información en la base de datos.

Seguido de dicha casilla se encontrará un contenedor con todos los comentarios ordenados de manera cronológica, siendo los más recientes los primeros en verse. Dicho contenedor podrá soportar hasta 20 comentarios antes de tener que pulsar un botón para cargar más.

Barra de búsqueda:

Se generará en el índice de los objetos que lo necesiten por volumen de contenido una barra de búsqueda que permitirá buscar los objetos por su nombre.

En caso de no encontrar ninguna coincidencia con el texto introducido en la barra de búsqueda, se volverán a mostrar toda la lista de objetos completa.

Ampliación de la base de datos:

Se ampliará la base de datos para añadir, además de objetos, localizaciones, NPCs y misiones, que estarán interconectados entre sí, ya que para la obtención de algunos objetos es necesario la finalización de ciertas misiones, y dichas misiones requieren encontrar a ciertos NPCs en ciertas localizaciones.

Gracias a ello, el apartado de descripción podría engrosarse y subdividirse, ya que ampliando e interconectando la información sería aconsejable separar la información en diversos apartados. De manera general estas subdivisiones serían:

Objetos:

- Descripción.
- Dónde encontrarlo.
- Requisitos de obtención.
- NPCs ligados a la obtención de dicho objeto.
- “Playthrough” o recorrido a seguir paso a paso para obtener el objeto.

Misiones:

- Descripción.
- NPCs ligados al desarrollo y finalización de la misión.
- Recompensas.
- “Playthrough” o recorrido a seguir paso a paso para la finalización de la misión.

NPCs:

- Descripción y trasfondo.
- Estadísticas (Nivel, vida, ataque, defensa, etc.)
- Localización del NPC según el momento cronológico de la historia.
- Condiciones para su desaparición.

Localizaciones:

- Área.
- Región.
- Peligros encontrados en la localización.

MANUAL DE USO:

Corrección del sistema de puntuaciones de Windows:

Los proyectos de VisualStudio2019 recogen las configuraciones del sistema de Windows para el control de la entrada de datos a la Base de Datos. Esto significa que puede haber datos mal interpretados a la hora de introducir datos del tipo fecha o decimal.

En nuestro caso no presentaremos fechas al usuario, por lo que no es necesario cambiar ese tipo de dato, lo que sí se interpretará mal para el usuario español es la forma de tomar decimales. Ya que los signos de puntuación para los decimales son diferentes:

- Con nomenclatura americana, la coma representa los miles y los puntos representan los decimales.
- Con nomenclatura española las puntuaciones son a la inversa, los puntos representan miles, y las comas representan los decimales.

Si tratamos de introducir o editar un objeto con parámetros decimales usando la nomenclatura española recibiremos un aviso de error al usar las comas como decimales, ya que el sistema no lo representará como un número decimal. Y al usar los puntos nos permitirá la introducción del dato, pero al mostrarlo se verán como miles, es decir:

Si introducimos 47.50 como valor mientras estemos usando la nomenclatura española el valor se mostrará posteriormente al usuario final como 4750,00.

Por tanto, para una correcta administración de los objetos de la base de datos se necesita la nomenclatura americana. Por lo que, en el panel de control del ordenador, clicaremos en Reloj y región, posteriormente en Región:

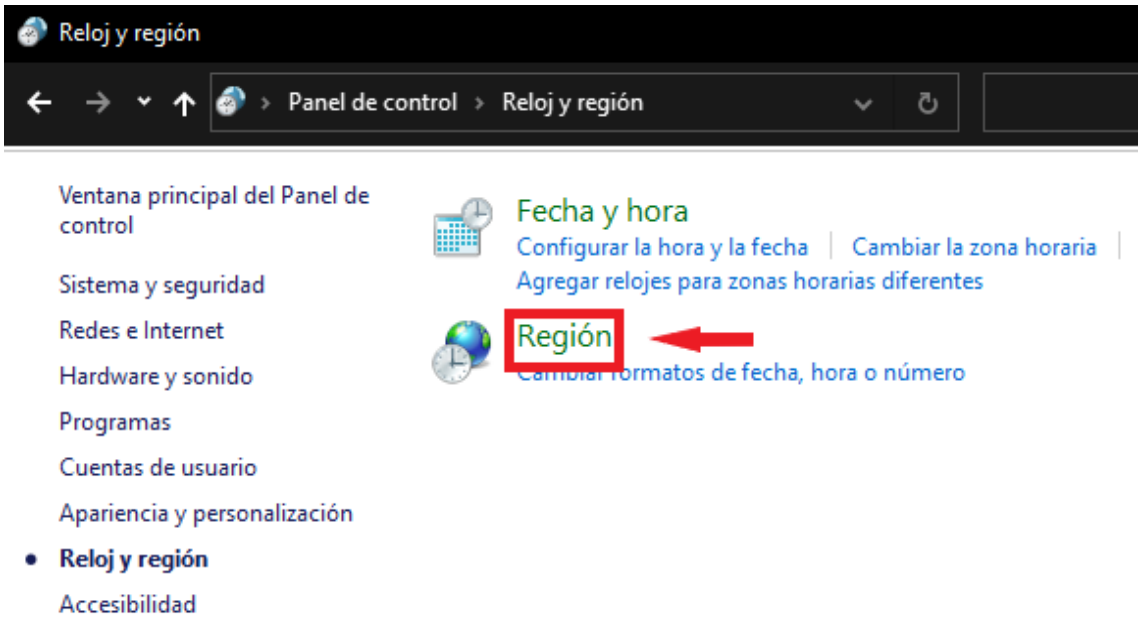

Ilustración 29 Panel de control, reloj y región

Aparecerá una opción de configuración adicional en la parte inferior izquierda de la nueva ventana emergente, clicamos en ella. En esta nueva ventana cambiaremos el valor de Símbolo decimal a un punto "." Y el valor de Símbolo de separación de miles a una coma ",":

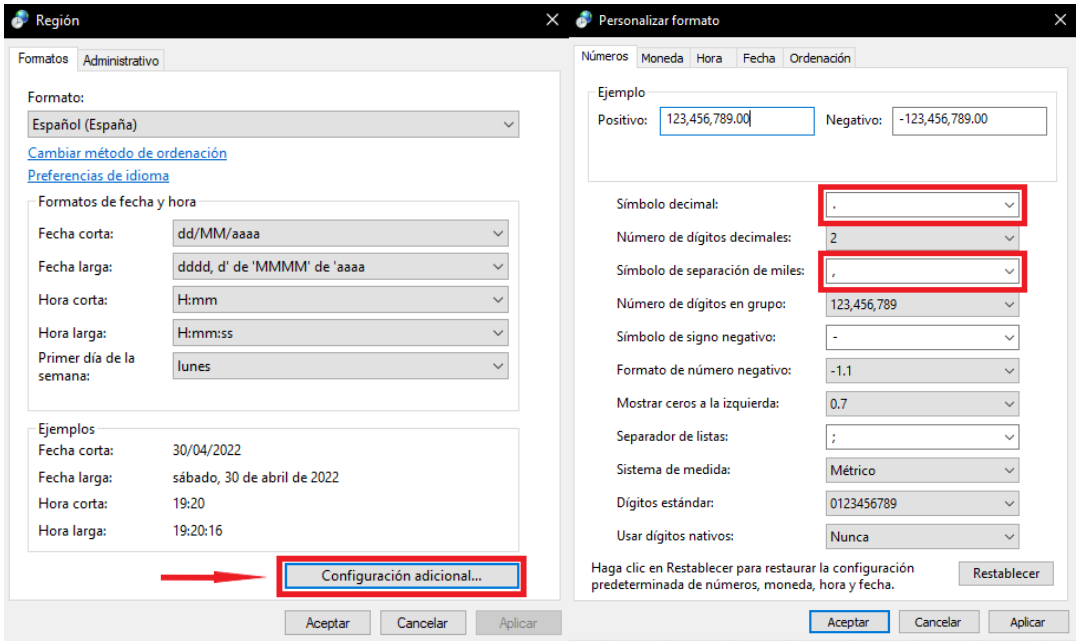

Ilustración 30 Cambio de los Símbolos realizado

Ilustración 31 Pantalla previa a la configuración adicional

Tras esto solo falta clicar en aplicar y aceptar en las ventanas para cerrarlas.

Popular la base de datos con información de ejemplo:

Para poder observar la página lo más cerca de cómo se vería en la realidad, ésta debe de tener contenido para visitar. En este apartado veremos cómo podemos aprovechar un guardado de la base de datos ya creada previamente por el desarrollador, para evitar el tedioso trabajo de crear cientos de elementos a mano.

Este archivo de guardado es un archivo ".dacpac" que se debería encontrar adjunto en el archivo comprimido del proyecto bajo el nombre "CopiaBBDDProyecto.dacpac".

Para comenzar abriremos el proyecto con VisualStudio2019, esto se hace clicando en el archivo con extensión ".sln" situado dentro de la carpeta con el nombre del proyecto. Una vez abierto el proyecto, en la consola de administración de paquetes, situada en la parte inferior de la ventana, escribiremos "add-migration" seguido del nombre que deseemos, ya que este es indiferente ahora mismo. En cuanto esto haya acabado y nos salga el mensaje "Build Succeeded" escribiremos "update-database" y esperaremos a el mismo mensaje de confirmación positivo en la consola.

Una vez hecho esto podemos desplegar la vista de la base de datos, que por defecto debería estar localizada en un menú con el título "Explorador de objetos de SQL Server", situado al margen izquierdo de la ventana. Ten en cuenta que para ver la base de datos que acabamos de crear debemos de refrescar este menú:

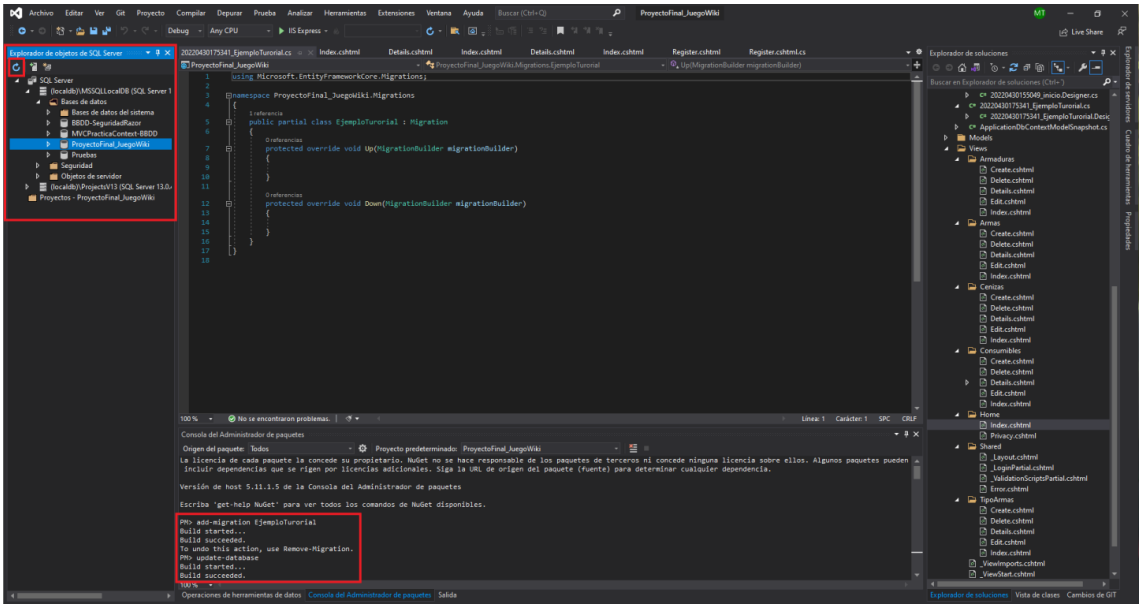

Ilustración 32 Vista general de VisualStudio2019

Buscamos la base de datos con el nombre “ProyectoFinal_JuegoWiki” y hacemos clic derecho sobre ella. Seleccionamos la opción “publicar aplicación de capa de datos”, nos saldrá una ventana emergente donde podremos seleccionar el archivo “.dacpac” mencionado previamente. Tan sólo queda clicar en el botón publicar y clicar en la opción de “Sí” en la ventana de aviso que nos saldrá justo después:

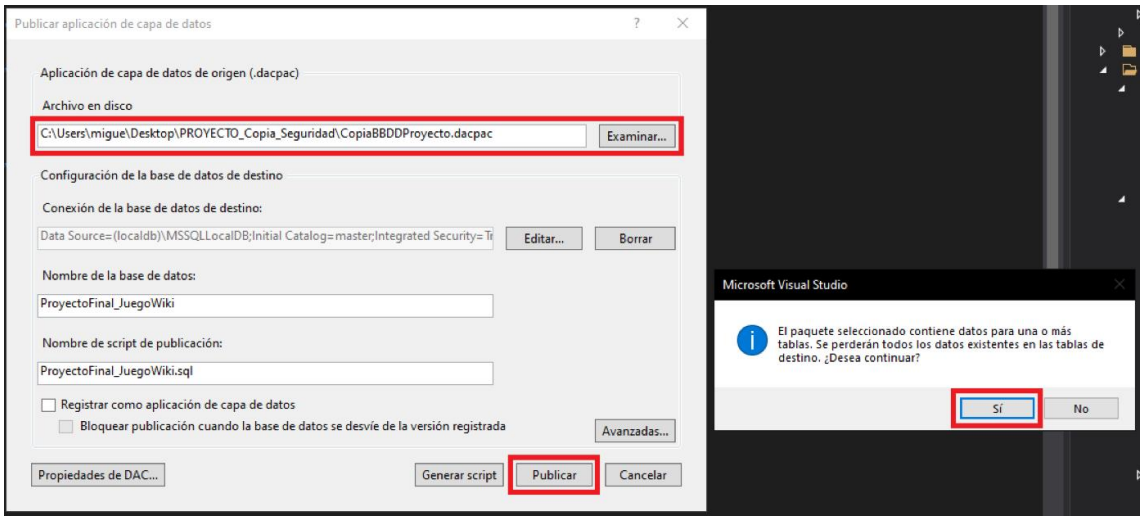

Ilustración 33 Pantalla de selección y confirmación para publicar una aplicación de capa de datos

Esperamos a que se realice la copia de los datos, sabremos que ha acabado cuando veamos un mensaje con un tick verde en la sección inferior de la ventana:

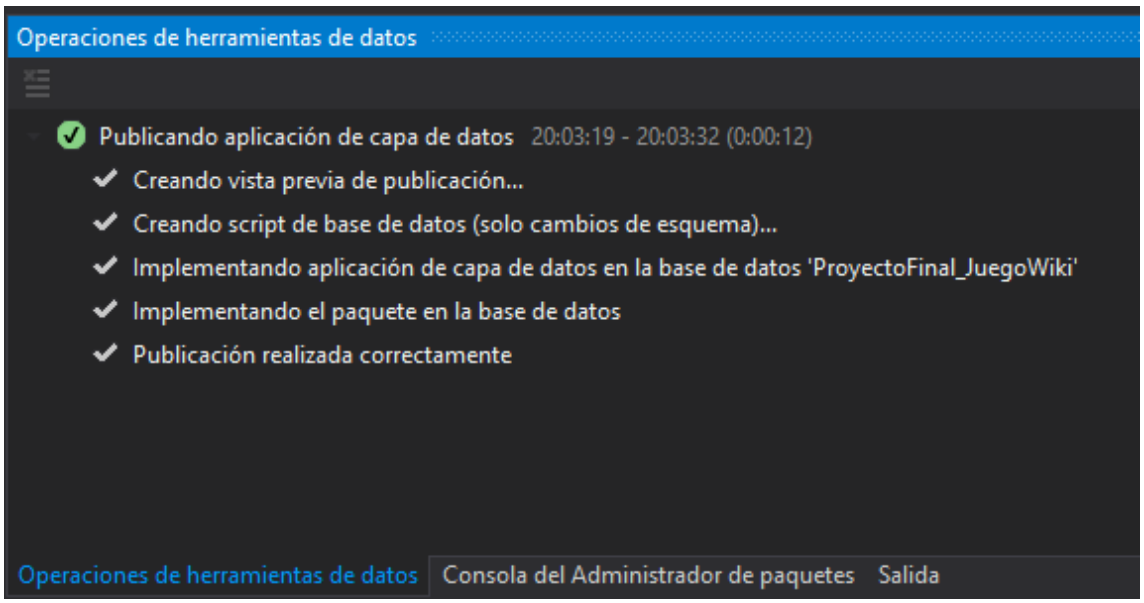

Ilustración 34 Publicación de los datos exitosa

Tras esto ya podemos clicar en el botón de IIS Express con el icono de reproducir y navegar por la página que ya debería tener una gran cantidad de contenido.

Para poder acceder a la página como administrados y poder visualizar todas las opciones que ofrece, introducimos las siguientes credenciales al iniciar sesión:

- Usuario: **admin1@administrador.admn**
- Contraseña: **C@ntrasenia_1**