

Título: Inventario Informático.

Proyecto de GRADO SUPERIOR DE DESARROLLO DE APLICACIONES WEB

Centro: IES Rodeira

Alumno: Gabriel Ibán Cruz Fragueta

Dirigido por: Pablo Vázquez Fernández

Curso: 2020 – 2021

Índice

Introducción	3
Presupuesto	4
Manual de usuario	5
Manual de Instalación	10
Instalación para Desarrollo	10
Instalación para Ejecución.....	11
Documentación técnica.....	12
Herramientas utilizadas para la realización del proyecto	12
Microsoft Blazor	12
Desventajas encontradas	14
Análisis de la aplicación.....	14
Implementación	18
Posibles mejoras a la aplicación	19
Funcionales.....	19
Técnicas.....	19

Introducción

La realización de este proyecto, además de para demostrar conocimientos teóricos y prácticos que me lleven a la obtención del título de Grado Superior en Desarrollo de Aplicaciones Web, me permite aplicar una nueva tecnología y filosofía de desarrollo que Microsoft ha presentado. El nombre de dicha tecnología es “**Blazor**”.

Para mí, el estado actual del arte del desarrollo web, permite hacer aplicaciones cada vez más parecidas a la ejecución de una aplicación local o smart client pero con un coste de desarrollo importante. En muchas ocasiones son personas distintas, con capacidades distintas, y herramientas distintas las que se encargan de la parte “Backend” y las que se encargan de la parte “Frontend”. En entornos empresariales donde un único programador se encarga de realizar soluciones web sin grandes requerimientos, pero que donde el coste de desarrollo es crítico, el tener una api web capaz de resolver gran cantidad de peticiones no es una prioridad. La prioridad es tener en un tiempo razonable y de una forma mantenible soluciones a las necesidades del día a día de los distintos departamentos de la empresa.

Tras realizar este proyecto, se ha confirmado para mí que Blazor ayuda en las necesidades que expongo en el párrafo anterior. En la parte técnica de la presente documentación se habla de la tecnología.

La excusa para realizar un desarrollo similar al de una aplicación real consiste en la implementación de una aplicación de **Inventario Informático**. Dicho tipo de aplicaciones, típicas de un entorno empresarial, se ajusta al tipo de desarrollo Blazor seleccionado: el **Blazor Server** donde no se espera una ingente cantidad de usuarios concurrentes y el servidor asume gran cantidad de esfuerzo en la ejecución, permitiendo que los clientes puedan tener unas características mínimas y ser perfectamente funcionales con la aplicación.

La aplicación en sí almacena, en una Base de Datos, la información acerca de Monitores, Estaciones de Trabajo y Portátiles. Además, permite la gestión de todas las tablas que se referencian como: Personas, Ubicaciones, Marcas y Modelos, y otras.

Presupuesto

Este proyecto está pensado como proyecto de investigación. En un entorno real pararía el trabajo facturable de recursos de la empresa. Esto se haría con la intención de obtener un retorno futuro por la inversión de capital humano. El cálculo de horas que se ha realizado teniendo en cuenta que se realizaría por una única persona la investigación y la realización de una aplicación interna con la que demostrar el uso práctico de la tecnología.

La dividimos en:

- 10 horas de investigación pura.
- 50 horas de codificación / investigación.

Partimos de que el coste/hora de un programador senior con gastos generales incluidos es de 37,5 €/hora.

Queda reflejado en la siguiente tabla:

FASE	HORAS	PRECIO/HORA	PRECIO FASE
INVESTIGACIÓN PURA	10	37,50€	375€
CODIFICACIÓN / INVESTIGACIÓN	50	37,50€	1.875€
TOTALES	60		2.250€

A ello habría que añadir el coste de oportunidad de no disponer de un recurso alrededor de dos semanas.

Manual de usuario

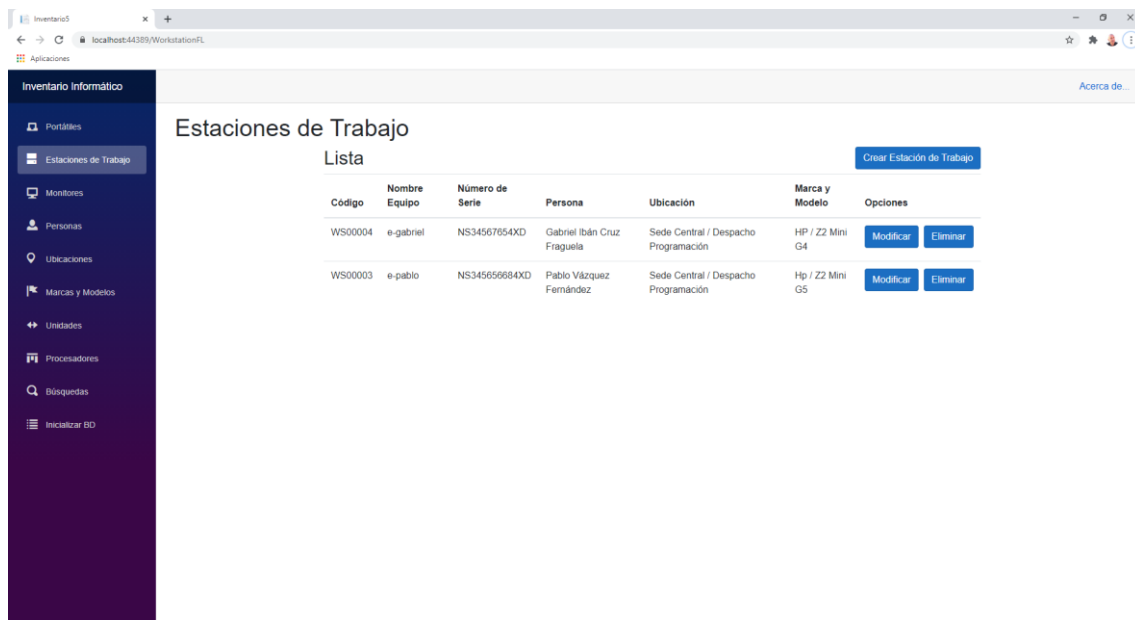
El uso de la aplicación es muy sencillo.



Como se puede ver tenemos un menú lateral que nos permite acceder a las distintas opciones del programa.

En las tres primeras tenemos un listado del elemento correspondiente permitiéndonos añadir nuevos elementos, modificar algún elemento o eliminarlo. Antes de eliminar un elemento nos muestra la pantalla con los datos para asegurarnos de que lo queremos eliminar.

Pantalla de listado.



Pantalla de edición de datos

Inventario Informático

Portátiles

Estaciones de Trabajo

Monitores

Personas

Ubicaciones

Marcas y Modelos

Unidades

Procesadores

Búsquedas

Inicializar BD

Portátil

Modificar

Código

POR000001

Nombre del equipo

p-comercial

Número de Serie

CSH456345CE

Memoria

8

Unidad

GigaByte

Disco Duro

250

Unidad

GigaByte

Procesador

AMD Ryzen Threadripper 3970X

Velocidad Procesador

2,8

Tamaño de la pantalla (pulgadas)

14

Persona asignada

Patricia Pazos Alves

Buscar

Ubicación

Sede Central / Despacho Gestión

Buscar

Inventario Informático

Portátiles

Estaciones de Trabajo

Monitores

Personas

Ubicaciones

Marcas y Modelos

Unidades

Procesadores

Búsquedas

Inicializar BD

Número de Serie

CSH456345CE

Memoria

8

Unidad

GigaByte

Disco Duro

250

Unidad

GigaByte

Procesador

AMD Ryzen Threadripper 3970X

Velocidad Procesador

2,8

Tamaño de la pantalla (pulgadas)

14

Persona asignada

Patricia Pazos Alves

Buscar

Ubicación

Sede Central / Despacho Gestión

Buscar

Marca y Modelo

Asus / Zenbook UX462DA

Buscar

Notas

Portátil del comercial

Modificar

Cancelar

Los elementos que nos podemos encontrar en la edición de datos son:

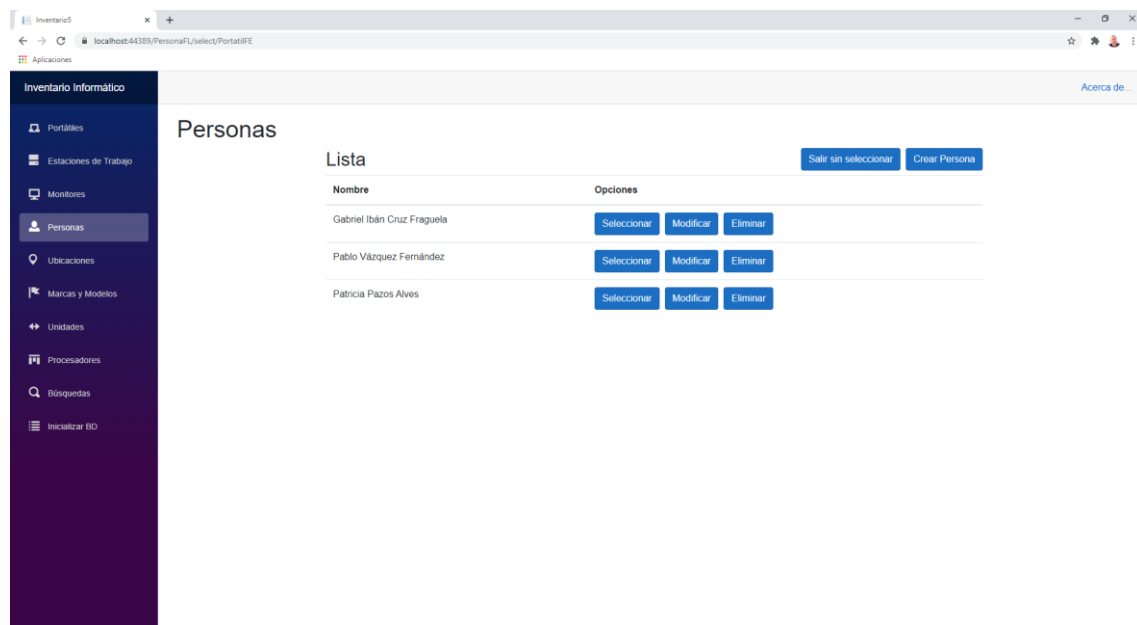
Introducción de texto: Simplemente habría que introducir el texto que nos solicitan. Puede ser de una única línea o de varias como en los apartados de “notas”.

Comboxes: En los que habría que, pulsando la flecha, seleccionar una de las opciones.

Campos de introducción numérica: En donde solamente podremos introducir números.

Y por último tenemos campos no editables donde se nos da la opción de Buscar. Por ejemplo, personas. Si pulsamos el botón correspondiente nos mostrará el equivalente a otra página donde podremos seleccionar una persona. También podremos desde la misma pantalla añadir, modificar o eliminar personas dando la opción de crear el nuevo elemento si todavía no existe.

Pantalla con la opción de seleccionar:



También tendremos la opción de salir sin seleccionar.

Las siguientes cinco opciones: “Personas”, “Ubicaciones”, “Marcas y Modelos”, “Unidades” y “Procesadores” son similares a las anteriores pero mucho más sencillas dando únicamente la posibilidad de cambiar el nombre.

Por último llegamos a la opción de búsqueda:

Búsquedas

Elementos a incluir

Incluir Portátiles ☒ Incluir Estaciones de Trabajo ☒ Incluir Monitores ☒

Elemento Inventariable (válido para todo tipo de elementos)

Código

Ordenador (válido para Portátiles y Estaciones de Trabajo)

Nombre

Número de serie

Persona asignada

Ubicación

Marca y Modelo

Buscar

En esta pantalla tendremos la capacidad de seleccionar qué elementos queremos buscar entre Portátiles, Estaciones de Trabajo y/o Monitores.

El resto de opciones especifican valores y se ha optado por que si alguno de los mismos coincide se muestre, es decir, si se introduce un código y un nombre de equipo los elementos se mostrarán si coincide con uno u otro de los campos especificados.

Si se especifica un texto en código, nombre o Número de serie se buscará que el elemento “contenga” lo buscado. Por ejemplo el nombre de un equipo es “e-Manuel”. Si buscamos “anue” nos lo mostrará en la pantalla siguiente.

Si se especifica una Persona Asignada, Ubicación y/o Marca y Modelo, aparecerán los elementos que tengan eso que se especifica además de las otras opciones que se marquen.

Al pulsar en buscar pasamos a la pantalla de resultados:

Resultados de búsqueda

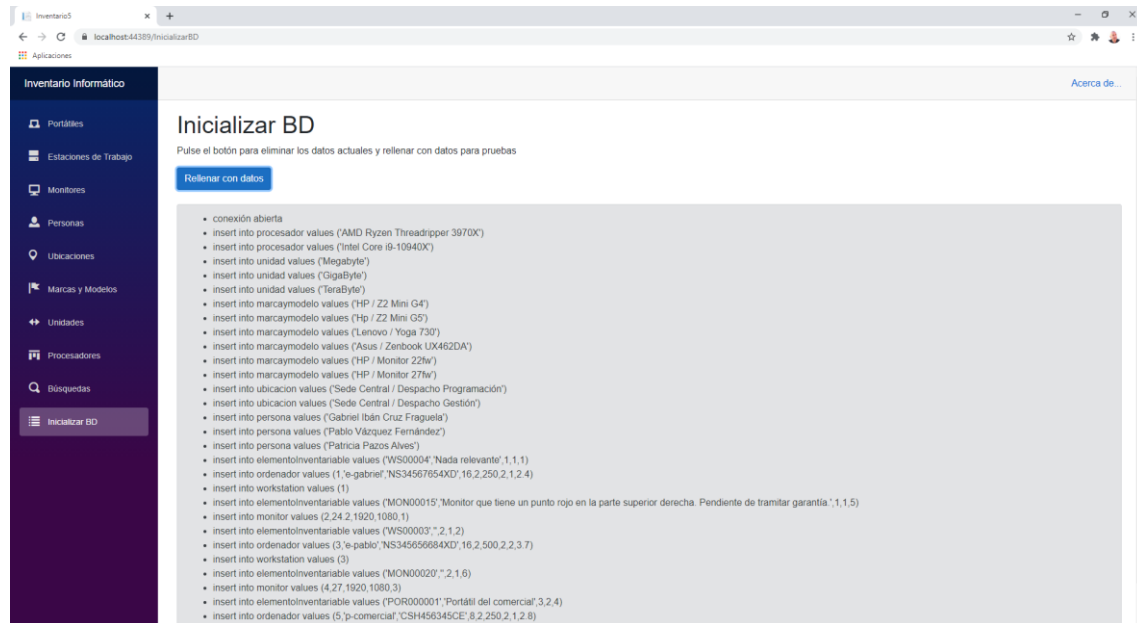
Lista

Buscar de nuevo

Tipo de Elemento	Código	Nombre	Número de serie	Opciones
Portátil	POR000001	p-comercial	CSH456345CE	Modificar Eliminar
Est. Trab.	WS00004	e-gabriel	NS34567854XD	Modificar Eliminar
Est. Trab.	WS00003	e-pablo	NS345656684XD	Modificar Eliminar
Monitor	MON00015			Modificar Eliminar
Monitor	MON00020			Modificar Eliminar

Desde aquí podemos modificar o Eliminar cualquiera de los elementos encontrados o buscar de nuevo. Las opciones que se han especificado en la búsqueda se mantienen, no se pierden con cada búsqueda.

Se ha mantenido la opción de “Inicializar BD” para poder, desde la propia aplicación borrar todos los datos y rellenar con datos de prueba la aplicación. Al entrar en la pantalla y pulsar el botón correspondiente muestra las sentencias sql ejecutadas.



The screenshot shows a web application interface for 'Inventario Informático'. The left sidebar contains a menu with options: Portátiles, Estaciones de Trabajo, Monitores, Personas, Ubicaciones, Marcas y Modelos, Unidades, Procesadores, Búsquedas, and Inicializar BD (highlighted). The main content area is titled 'Inicializar BD' and includes a sub-header 'Pulse el botón para eliminar los datos actuales y rellenar con datos para pruebas'. Below this is a blue button labeled 'Rellenar con datos'. The main body of the page displays a list of SQL statements to be executed, including commands to insert data into tables like 'conexion', 'procesador', 'unidad', 'marca', 'modelo', 'ubicacion', 'persona', 'elemento', and 'ordenador'.

- conexión abierta
- insert into procesador values (AMD Ryzen Threadripper 3970X)
- insert into procesador values (Intel Core i9-10940X)
- insert into unidad values (Megabyte)
- insert into unidad values (GigaByte)
- insert into unidad values (TeraByte)
- insert into marca values (HP / Z2 Mini G4)
- insert into marca values (HP / Z2 Mini G5)
- insert into marca values (Lenovo / Yoga 730)
- insert into marca values (Asus / Zenbook UX462DA)
- insert into marca values (HP / Monitor 22fw)
- insert into marca values (HP / Monitor 27fw)
- insert into ubicacion values (Sede Central / Despacho Programación)
- insert into ubicacion values (Sede Central / Despacho Gestión)
- insert into persona values (Gabriel Ibán Cruz Fraguera)
- insert into persona values (Pablo Vázquez Fernández)
- insert into persona values (Patricia Pazos Alves)
- insert into elemento values ('WS00004', 'Nada relevante', 1, 1, 1)
- insert into ordenador values ('e-gabriel', 'NS34567854XD', 16, 2, 250, 2, 1, 2, 4)
- insert into workstation values (1)
- insert into elemento values ('MON00015', 'Monitor que tiene un punto rojo en la parte superior derecha. Pendiente de tramitar garantía.', 1, 1, 5)
- insert into monitor values (2, 24, 2, 1920, 1080, 1)
- insert into elemento values ('WS00003', 2, 1, 2)
- insert into ordenador values (3, 'e-pablo', 'NS345656684XD', 16, 2, 500, 2, 2, 3, 7)
- insert into workstation values (3)
- insert into elemento values ('MON00020', 2, 1, 6)
- insert into monitor values (4, 27, 1920, 1080, 3)
- insert into elemento values ('POR000001', 'Portátil del comercial', 3, 2, 4)
- insert into ordenador values (5, 'p-comercial', 'CSH456345CE', 8, 2, 250, 2, 1, 2, 8)

Manual de Instalación

Instalación para Desarrollo

En un equipo con Windows, unos pasos que nos permitirían continuar con el desarrollo de la aplicación son:

Asumimos que al equipo se le ha instalado la posibilidad de instalar programas a través del gestor de paquetes chocolatey (<https://chocolatey.org/>)

1 – Instalar Visual Studio Community

```
choco install visualstudio2019community
```

2 – Instalar opciones de Visual Studio

Una vez instalado acudir al menú de “herramientas” y “obtener herramientas y características”

Aquí seleccionaremos al menos las cargas de trabajo “Desarrollo ASP.net y Web” así como “Desarrollo multiplataforma .net core”

3 – Obtener el proyecto

Arrancamos el Visual Studio con las herramientas instaladas y, en la ventana de nuevo proyecto, seleccionaremos clonar proyecto existente y utilizaremos <https://github.com/Gabriel-Iban/Inventario5.git>

4 – Instalar SqlServer

En principio valdría cualquier versión de SqlServer. Para este desarrollo se ha utilizado SqlServer Express

```
choco install sql-server-express
```

5 – Instalar Microsoft SqlManagement Studio

Esta aplicación se utiliza para facilitar el manejo de SqlServer

```
choco install sql-server-management-studio
```

6 – Ejecutar el script de creación de Base de Datos

En la carpeta “sql” del proyecto se encuentra el script de creación de la Base de datos y las tablas contenidas en ella. Debemos ejecutarlo usando el SqlServer Management Studio.

7 – Ajustar la cadena de conexión

Por último nos quedaría ajustar en appsettings.json, uno de los ficheros en la raíz del proyecto, en el apartado “AppConnection” la cadena de conexión a la Base de Datos adecuándola a nuestra instalación de sqlserver.

Ya podremos ejecutar desde el Visual Studio el proyecto.

Instalación para Ejecución

La tecnología .net core permite, aunque no se ha comprobado, la ejecución tanto el Linux, como en Mac o Windows. En este manual nos centraremos en Windows como sistema operativo. Asumiremos también que se tiene instalado el gestor de paquetes chocolatey (<https://chocolatey.org/>)

1 – Instalar .net core 5 sdk

```
choco install dotnet-5.0-sdk
```

2 – Instalar git

```
choco install git.install
```

3 – Obtener el proyecto.

Ejecutaremos el comando git clone <https://github.com/Gabriel-Iban/Inventario5.git>

4 – Compilar el proyecto

En el directorio Inventario5 que acabamos de crear con git clone ejecutaremos el comando:

```
dotnet build --configuration Release
```

Esto nos generará un directorio Inventario5\bin\Release\net5.0 que contendrá el ejecutable y el fichero de configuración.

5 – Crear la Base de Datos

De no tener una Base de Datos sql server disponible, seguir los pasos correspondientes de la instalación para desarrollo para instalar la misma en la máquina que corresponda.

Ejecutar en el SqlServer disponible el script que se encuentra en el directorio sql del proyecto.

6 – Ajustar la cadena de conexión

En el fichero de configuración que se encuentra en el directorio creado con la compilación, ajustar en appsettings.json el apartado "AppConnection". Escribiremos aquí la cadena de conexión a la Base de Datos adecuándola a nuestra instalación de sqlserver.

7 – Ejecutar el programa

Por último, solo nos queda ejecutar la aplicación Inventario5.exe y ya la tendremos accesible en el puerto 5001 por defecto. Puede ser que en la máquina donde se aloje haya que habilitar el puerto en el firewall para que puedan conectarse desde otras máquinas

Documentación técnica

Herramientas utilizadas para la realización del proyecto

SqlServer: Como Base de Datos se ha utilizado SqlServer por su capacidad para realizar gráficos directamente mostrando el diagrama entidad relación directamente desde la herramienta y que, al ser una herramienta de la propia Microsoft se usa habitualmente con tecnologías de la misma empresa como Blazor.

Visual Studio Community 2019: Para la programación se ha utilizado esta herramienta dada su potencia y que es la que, de forma natural se utilizará con Blazor. También se podría haber utilizado VSCode u cualquier otro ide, pero se perderían muchas de las herramientas que Visual Studio proporciona.

StartUml: Para la realización del Diagrama de Clases.

Word y PowerPoint para la redacción de textos como el presente y como herramienta de presentaciones.

Microsoft Blazor

Descripción de la tecnología

Como se ha mencionado en la Introducción, la aplicación está realizada con la nueva tecnología de Microsoft Blazor.

Dicha tecnología forma parte de .net core 5 y por lo tanto tiene licencia MIT con todo lo que ello permite.

Actualmente, en Blazor podemos elegir entre BlazorServer y Blazor WebAssembly. Se ha seleccionado BlazorServer dado el entorno en el que se esperaría ejecutar una aplicación de este tipo. Siendo este entorno clientes siempre conectados y no necesariamente con grandes recursos. Además, permite mejores herramientas por ejemplo en cuanto a depuración, lo que facilita un desarrollo más veloz y exige menos en cuanto a la potencia y modernidad de los navegadores web utilizados como clientes. En Blazor Server no sería obligatorio que los navegadores implementasen el standar WebAssembly. El DOM del cliente se genera en el servidor y se comunica con el cliente a través de Signal-R.

Otra característica de Blazor en general es que el desarrollo se realiza utilizando C#. Este lenguaje lo considero más robusto que typescript y por supuesto que javascript. Esto es fundamental para desarrollos medianos o grandes en los que el código se pueda modificar en el tiempo y por varios desarrolladores distintos. No elimina la posibilidad de utilizar javascript en la ejecución en cliente con lo que se pueden aprovechar librerías existentes sin problema.

Se ha utilizado como gestor de Base de Datos SqlServer de Microsoft por su facilidad de uso en entornos de la misma marca, además del aporte que hace de herramientas gráficas para el diseño de la Base de Datos. Las mismas están incluidas en el propio paquete. No se han utilizado las características avanzadas de la misma como sus búsquedas por texto. Esto, facilita el cambio de base de datos en un futuro si se desea.

Ventajas encontradas

Además de las características que destaca el vendedor como su capacidad de ser ejecutado tanto en Windows, como Linux o Mac, he constatado las siguientes ventajas que me gustaría comentar:

Uso de toda la librería .net core framework

Al utilizar esta tecnología tenemos acceso directo a toda la potencia de .net core estándar y componentes que puedan haber realizado terceros. Por ejemplo, el uso del orm Entity Framework es directo.

Facilidad de depuración

Uno de los problemas en un entorno habitual hoy en día con una parte backend y otra frontend, con en ocasiones lenguajes distintos y herramientas poco integradas con IDEs es la dificultad para depurar el código. Con Blazor Server la depuración es inmediata. Se pueden establecer breakpoints en cualquier parte del código siguiendo desde ahí paso a paso las distintas llamadas a funciones y visualización muy cómoda de cualquier variable.

Data Binding y reactividad

Aunque ya es algo habitual en los entornos cliente como Angular, React o Vue destacar la tremenda facilidad de uso del Data Binding con Blazor Server que se une también a otra ventaja que se mencionará del autocompletado. El mismo es totalmente bidireccional y se usa de forma muy natural. Cualquier modificación en los datos se refleja en la presentación sin tener que usar librerías que faciliten esa reactividad.

Autocompletado de código

Con el uso de Typescript, lenguaje promovido por Microsoft, y hoy utilizado por los principales frameworks de cliente, se ha mejorado mucho el autocompletado en el desarrollo web de cliente. Los diferentes IDEs pueden obtener la información de typescript mucho mejor para proponer esas sugerencias con las que autocompletar. En cualquier caso está muy lejos de lo que Blazor permite que es una auténtica maravilla. No ya dentro del código C# sino en medio del html podemos escribir una @ y el entorno nos propone variables o cualquier cosa que proceda agregar en ese punto. Si escribimos el nombre de un objeto inmediatamente nos propone sus propiedades o sus métodos mostrando además los parámetros del mismo.

Facilidades para Refactorización

Otra ventana del uso de C# y las capacidades de un IDE como Visual Studio es que si, por ejemplo, no hemos dado un nombre adecuado a una clase, variable, propiedad, método cambiarle el nombre es inmediato. El entorno se encarga de buscar dónde, dentro del código del proyecto se usa y le cambia el nombre.

Uso inmediato de componentes y facilidad de comunicación con los mismos

Todos los frameworks principales de cliente usan componentes. Blazor no solo no es menos, sino que facilita tremendamente el uso y creación de dichos componentes. En Visual Studio directamente tenemos la opción de crear el componente y ya podremos usarlo en cualquier página añadiendo la directiva using del paquete donde se encuentre el componente creado. El paso de variables al mismo es inmediato mediante el decorador [parameter] y el mismo puede llamar a eventos definidos en su contenedor simplemente pasándole variables de tipo Action al componente. El componente puede llamar a dichas Action y pasarle parámetros a las mismas por lo que la comunicación contenedor-componente y componente-contenedor es inmediata.

Facilidad de enrutado

En los frameworks cliente que conozco es necesario definir rutas en algún lugar y luego implementar el componente/página que hemos definido en un fichero distinto. Con Blazor en la directiva @page de cada página definimos la ruta así como los parámetros y no es necesario definirlo en ningún otro sitio lo que minimiza la introducción de errores por incongruencias.

Desventajas encontradas

En cuanto a la parte negativa destacaré lo siguiente todo ello como resultado de que la tecnología es muy reciente. Blazor Server se lanzó en septiembre de 2019. Blazor WebAssembly en mayo del presente 2020.

Falta de documentación

No hay gran cantidad de documentación, libros, preguntas y respuestas, etc disponibles.

Falta de bibliotecas

Además no hay gran cantidad de bibliotecas de componentes que nos permitan dar un mejor aspecto y funcionalidad a nuestras aplicaciones. En todo caso menciono algunas gratuitas.

- <https://blazorise.com/> para usar Bootstrap de forma nativa
- <https://www.matblazor.com/> para usar Material Design
- <https://github.com/Blazored> conjunto de componentes

Y de pago:

- <https://www.syncfusion.com/blazor-components>
- <https://www.telerik.com/blazor-ui>
- <https://www.grapecity.com/componentone/blazor-ui-controls>
- <https://www.devexpress.com/blazor/>

Navegadores antiguos no soportados

Aunque Blazor Server sí soporta navegadores antiguos aunque necesita que soporten SignalR, Blazor Web Assembly necesita que el navegador soporte Web Assembly que es un estándar más reciente aunque en la actualidad ampliamente aceptado.

Pasamos a explicar la aplicación en sí:

Análisis de la aplicación

Requisitos

Partamos de que queremos una aplicación para almacenar elementos de los cuales queremos llevar un inventario. En principio Portátiles, Estaciones de Trabajo y Monitores pero se prevé que puedan añadirse otro tipo de elementos en el futuro.

Cada elemento se identificará físicamente por un código. Dicho código se pegará físicamente al elemento por medio de pegatinas. En principio el código de un elemento no cambiará pero no se descarta que pueda haber “recodificaciones” si se cambia la manera de codificar algún elemento.

También de todo elemento se almacenará un texto como notas de dicho elemento que será un texto libre.

Por último, de cada elemento se podrá almacenar una persona relacionada, la ubicación donde se encuentra y a qué marca y modelo pertenece dicho elemento.

En cuanto a los Portátiles almacenaremos:

Un nombre del equipo (host name), su número de serie, la cantidad de memoria ram, la cantidad de espacio en disco, el tipo de procesador y la velocidad del mismo. También almacenaremos el tamaño en pulgadas de su pantalla.

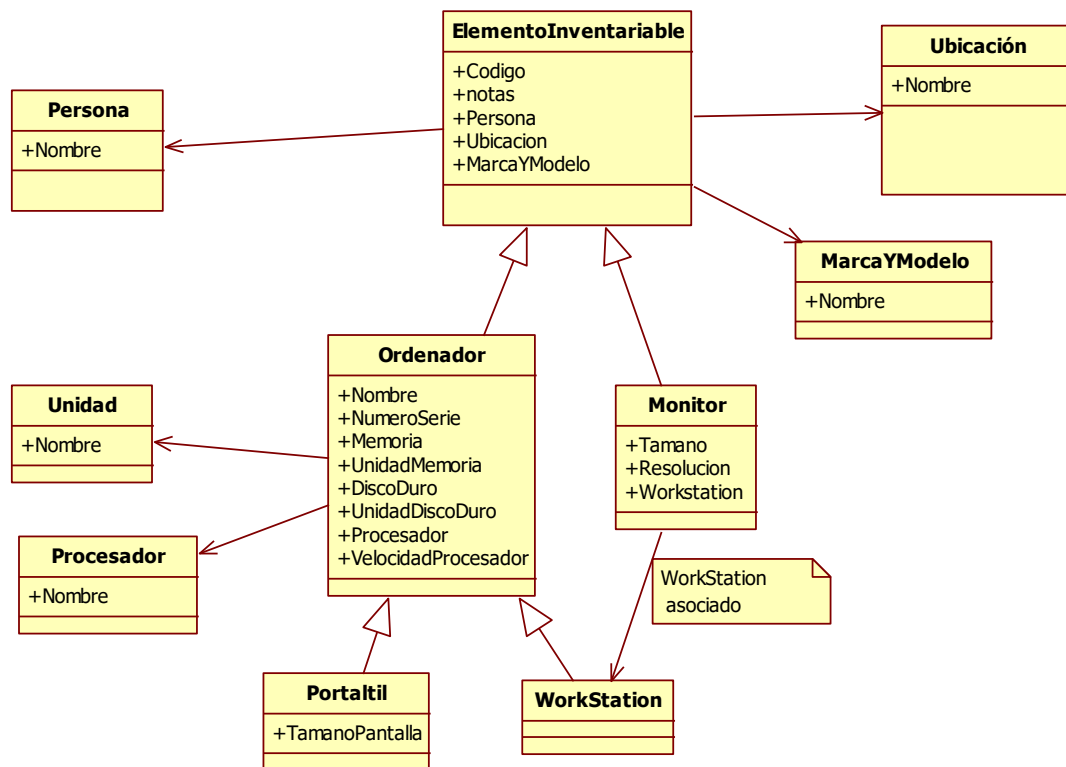
Sobre las Estaciones de Trabajo almacenaremos lo mismo que para los portátiles salvo el tamaño de pantalla dado que no tendrá pantalla en sí, sino que tendrá monitores asociados al él.

En cuanto a los monitores no almacenaremos su número de serie por ser elementos más baratos y por tanto no es necesario un control tan estricto, pero sí se almacenará su tamaño de pantalla en pulgadas así como su resolución máxima. Por último se almacenará la Estación de trabajo a la que está conectada el monitor.

Actores.

No se ha considerado el uso de Actores. Todos los usuarios serán iguales y permite no tener que realizar validación de usuarios. Esto facilita enormemente las pruebas de la aplicación.

Diagrama de clases.



Explicaré someramente las razones que nos llevan a este diagrama de clases:

Como elemento principal tenemos el “ElementoInventariable”. Partimos de la base de que todo elemento inventariable tendrá una serie de elementos comunes y por eso forma parte de la base de una jerarquía de clases. En el futuro, cualquier elemento que se pretenda inventariar deberá heredar en algún nivel de esa clase.

Dado que los elementos persona, ubicación y MarcaYModelo se pueden repetir para varios elementos inventariables lo extraemos a clases aparte. También son elementos que en el futuro se podría relacionar con otros elementos como personas con la gestión de usuarios.

También aporta a todo elemento un código (que no será la clave primaria en su plasmación en el diagrama ER dado que un usuario podría querer cambiarlo).

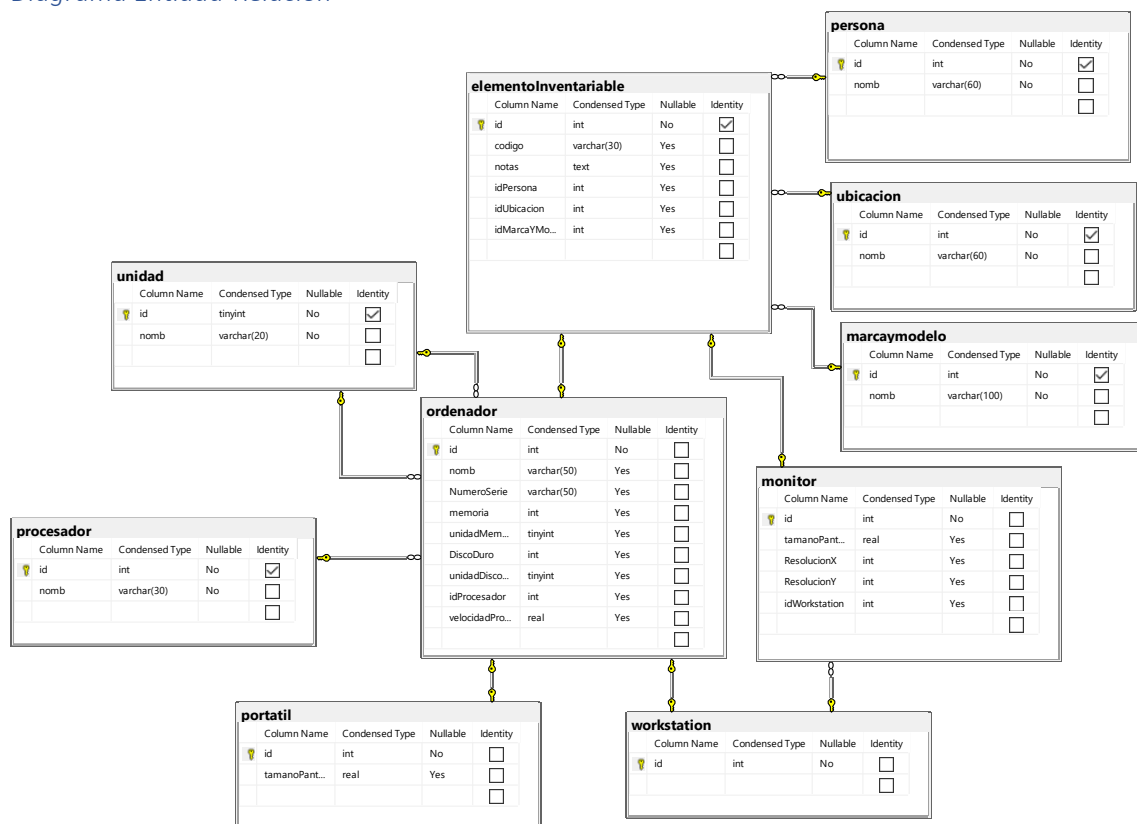
Para reflejar los elementos comunes entre Portátil y Estación de Trabajo se crea también la clase Ordenador.

Por último destacar del modelo que se han creado las clases Unidad y Procesador para facilitar la codificación de estos elementos en la clase Ordenador.

Lista de casos de uso.

No se detalla la lista de casos de uso dada la simplicidad de todos ellos restringiendo su utilidad a validaciones que en su mayoría gestiona la propia integridad referencial de la Base de Datos.

Diagrama Entidad-Relación



Destacaré algunos elementos que llaman la atención del ER:

La implementación de la Herencia:

En primero lugar hablamos de la implementación de la herencia. SGBD como postgres permiten mediante extensiones de sql declarar esa herencia. En Sql Server, el SGDB seleccionado para este proyecto, no permite esas capacidades por lo que se ha implementado como relaciones entre tablas.

Hay tres formas de implementar la herencia: Todos los elementos de todas las clases en una sola tabla. Una tabla por cada clase final en la que se almacenan todos los elementos comunes y finalmente una tabla por cada clase en la que se almacenan los elementos de esa clase y se relaciona con su clase padre por la clave primaria siendo esta clave primaria y clave

foránea al mismo tiempo. Se ha seleccionado esta última forma según se puede observar en el diagrama ER.

Claves primarias.

Todas las claves primarias están formadas por un solo campo y se llama id. Se le ha cambiado el tipo de int a tinyint cuando por la baja cantidad de elementos esperados no era necesario un campo de mayor tamaño.

Claves Foráneas.

A las claves foráneas, siempre que ha sido posible, se les ha nombrado con id + [Nombre de tabla referenciada]. En la relación de ordenador con unidad, por haber dos claves foráneas a la misma tabla se ha buscado un nombre adecuado.

Tablas de referencia:

Las tablas: Persona, Ubicación, MarcaYModelo, Unidad y Procesador; siguen todas la misma estructura. El id que las referencia y un nombre. En varias de ellas como Ubicación y MarcaYModelo sería lógico emplear una estructura de árbol teniendo por ejemplo ubicaciones dentro de ubicaciones. Esto se comentará en las posibles ampliaciones del proyecto. Se ha utilizado esta estructura para no complicar excesivamente el modelo y la programación que alimentase dicho modelo.

Descripción de algunos de los campos:

ElementoInventariable:

Código: En este campo se almacenará el código que las personas asociarán al objeto. Por lo tanto no se podrán repetir valores y es un campo por el que se espera acceder al registro. Teniendo en cuenta ambas cosas se le ha creado un índice único.

Notas: Texto en el cual el usuario almacenará cualquier cosa que tenga que ver con el ElementoInventariable.

Ordenador:

Nombre: Almacenamos aquí el nombre (host) del equipo. Se ha creado por tanto un índice único asumiendo que no se puede repetir dentro de la organización.

NumeroSerie: Almacenamos aquí el número de serie del ordenador. También con índice único dado que no permitiremos que se repita.

Memoria y Disco Duro: En estos campos almacenaremos la cantidad correspondiente a cada elemento.

UnidadMemoria y UnidadDiscoDuro: En estos campos se referencia la tabla de unidad cada uno de ellos para el concepto que representa.

Portátil:

TamañoPantalla: Se almacena el tamaño en pulgadas de la pantalla.

Workstation:

Únicamente se almacena en esta tabla el id para identificar que “es” una estación de trabajo. En el futuro es probable que dicha tabla aumentase su número de campos con elementos concretos de estación de trabajo que se quisiesen almacenar.

Monitor:

ResolucionX y ResolucionY: Únicamente destacar en esta tabla que la resolución se ha almacenado como dos campos independientes.

Implementación

Solamente añadiré algunos comentarios sobre la implementación.

Tipos de páginas

Se han creado un par de páginas típicas: Las que acaban en “FE” y las que acaban en “FL”.

Formulario de Edición: Las páginas que acaban en FE (Formulario Edición) sirven para editar un registro. Tanto al añadirlo, modificarlo o eliminarlo.

Formulario de Lista: Las páginas que acaban en FL (Formulario de lista) sirven tanto para ver la lista de elementos de un tipo como para seleccionar un elemento concreto de ese tipo.

Otros comentarios

Además de los directorios que por defecto aparecen en una solución de Blazor Server se han añadido. “Components” para componentes y “Clases” dentro de Shared para clases que se usan a lo largo de toda la aplicación.

Para mantener el estado de la aplicación entre llamadas y permitir cosas como que a partir de una lista de estaciones de trabajo, seleccionar una persona y dentro de seleccionar una persona se pueda crear una persona para luego seleccionarla se ha usado una clase History para almacenar los estados por los que va pasando la aplicación y comunicar ese estado con otras páginas o consigo misma en el tiempo.

Para el trabajo con la Base de Datos se ha utilizado el ORM EF Core.

Se ha seguido la filosofía “Database first” y para generar el modelo se ejecuta el siguiente comando dentro del proyecto Inventario5:

```
dotnet ef dbcontext scaffold  
"Server=localhost\SQLEXPRESS;Database=InventarioDB;Trusted_Connection=True;"  
Microsoft.EntityFrameworkCore.SqlServer -o Models -force
```

Habría que ajustar la cadena de conexión a su instalación específica.

Posibles mejoras a la aplicación

Funcionales

Existe gran cantidad de mejoras funcionales que se podrían hacer a la aplicación. De hecho podría ser la típica aplicación que no se acabase nunca, siendo una aplicación que se adaptase al funcionamiento del inventario de la empresa y pasando de ser únicamente informático a ser el inventario general de la misma. Pero mencionaré algunas.

Paso de lista a árbol en algunas tablas

Las tablas, tanto de Ubicación, como de Marca y Modelo deberían ser árboles en lugar de listas de manera que, si yo defino, por ejemplo una Marca pueda incluir los Modelos dentro de la misma. Lo mismo pasaría con Ubicación. Si por ejemplo defino un centro de trabajo debería poder definir las distintas partes de ese centro, pero dentro siempre del centro de trabajo definido.

Obtención de datos de fuentes externas

Por ejemplo, en lugar de tener una tabla de personas tan “simple” se podría implementar la importación o enlazado de datos de otras fuentes, por ejemplo, de un directorio Activo.

Implementación de control de usuarios

Se podrían implementar roles de manera que hubiese usuarios que solo pudiesen acceder a cierta información pero no modificarla etc.

Creación de Elementos Inventariables por parte del usuario.

Esta sería la gran mejora pero implica gran dificultad y trabajo. La idea es que un usuario pudiese definir características de un elemento y dicho elemento se almacenase en la base de datos.

Por ejemplo, que la empresa decidiese almacenar información sobre impresoras. Definiese características de dicho elemento y que el sistema le crease la estructura en Base de Datos para almacenar ese elemento.

También tendría que crear la forma de listar ese elemento y de añadirlo, modificarlo y eliminarlo.

Por supuesto necesitaría también la manera de crear “Tablas de Referencia” como en el modelo actual son Unidad, o Procesador.

Técnicas

Uso de EF Core con herencia de tablas

El ORM EF Core, en versiones anteriores a la 5 tenía una implementación parcial del concepto de herencia entre tablas en las que se permitía el primer uso descrito de la herencia. El que almacena los datos de toda la jerarquía de herencia en la misma tabla. Con la versión 5 ya se puede tener una parte en la tabla que hace de parte padre de la jeraquía y la parte correspondiente en los hijos.

Refactorización

Hay partes del código que se repiten a través del programa y que Blazor facilitaría que se implementasen en clases y componentes comunes. Por ejemplo incluir el botón de acción de los formularios de edición, con su funcionalidad de reflejar la operación que se está realizando, como componente aparte. Este a su vez formar parte de una botonera estándar para dicho tipo de formulario. Esto permitiría que si en un momento dado se quiere cambiar la funcionalidad o

estética de dichos botones se haga en un sitio para toda la aplicación. Si en un formulario en concreto se requiere otra funcionalidad o estética habría que implementarlo aparte.

Uso de librerías de controles

La estética y funcionalidad es la típica de Bootstrap. Con la implementación de controles de terceros se podría mejorar ambas. Por ejemplo el uso por ejemplo de componentes de grid que permitan definir una fila activa y que ya implementan la botonería de edición y eliminación. Orden y filtrado de elementos así como paginación en su caso mejoraría el uso de la aplicación.