

DICIEMBRE DE 2021

PREDICCIÓN DE INCENDIOS UME



TRABAJO REALIZADO POR:
Jorge Prieto Medina

Explicación del proyecto:

El proyecto que he decidido hacer es una aplicación de escritorio para la predicción de incendios en el que los usuarios pueden saber la probabilidad de incendio, en cualquier época del año y cualquier localidad de toda España, la aplicación solo estará disponible para ordenadores con el sistema operativo Windows.

La aplicación se basa en la realización de cálculos del porcentaje de la probabilidad de un incendio a través de los datos almacenados de incendios de otros años, dependiendo del lugar y de la época del año, y de las condiciones climáticas que se esperan para esas fechas, datos como la temperatura y el grado de humedad, además también se puede considerar el grado de limpieza de los bosques de la zona, pues es más difícil que ocurra un incendio en un bosque sin desechos y sin ramas y hierbas secas.

En esta aplicación los administradores podrán registrar incendios en una base de datos, la cual los almacenará por zona y época del año, estos serán los principales datos que la aplicación usará para determinar la probabilidad de que ocurra un incendio.

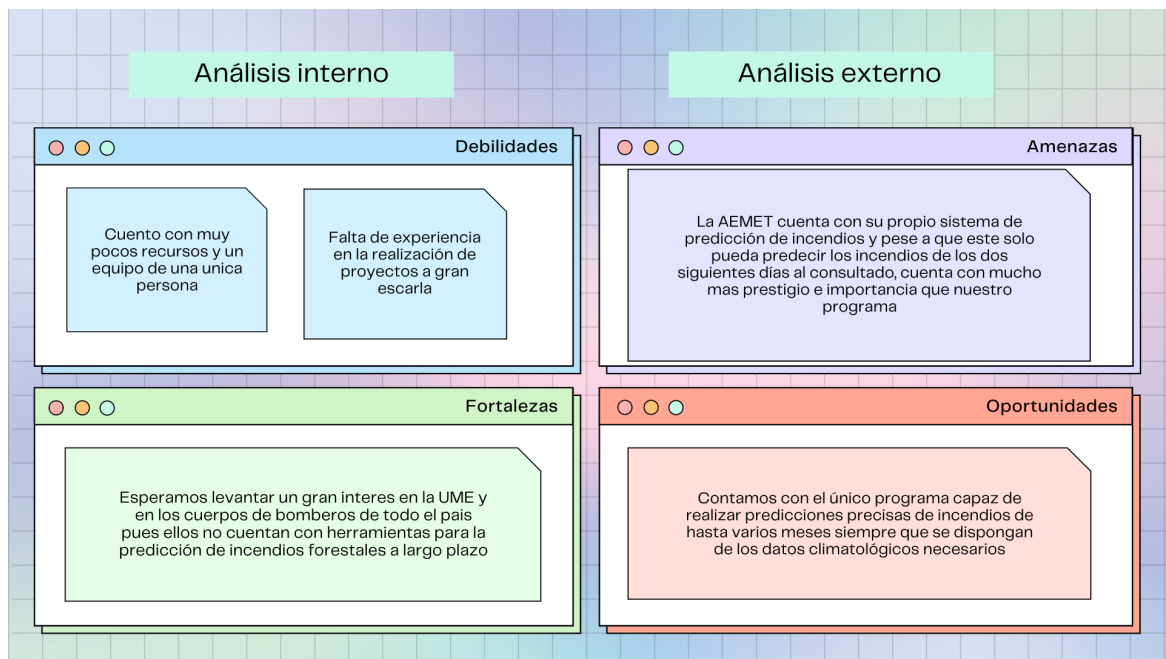
Aunque los administradores han de estar registrados en la base de datos, los demás usuarios no, por lo que podrán acceder a la aplicación y buscar las probabilidades de un incendio o mirar el historial de incendios, lo cual es la principal función de la aplicación.

El proyecto además será desarrollado en C# wpf y almacenará los datos en una base de datos de microsoft SQL Server a la cual se accederá a través de un ADO.NET Entity Data Model

Estudio de mercado:

Este estudio de mercado se basa en el análisis de otras aplicaciones que ofrecen características similares a las ya mostradas en la explicación del proyecto

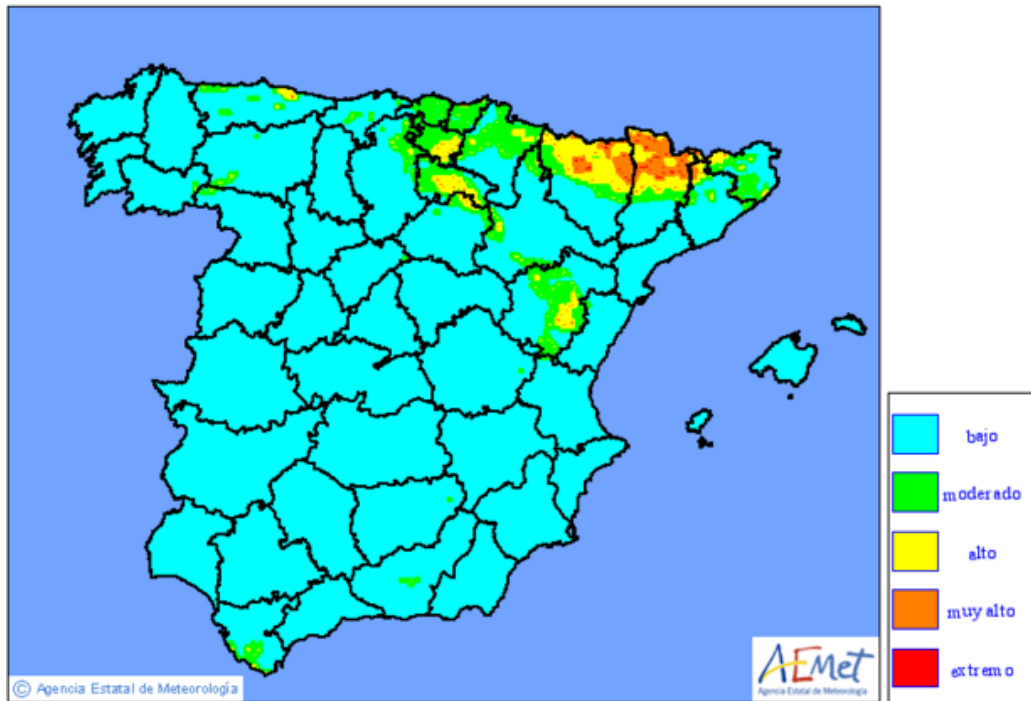
Análisis Dafo:



Competencia en el sector:

La principal competencia que tiene mi programa es la página de predicción de incendios de la AEMET, que es la Agencia Estatal de Meteorología, cuya página aunque cubre toda España de una forma bastante exacta, se encuentra muy limitada:

Por ejemplo la página de la AEMET no habla de porcentajes exactos, sino que cuenta con 5 niveles de riesgo, siendo más impreciso.



Además, la página de la AEMET solo predice incendios en dos días, estando muy limitada en este aspecto, además de que no cuenta con un historial de incendios. Al contrario, mi aplicación es capaz de mostrar porcentajes con hasta meses de antelación y con números exactos, por lo que sería más precisa que la página de la AEMET

Otra página similar a mi aplicación es la European Forest Fire Information System, la cual es una página europea que muestra el historial de incendios de toda Europa, cuales son los incendios activos actualmente y las anomalías en las temperaturas registradas.

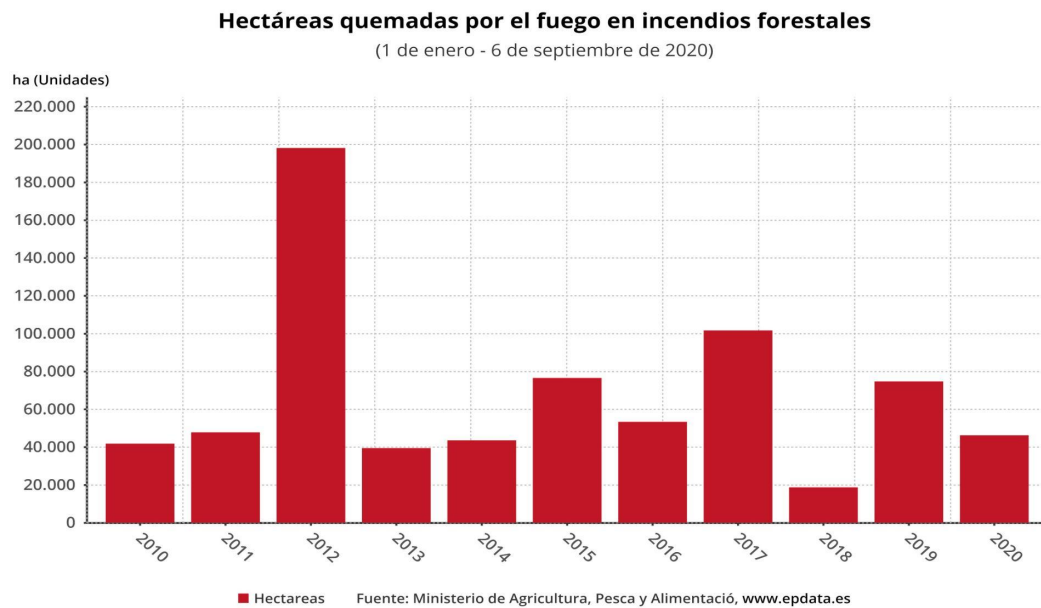
En cuanto a programas instalables, no hay ninguna oferta similar a lo que ofrece mi aplicación aparte de alguna que muestra los incendios activos como firemap

Estrategia comercial

Nuestra estrategia comercial no solo se basa en el uso civil, esa es una parte secundaria de la aplicación, su principal objetivo es proporcionar ayuda

a los cuerpos de bomberos de toda España y a la Unidad Militar de Emergencias

Gracias a esta aplicación las unidades de emergencias podrán desplegarse con antelación en las zonas con mayor riesgo de incendio, minimizando así los daños producidos por un posible incendio, pues como podemos ver en la siguiente gráfica cada año se pierden una gran cantidad de hectáreas por culpa de los incendios



Por esto considero que este programa sería una herramienta muy útil en el campo de predicción y extinción de incendios, campo en el cual no hay muchas herramientas digitales que ayuden tanto a la población como a los servicios de emergencia

Canales de distribución:

Al tratarse de una aplicación de escritorio, el único canal de distribución sería la instalación del programa en un ordenador con sistema operativo Windows

Valor del producto:

Como ya se comentó en el punto de estrategia comercial el principal valor del producto es la ayuda a la población y a las unidades de emergencia a la hora de predecir un incendio, algo en lo que somos pioneros en el sector

Análisis de la solución:

Requisitos funcionales:

Explicación: los requisitos funcionales son aquellos que definen una función del sistema de software o sus componentes, es decir, los requisitos funcionales establecen los comportamientos del software.

Así mismo, una función es una serie de entradas, comportamientos y salidas los cuales varían desde cálculos, manipulación de datos y otro tipo de acciones.

Los requisitos funcionales de este programa son:

Nº 1: El programa debe dejar registrarse a los administradores.

Nº 2: El programa debe impedir el paso de los usuarios normales (sin registrar) a las partes de administración, y permitirle el acceso al resto de la aplicación.

Nº 3: El sistema debe permitir a los administradores activar las cuentas de los nuevos administradores registrados.

Nº 4: El sistema debe avisar al usuario que se intente registrar si el nombre que ha elegido ya está en uso.

Nº 5: El programa debe dejar iniciar sesión.

Nº 6: Los administradores podrán acceder a cualquier parte del programa, incluida la parte de los usuarios sin registrar.

Nº 7: Los administradores podrán añadir nuevos registros de incendios y sus datos en un formulario.

Nº 8: El sistema debe procesar los registros nuevos y añadirlos a su tabla correspondiente en la base de datos.

Nº 9: El usuario podrá acceder a las estadísticas y probabilidades de que ocurra un incendio, pudiendo elegir la localidad y la época del año.

Nº 10: El usuario podrá acceder al historial de incendios los cuales serán visualizados en una tabla además los podrá ordenar y filtrar por zona, localidad o ambos.

Nº 11: El programa realizará los cálculos de forma automática dependiendo del historial de incendios en la zona elegida y la época del año

Nº 12: El programa debe dejar a los administradores modificar o eliminar registros de incendios, los cuales serán seleccionados en una tabla

Nº 13: El programa permitirá al usuario generar documentos PDF de el historial de incendios

Nº 14 El administrador podrá introducir nuevos datos climáticos estimados por localidad y el sistema detectará y avisará si ya existían tales datos

Nº 15 El administrador podrá realizar log out y entrar con otra cuenta

Requisitos no funcionales:

Explicación: los requisitos no funcionales, son los que especifican criterios que pueden usarse para juzgar la operación de un sistema en vez de sus comportamientos específicos puesto que estos corresponden a los requisitos funcionales, es decir, estos requisitos se refieren a características de funcionamiento de la aplicación y no a funciones propias del programa

Los requisitos no funcionales de este programa son:

Nº 1: La base de datos será SQL y estará instalada en SQL Server

Nº 2: El programa será desarrollado en WPF lo que permite una interfaz y funcionalidad más moderna

Requisitos de Información:

Los requisitos de información, los cuales son la información que se almacenará en la base de datos son:

Nº 1: Información de usuarios:

- Nombre de usuario
- Contraseña
- Cuenta activada
- Email

Nº 2: Información de incendios:

- Código de la localidad
- Época del año
- Temperatura medía durante el mes en la localidad
- Hectareas quemadas
- Humedad ambiental
- Tiempo de extinción del incendio

Nº 3: Información de las localidades:

- Nombre de la localidad
- Código de la localidad

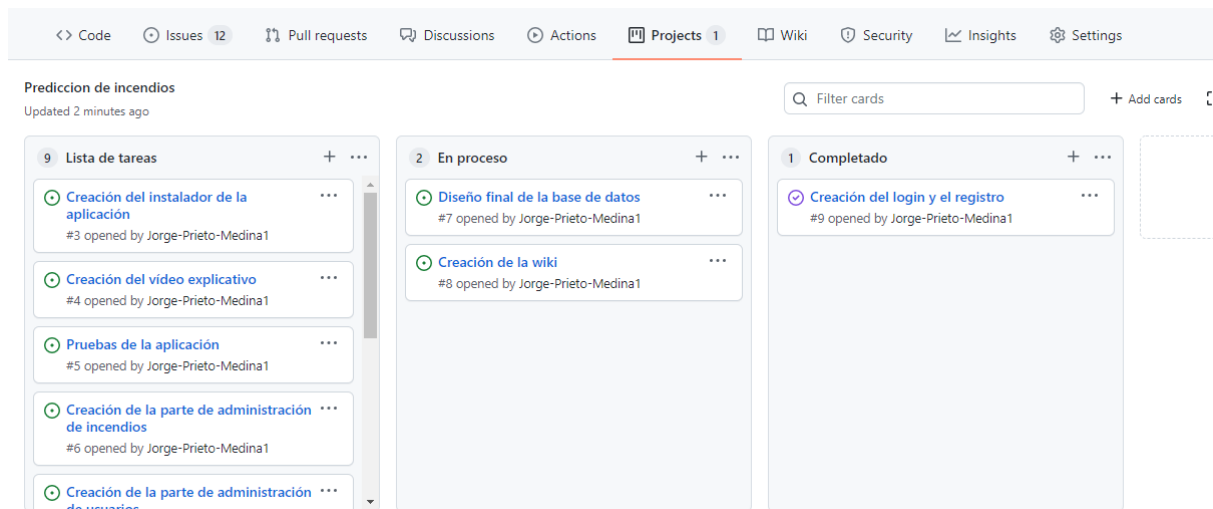
Nº 4: Datos climáticos estimados por mes:

- Código de la localidad
- Temperatura media estimada para el mes
- Humedad media estimada para el mes
- Año y mes

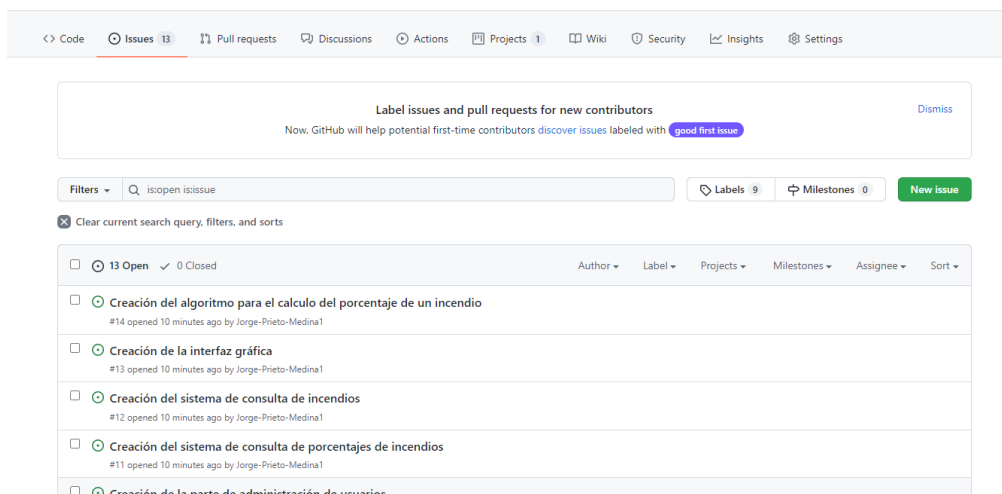
Planificación de tareas y estimación de costes

Este es el apartado de la planificación de tareas y la planificación de costes, parte necesaria en todo proyecto serio, para no exceder los plazos de entrega establecidos y producir un sobrecoste

El proyecto de predicción de incendios usa las herramientas de planificación de git cómo Projects:



En projects de git se pueden planificar las issues o tareas del proyecto, su tiempo de duración, su plazo estimado, etc y su estado ya sea en tareas pendientes, en proceso o ya finalizadas



Tareas y tiempo estimado

Las tareas principales de este proyecto son:

- Creación del login y el registro: tiempo estimado 6 horas, tiempo real 8 horas
- Diseño de la base de datos: tiempo estimado 1 hora, tiempo real 1 hora
- Creación de la wiki del proyecto: tiempo estimado 10 horas ,tiempo real 5 horas
- Diseño del algoritmo que calcule el porcentaje de incendios: tiempo estimado 2 horas, tiempo real 3 horas
- Creación del login y el registro de la aplicación: tiempo estimado 4 horas, tiempo real 6 horas
- Creación de la interfaz gráfica: tiempo estimado 5 horas, tiempo real 10 horas
- Creación del sistema de consulta y modificación de usuarios: tiempo estimado 3 horas, tiempo real 4 horas
- Creación del sistema de consulta y modificación de localidades: tiempo estimado 4 horas, tiempo real 5 horas
- Creación del sistema de consulta y modificación de datos climatológicos: tiempo estimado 4 horas, tiempo real 6 horas
- Creación del sistema de consulta y modificación de incendios: tiempo estimado 4 horas, tiempo real 2 horas
- Creación de la parte de administración de usuarios: tiempo estimado 4 horas, tiempo real 5 horas
- Realización de las pruebas de la aplicación: tiempo estimado 3 horas, tiempo real 4 horas

- Creación del video explicativo: tiempo estimado 2 horas, tiempo real 2 horas

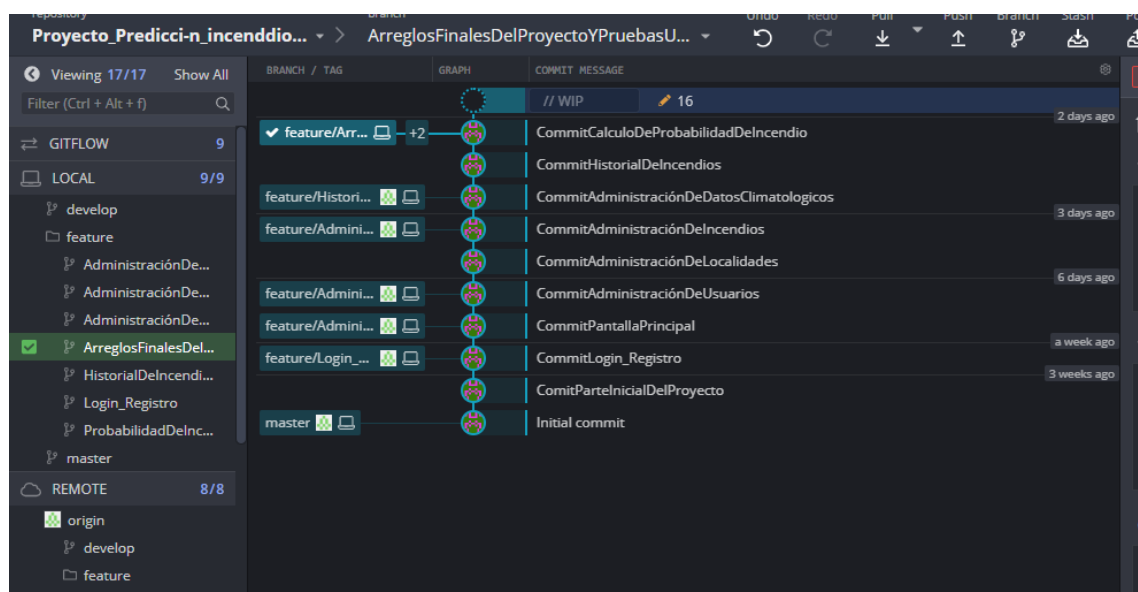
La planificación de las tareas se han organizado de este modo:

- **Presentación de la explicación del proyecto:** El 18 de octubre es la fecha límite para realizar una explicación resumida de en qué consiste mi proyecto Predicción de incendios
- **Análisis de la solución:** El 22 de octubre es la fecha límite para determinar los requisitos funcionales, es decir, que hace la aplicación, los requisitos no funcionales y los requisitos de información necesarios en el proyecto
- **Presentación del estudio de mercado:** Siendo el 25 de octubre la fecha límite para crear un estudio de los elementos y soluciones similares a nuestro proyecto ya existente en el mercado y determinar el valor del producto, es decir la cualidad que hace destacar a nuestra aplicación por encima del resto
- **Análisis de la solución:** El 29 de octubre es el último día para la definición de los actores de la aplicación y la creación del diagrama de casos de uso
- **Planificación de tareas:** El 1 de noviembre es la fecha límite para la planificación de tareas
- **Estimación de costes:** El 5 de noviembre es la fecha límite para la estimación de los costes, dependiendo de las horas invertidas y los programas utilizados y para la planificación de la gestión de los riesgos laborales
- **Diseño de la interfaz de usuario:** El 8 de noviembre es la fecha límite para la finalizar el diseño de la interfaz de usuario y la creación del prototipo
- **Diseño del diagrama de clases y de la persistencia de la información:** El 12 de noviembre es la fecha límite para la creación del diagrama de clases, el diagrama de persistencia de la información y la explicación del porqué del tipo de base de datos elegido

- **Diseño de la arquitectura del sistema:** El 15 de noviembre es la fecha límite para diseñar y explicar la arquitectura del sistema
- **Análisis tecnológico:** El 19 de noviembre es la fecha límite para analizar la tecnología elegida y la susceptible de ser utilizada
- **Implementación de la solución:** El 3 de diciembre es la fecha límite para justificar los elementos que van a ser implementados en el programa
- **Entrega del proyecto:** El 10 de diciembre es la fecha límite para la entrega del proyecto, incluyendo el repositorio con todo el código, la documentación completa, la explicación del código, casos de pruebas y la creación de un video mostrando el proyecto

Para la organización y el guardado del código del proyecto he usado el Git Flow en Git Kraken como flujo de trabajo, para este cometido he creado 2 ramas principales: la rama Master, la cual es la principal y de la que sale la otra rama: la develop desde donde se crean las features o ramas secundarias que se usan para cada parte del proyecto y que luego se juntan con la develop la cual al final del proyecto se une a la master

Imagen del git Flow del proyecto:



Por último, he añadido la información sobre el proyecto y la documentación a la wiki del github del proyecto, para que cualquier usuario pueda acceder sin ningún problema a esta

Estimación de costes y recursos:

En este apartado hablaremos de la estimación de los costes que conlleva el desarrollo del proyecto, desde el hardware y el software en el que se desarrollan, el coste del desarrollo de el programa y de su publicación en la tienda de windows

Coste del hardware y el software del proyecto

Puesto que yo ya disponía de ordenador para realizar el desarrollo, no voy a considerar su precio como un gasto, pero como la base de datos se encuentra en SQL Server se necesitaría un ordenador donde montar el servidor

- Precio del servidor 600 €
- Precio mensual para el mantenimiento del servidor y demás gastos: 25 € mensuales

En cuanto al coste del software se han utilizado herramientas en sus versiones gratuitas: para el código visual studio individual que es para proyectos individuales y estudiantes, en cuanto al servidor Microsoft SQL server edición desarrollador y para la administración del código GitHub que es completamente gratuito

Pero, en el caso de publicarse la aplicación sería necesario el uso de las versiones de pago tanto de visual studio y de SQL server:

En cuanto al coste de SQL server en su versión Enterprise, tiene un coste de 250 dólares al mes

Suscripción a Enterprise

- IDE de Visual Studio Enterprise
- Azure DevOps (plan básico + plan de pruebas)

Suscribirse por 250 USD

Al mes

Y SQL server también en su versión enterprise un coste total de 133.748 dólares

Precios de SQL Server 2019

Ediciones	Precio de Open No Level (USD)	Modelo de licencia	Disponibilidad de canal
Enterprise	\$13, 748 ^[1]	Lote de 2 núcleos	Licencias por volumen, hosting
Standard: por núcleo	\$3,586 ^[1]	Lote de 2 núcleos	Licencias por volumen, hosting
Standard: servidor	\$899 ^[1]	Servidor ^[2]	Licencias por volumen, hosting
Standard: CAL	\$209	CAL	Licencias por volumen, hosting
Developer	Gratis	Por usuario	Descarga gratuita
Web	Consulta los precios con tu partner de hosting	No aplicable	Solo hosting
Express	Gratis	No aplicable	Descarga gratuita

En cuanto al coste de la publicación del programa en la tienda de windows en España, tendría un coste de 14 euros

Coste de la mano de obra

En cuanto al coste de la mano de obra está calculado con respecto a las horas necesarias para realizar el proyecto y la documentación al precio de 8 € la hora lo cual es un ligero incremento con respecto al sueldo mínimo en España que son 7,40 € a la hora y es lo que creo que aproximadamente cobraría un programador recién titulado

Tiempo estimado del desarrollo de la aplicación: 60 horas considerando retrasos lo que sería un total de 480 €

Tiempo estimado para la creación de la documentación: 40 horas o en euros: 320 €

En cuanto a los otros gastos que conlleva el desarrollo del software como puede ser el gasto energético y los del lugar de trabajo, calculo un gasto total de unos 80 € al mes y puesto que el desarrollo del programa dura un total de dos meses serían unos 160 €

Herramientas usadas

Las herramientas usadas para el desarrollo del proyecto son: para el desarrollo Visual Studio code edición individual, para la base de datos Microsoft SQL server edición desarrollador y para la administración del código y de repositorios GitHub

Gestión de riesgos

Riesgos de la planificación:

- El total de horas estimadas es demasiado optimista, pues no se tienen en cuenta retrasos por fallos en el programa y otro tipo de retrasos
El tratamiento se basa en priorizar las partes más importantes de la aplicación dejando la parte visual para el final del desarrollo
- Si se producen demasiados retrasos por culpa de inconvenientes con la programación del programa, la calidad del resto de los componentes de la aplicación puede disminuir por falta de tiempo
El tratamiento en este caso al igual que en el anterior es priorizar las partes y funcionalidades más importantes

Riesgos de organización

- Trabajando una sola persona en el proyecto que además tiene una jornada laboral normal deja poco tiempo para realizar el trabajo

Tratamiento: realizar el proyecto poco después de terminar de trabajar y aprovechar mucho los fines de semana, dedicando especial atención a las tareas más importantes del proyecto

Riesgos del producto

- Es muy probable que algunas partes del programa contenga bugs y otros problemas por lo anteriormente expuesto
Tratamiento: realizar pruebas unitarias y comprobar cada parte del programa después de desarrollarlo
- El uso de microsoft sql server requiere de un ordenador que actúe como servidor en remoto
Tratamiento: esta primera versión del programa contará con su propio servidor instalado en su misma máquina para realizar las pruebas

Riesgos personales

- El proyecto solo cuenta con un trabajador aunque este ya tiene experiencia tanto en SQL Server como en C#
Tratamiento: se empleará la mayoría del tiempo en las partes principales del proyecto

Análisis de escenarios

En el análisis de escenarios analizaremos los actores y los caso de uso del programa Predicción de incendios de la UME sin antes explicar que son:

Los actores son los distintos usuarios del programa, cada actor interactúa de forma distinta con el programa y sus casos de uso

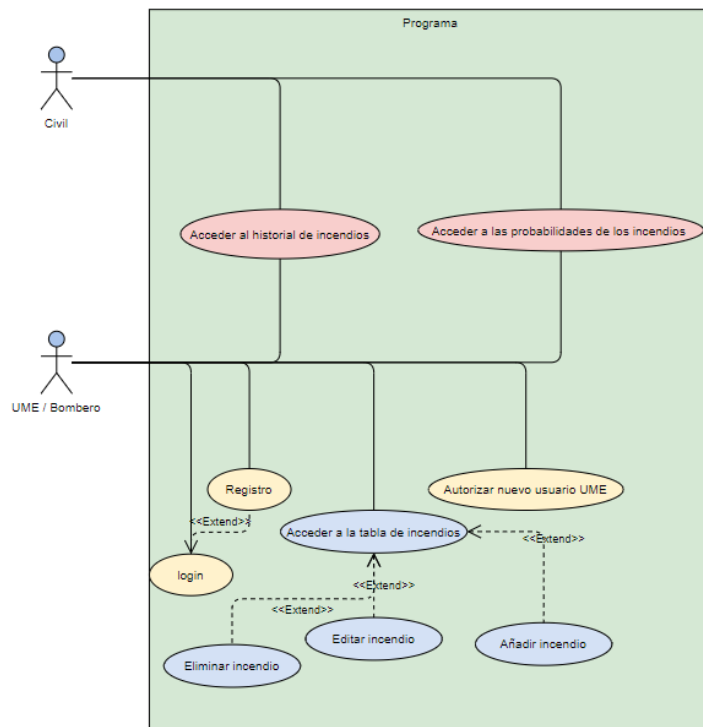
Los casos de uso son cada uno de los casos de uso que se producen dentro de nuestro programa

Actores:

Actor civil: el usuario civil es cualquier persona que quiera informarse sobre las posibilidades de un incendio o del historial de incendios

Actor UME o bombero: estos actores son administradores y son capaces de añadir nuevos incendios , aunque su objetivo real es usarla para poder desplegarse cerca de lugares donde se prevean incendios

Diagrama de casos de uso del programa



Esta imagen muestra el diagrama de los casos de uso del programa, como podemos ver el usuario “civil” el invitado solo tiene acceso a ver los registros de los incendios y las probabilidades que tienen estos de ocurrir mientras que los administradores pueden acceder a toda la funcionalidad del programa

Diseño de la solución:

Interfaz de usuario y prototipos

En este apartado de la documentación analizamos la parte visual del proyecto: el prototipo, la paleta de colores y el logo del proyecto

Logotipo del proyecto



Para el logotipo del proyecto he elegido el escudo de la UME o unidad militar de emergencias, la principal razón por la que lo he escogido es porque el programa está principalmente orientado al uso de la UME y de los bomberos mas que a los civiles normales, pues son ellos los únicos que pueden registrarse como administradores y realizar cambios en los datos, los civiles solo pueden acceder a la información pero no modificarla

Paleta de colores:

En cuanto a la paleta de colores contamos con cuatro colores principales y un color derivado de otro:

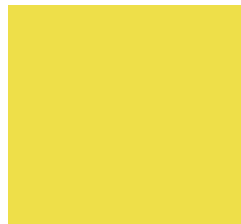
- El primero de los colores es el rojo, este color, junto con el azul y el amarillo representan los colores del escudo de la UME, su código es: #EE4958



- El segundo color es el azul, este cuenta con dos tonalidades distintas: una más clara #499BEE para los fondos de los formularios y otra más oscura para las letras #006AD4



- El tercer color es el amarillo #EEDF49:



- El cuarto color es el verde, el cual representa el carácter en parte militar de la aplicación #53C27D:



Diseño del prototipo

En este apartado de la documentación expondré el prototipo, realizado en Just In Mind y con él, el diseño de la interfaz gráfica:

Login:

Esta imagen muestra el diseño del login de la aplicación Predicción de incendios Ume:

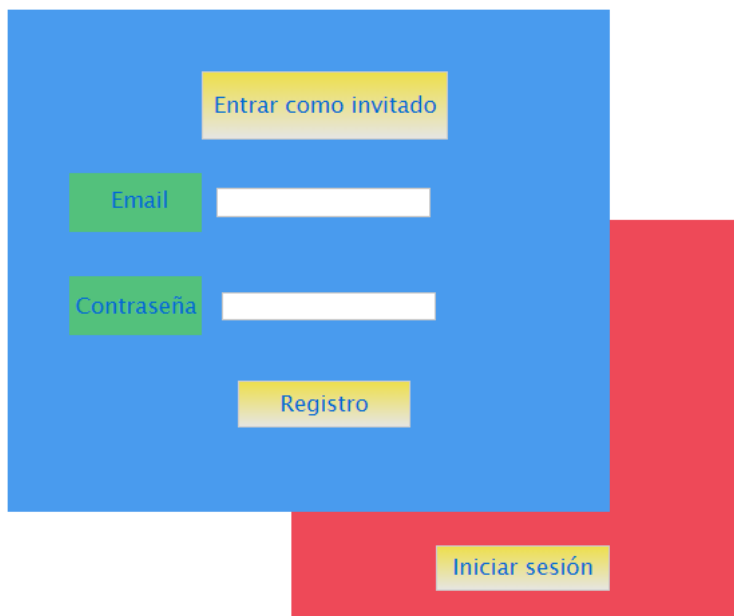
El prototipo de la interfaz de login se presenta sobre un fondo dividido en dos secciones: una superior roja y una inferior azul. En la sección roja, se encuentran los elementos de login: un botón amarillo con el texto 'Entrar como invitado' en la parte superior; a continuación, dos campos de entrada blancos, el primero precedido por una etiqueta verde con el texto 'Email' y el segundo por una etiqueta verde con el texto 'Contraseña'; y finalmente, un botón amarillo con el texto 'Iniciar sesión' en la parte inferior. En la sección azul, se encuentra un botón amarillo con el texto 'Registro'.

En el login se dispone de tres opciones:

- Entrar como invitado, en el caso de entrar como invitado no se dispondrá de acceso a las partes de modificación de datos de la aplicación
- Iniciar sesión, con un usuario o contraseña
- Registrarse

Registro:

Esta imagen muestra el diseño del registro, para registrarse solo se necesita escoger un correo y una contraseña, desde esta ventana además se puede acceder como invitado y volver a la parte del login:



The image shows a registration and login form. It has a blue background for the registration section and a red background for the login section. The registration section includes a yellow button labeled 'Entrar como invitado', two green labels 'Email' and 'Contraseña' next to white input fields, and a yellow button labeled 'Registro'. The login section has a yellow button labeled 'Iniciar sesión'.

Menú principal:

Este menú sirve para acceder a cada una de las partes de la aplicación, el menú es diferente dependiendo de si se ha realizado el login o se ha accedido como invitado

Menú en caso de haber realizado el login con usuario y contraseña:



The image shows the main menu for users who have logged in. It has a green background. At the top, it says 'Bienvenido a Predicción de incendios UME'. In the center is the coat of arms of the UME (Unidad Militar de Emergencias). Surrounding the coat of arms are several yellow buttons: 'Administración de usuarios', 'Acceder al historial de incendios', 'Modificar registros de incendios', 'Modificar datos meteorológicos', 'Consultar probabilidades de incendio', 'Modificar localidades', and 'Salir del programa'.

Menú cuando el usuario ha accedido como invitado, como podemos ver se impide el acceso a las partes de administración de datos de la aplicación:



Gestión de usuarios:

Esta ventana de la aplicación se usa para la eliminación y activación o desactivación de usuarios, tras seleccionar a un usuario en la tabla y clicar en alguno de los dos botones:



Gestión de localidades:

Esta es la ventana de gestión de localidades, desde ella se pueden añadir nuevas localidades, modificar las existentes o borrarlas:

Provincia	Localidad
Ciudad Real	Puertollano
Ciudad Real	Ciudad Real

[Volver atras](#)

[Añadir Localidad](#) [Modificar Localidad](#) [Eliminar localidad](#)

Para añadir localidades se usa esta ventana, la cual cuenta con una combobox que sirve para seleccionar la provincia donde se quiere situar la localidad, no puede haber dos localidades con el mismo nombre en una provincia, las temperaturas y humedades medias se usan en caso de no existir datos climatológicos en la fecha elegida para el cálculo de las probabilidades de un incendio, la temperatura no puede ser superior a 60 C° ni inferior a -60° y la humedad no puede ser inferior a 0% ni superior a 100%:

[Volver atras](#)

Provincia:

Nombre de la localidad:

Temperatura media primavera: Humedad media primavera:

Temperatura media verano: Humedad media verano:

Temperatura media otoño: Humedad media otoño:

Temperatura media invierno: Humedad media invierno:

[Añadir](#)

Para modificar las localidades se usa una ventana similar que sigue las mismas reglas excepto por el hecho de que no se puede modificar ni el nombre de la localidad ni su provincia:

Modificar localidad

[Volver atras](#)

Provincia:

Nombre de la localidad:

Temperatura media primavera: Humedad media primavera:

Temperatura media verano: Humedad media verano:

Temperatura media otoño: Humedad media otoño:

Temperatura media invierno: Humedad media invierno:

[Modificar](#)

Gestión de datos climatológicos:

Esta es la ventana de gestión de datos climatológicos, desde ella se pueden añadir nuevos datos, modificar los existentes o borrarlos:

Modificación de registros climatológicos

[Volver atras](#)

Provincia	Localidad	Temperatura media	Fecha de inicio	Fecha de finalización
Ciudad Real	Puertollano	20 C	11/10/23	20/10/23
Ciudad Real	Ciudad Real	30 C	10/8/21	25/8/21

[Modificar registro](#)[Añadir registro](#)[Eliminar registro](#)

Para añadir registros climatológicos se utiliza la siguiente ventana, la cual cuenta con dos combobox, el primero sirve para seleccionar la provincia y el segundo para seleccionar una localidad de esa provincia, las normas con respecto a la temperatura y la humedad son las mismas que en la creación de localidades y las fechas del registro climatológico no pueden coincidir ni en un solo día con las de otro registro en la misma localidad:



Añadir registro [Volver atras](#)

Provincia:

Localidad:

Temperatura media:

Grado de humedad:

Fecha de inicio:

Fecha de final:

[Añadir](#)

Para modificar los registros se utiliza una ventana muy similar a la de añadir pero en esta no se puede modificar ni la provincia ni la localidad:



Modificar registro [Volver atras](#)

Provincia:

Localidad:

Temperatura media:

Grado de humedad:

Fecha de inicio:

Fecha de final:

[Modificar](#)

Gestión de incendios:

Esta es la ventana de gestión de incendios, desde ella se pueden añadir nuevos incendios, modificar los existentes o borrarlos:

Modificación de registros de incendios

[Volver atras](#)

Incendios	Localidad	Fecha de inicio
Incendio N º 1	Puertollano	01/11/2020
Incendio N º 1	Ciudad Real	06/10/2023

[Modificar registro](#)[Añadir registro](#)[Eliminar registro](#)

El proceso de añadir incendios es el mismo que el de añadir datos climatológicos, con la diferencia de que en incendios es necesario especificar las hectáreas quemadas y que las fechas de dos incendios en una misma localidad pueden coincidir:

Añadir registro

[Volver atras](#)

Provincia:

Ciudad Real

Localidad:

Puertollano

Hectáreas quemadas:

Temperatura media:

Grado de humedad:

Fecha de inicio:

Fecha de extinción:

[Añadir](#)

Para modificar los incendios se utiliza una ventana muy similar a la de añadir pero en esta no se puede modificar ni la provincia ni la localidad:

Modificar registro

[Volver atras](#)

Provincia:

Ciudad Real

Localidad:

Puertollano

Hectáreas quemadas:

800

Temperatura media:

32 C

Grado de humedad:

5%

Fecha de inicio:

11/25/2021

Fecha de extinción:

11/30/2021

[Modificar](#)

Historial de incendios:

En esta ventana cualquier usuario puede consultar el historial de incendios, para facilitarle la búsqueda de un incendio en específico puede buscar por provincia, localidad y fecha:

Historial de registros de incendios

[Volver atras](#)

Incendios	Localidad	Fecha de inicio
Incendio N º 1	Puertollano	01/11/2020
Incendio N º 1	Ciudad Real	06/10/2023

Agrupar por:

Fecha:

Provincia:

Ciudad Real

Desde:

11/04/2021

Localidad:

Puertollano

Hasta:

11/04/2021

[Buscar](#)

Predicción de incendios:

Por último la ventana final del programa es la de predicción de incendios, en ella se puede consultar las probabilidades de que ocurra un incendio en una localidad:

Consulta de probabilidades de un incendio

Volver atras

Provincia: Ciudad Real

Localidad: Puertollano

Fecha: 11/24/2021

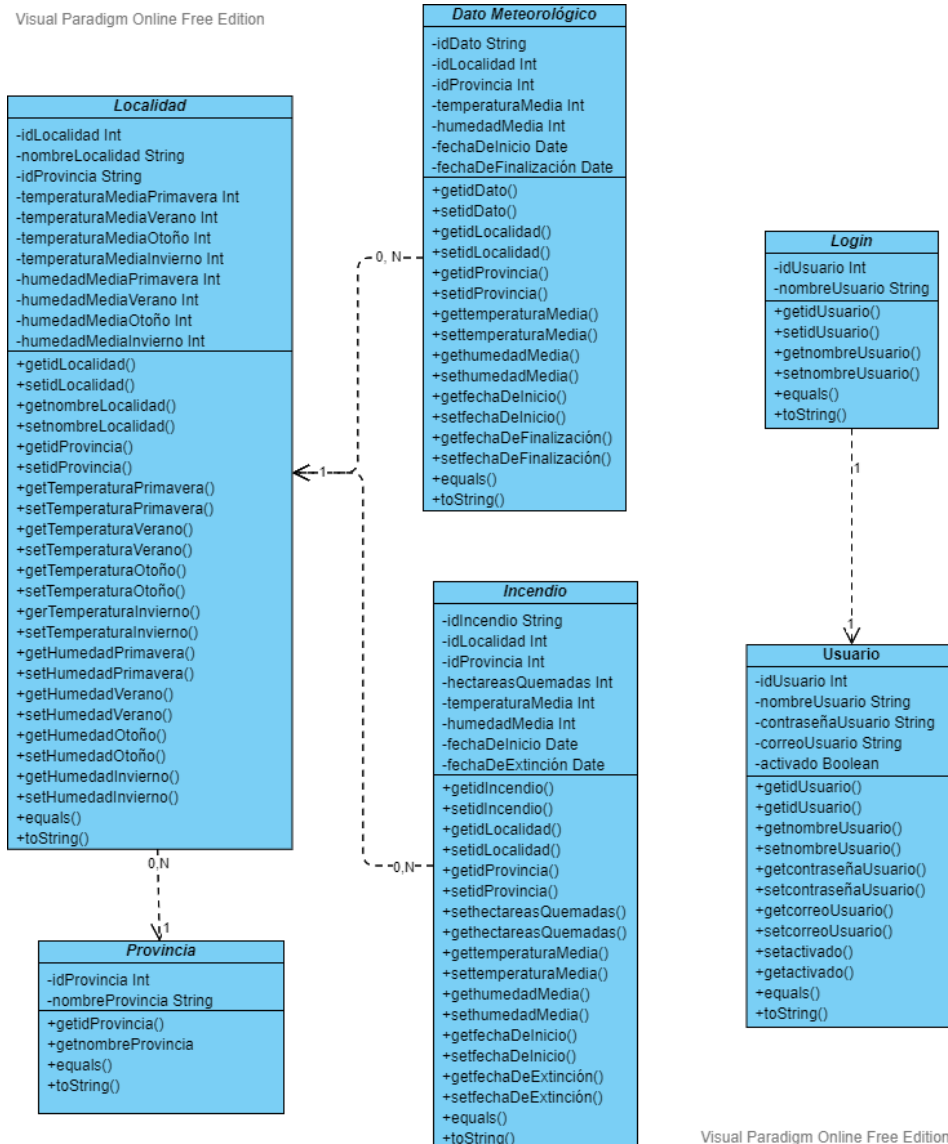
Riesgo de incendio: 60 % de riesgo

Calcular riesgo de incendio

Diagrama de clases

En esta parte de la documentación hablaré del diagrama de clases, mostraré una imagen y hablaré de cada clase

Imagen del diagrama de clases



El diagrama de clases cuenta con 6 clases principales, las cuales son:

- Clase Incendio:

La clase incendio recoge la información de cada incendio los cuales se guardan en la base de datos y son luego usados para calcular las probabilidades de un incendio

La clase incendio cuenta con los siguientes atributos:

- idIncendio

- idLocalidad
- idProvincia
- hectareasQuemadas
- temperaturaMedia
- humedadMedia
- fechaDelInicio
- fechaDeExtinción

y cuenta con los siguientes métodos:

- getIdIncendio()
- setIdIncendio()
- getIdLocalidad()
- setIdLocalidad()
- getIdProvincia()
- setIdProvincia()
- getHectareasQuemadas()
- setHectareasQuemadas()
- getTemperaturaMedia()
- setTemperaturaMedia()
- getHumedadMedia()
- setHumedadMedia()
- getFechaDelInicio()
- setFechaDelInicio()
- getFechaDeExtinción()
- setFechaDeExtinción()
- equals()
- toString()

- Clase usuario:

La clase usuario es usada para guardar en la base de datos la información de los usuarios y de esta se extrae la información para la clase login

La clase usuario cuenta con los siguientes atributos:

- idUsuario
- nombreUsuario
- contraseñaUsuario

- correoUsuario
- activado

y cuenta con los siguientes métodos

- getIdUsuario()
- setIdUsuario()
- getNombreUsuario()
- setNombreUsuario()
- getCorreoUsuario()
- setCorreoUsuario()
- getActivado()
- setActivado()
- equals()
- toString()

- Clase login:

La clase Login es utilizada una vez el login ha sido realizado, la clase guarda el nombre y el id del usuario y se utiliza en otros formularios para mostrar el nombre y comprobar si puede realizar operaciones en la base de datos

La clase login cuenta con los siguientes atributos:

- idUsuario
- nombreUsuario

y cuenta con los siguientes métodos

- getIdUsuario()
- setIdUsuario()
- getNombreUsuario()
- setNombreUsuario()
- equals()
- toString()

- Clase provincia

La clase provincia, se usa para guardar todos los nombres de las provincias de España, no se pueden crear ni modificar, pues todas vienen ya creadas en la propia base de datos, las provincias cuentan con localidades que sí pueden ser creadas y modificadas

La clase provincia cuenta con los siguientes atributos:

- idLocalidad
- nombreLocalidad
- idProvincia
- temperaturaMediaPrimavera
- temperaturaMediaVerano
- temperaturaMediaOtoño
- temperaturaMediaInvierno
- humedadMediaPrimavera
- humedadMediaVerano
- humedadMediaOtoño
- humedadMediaInvierno

y cuenta con los siguientes métodos

- getIdLocalidad()
- setIdLocalidad()
- getNombreLocalidad()
- setNombreLocalidad()
- getIdProvincia()
- setIdProvincia()
- getTemperaturaMediaPrimavera
- setTemperaturaMediaPrimavera
- getTemperaturaMediaVerano
- setTemperaturaMediaVerano
- getTemperaturaMediaOtoño
- setTemperaturaMediaOtoño
- getTemperaturaMediaInvierno
- setTemperaturaMediaInvierno
- getHumedadMediaPrimavera
- setHumedadMediaPrimavera
- getHumedadMediaVerano
- setHumedadMediaVerano

- getHumedadMediaOtoño
 - setHumedadMediaOtoño
 - getHumedadMediaInvierno
 - setHumedadMediaInvierno
 - equals()
 - toString()
- Clase localidad

La clase localidad representa los lugares en los que ocurren los incendios, que un incendio ocurra en una localidad no significa que sea en la ciudad o el pueblo en sí, si no en sus alrededores

La clase localidad cuenta con los siguientes atributos:

- idProvincia
- nombreProvincia

y cuenta con los siguientes métodos

- getIdProvincia()
- setIdProvincia()
- getNombreProvincia()
- setNombreProvincia()
- equals()
- toString()

- Clase dato Meteorológico

La última es Dato Meteorológico, la cual representa la temperatura y humedad medias que se esperan en cierta localidad durante un cierto periodo de tiempo, esto junto con el historial de incendios es lo que se usa para calcular las probabilidades de que ocurra un incendio

La clase incendio cuenta con los siguientes atributos:

- idDato
- idLocalidad
- idProvincia
- temperaturaMedia

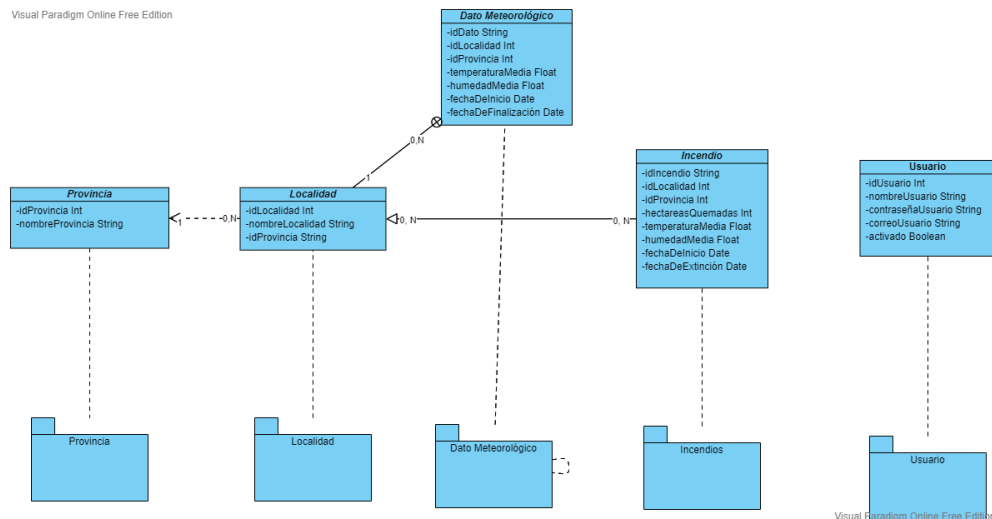
- humedadMedia
- fechaDelInicio
- fechaDeFinalización

y cuenta con los siguientes métodos:

- getidDato()
- setidDato()
- getidLocalidad()
- setidLocalidad()
- getidProvincia()
- setidProvincia()
- gettemperaturaMedia()
- settemperaturaMedia()
- gethumedadMedia()
- sethumedadMedia()
- getfechaDelInicio()
- setfechaDelInicio()
- getfechaDeFinalización()
- setfechaDeFinalización()
- equals()
- toString()

Persistencia de la información

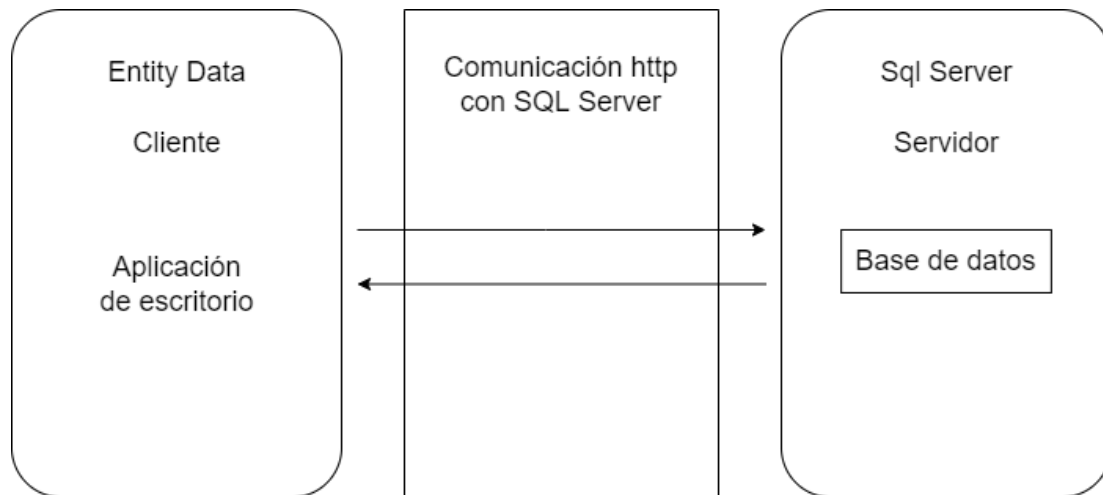
Esta imagen representa el diagrama de la persistencia de la información de mi proyecto predicción de incendios el cual está organizado por la información de las clases que se guarda y las distintas tablas representadas por ficheros



Arquitectura del sistema

En esta parte de la documentación hablaré de la arquitectura de Predicción de Incendios

Como se indicó anteriormente la base de datos usada en este proyecto es Microsoft SQL Server, instalado en la máquina local para simplificar el uso del programa en su versión de prueba y no tener así que montar un servidor, pese a ello este diseño de la arquitectura ha sido realizado teniendo en cuenta el uso de un servidor común externo, que use los mismos datos para todos los usuarios



Este diagrama representa la comunicación entre la base de datos y el programa, la cual es bastante sencilla pues no se usan métodos de autenticación preestablecidos, ni ningún sistema especial para guardar imágenes, pues toda la información se almacena en tablas normales de SQL Server

- Login: Se utilizan desde la aplicación una consulta SQL con la contraseña y el correo del usuario para realizar el login
- Registro: Se utiliza desde la aplicación una consulta SQL de inserción para añadir un nuevo usuario
- Tratamiento de datos: Para el tratamiento de datos se realizan consultas SQL desde la aplicación, ya sea para modificarlos, añadirlos o consultarlos

Análisis tecnológico

En esta parte de la documentación hablaré de las tecnologías de mi proyecto, de por qué las escogí, las compararé con otras tecnologías similares y explicaré las ventajas y desventajas que otorgan a un proyecto como el mío

Análisis de los posibles lenguajes:

- C#

C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, aunque C# destaca sobre todo en su parte de aplicaciones de escritorio sobre todo en el WPF o Windows Presentation Foundation, la cual supera con creces a la ya obsoleta Windows Forms, permitiendo crear formularios mucho más complejos que en este al no estar basado en píxeles, por lo que en conclusión la principal razón por la que he escogido C# es por sus buenas cualidades a la hora de crear programas de escritorio, y el hecho de poder usar herramientas como blend de Visual Studio para poder crear animaciones y en general una interfaz mucho más amigable que la que podría crear con los formularios de Java

- Java

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems, el cual cuenta con numerosas ventajas con respecto a otros lenguajes como el hecho de que su código compilado se ejecuta en la Java Virtual Machine sin que importe la arquitectura del dispositivo en el que está instalado ya sea un ordenador, un móvil, una tablet... la máquina virtual de java permite su ejecución sin apenas realizar cambios en el código de una versión a otra, el principal problema para mi de Java para poder realizar este proyecto son los formularios que ofrece Java, demasiado obsoletos y con pocas capacidades para ser personalizados, esa es la razón por la que aunque piense que Java es un gran lenguaje no fué escogido para este proyecto

- C++

C++ es un lenguaje de programación diseñado en 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos, pese a que C++ y C# son lenguajes prácticamente iguales al C# estar basado en C++, este no cuenta con las ventajas que ofrece el framework .NET, como pueden ser las conexiones a la base de datos que usan .NET, un ejemplo es el ado.net entity data model el cual va a ser usado en esta aplicación, por estas razones no he escogido C++ aunque este sea más rápido y esté mejor optimizado a la hora de ejecutarlo

Análisis de las posibles bases de datos:

- Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft del tipo SQL, la razón por la que he escogido esta base de datos es por la buena integración que tiene con C# siendo bastante fácil conectarla al programa ya sea con usuario y contraseña o a través del Windows Authentication lo que hace muy sencillo conectar un programa de C# con esta base de datos

- MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation, al igual que SQL server usa un lenguaje del tipo SQL, pero el hecho de que sea de Oracle y no de Microsoft, no permite aprovechar de la misma forma las ventajas de la conectividad que ofrece C# con el .NET que sí permite Sql server

- En conclusión:

La razón por la que he elegido WPF de C# es por la capacidad de poder realizar formularios más complejos al no estar basado en los controles de Windows y SQL Server en la base de datos por su buena integración con C# gracias al .NET de este

Implementación de la solución:

Elementos implementados:

- Login: el login de predicción de incendios consta de dos partes:
- La primera parte del login es el login en sí, donde un usuario puede identificarse, entrar como un invitado, también se dispone de un botón para ir a la segunda parte del login a través de una animación

Imagen del login tras iniciar sesión:



Diseño de la consulta a la base de datos para realizar el login:

```
public Boolean[] consultaUsuarioLogin(String Correo, String Contraseña)
{
    Boolean[] resultado = new Boolean[3];
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consultaCorreo = from busqueda in baseDeDatos.usuario
                             where busqueda.correoUsuario.Contains(Correo)
                             select new { busqueda.contrasenaUsuario, busqueda.activo };

        if (consultaCorreo.Count() != 0)
        {
            resultado[0] = true;

            if (Contraseña.Equals(consultaCorreo.FirstOrDefault().contrasenaUsuario))
            {
                resultado[1] = true;
                if (consultaCorreo.FirstOrDefault().activo.Equals("Activo"))
                {
                    resultado[2] = true;
                }
                else
                {
                    resultado[2] = false;
                }
            }
            else
            {
                resultado[1] = false;
                resultado[2] = false;
            }
        }
        else
        {
            resultado[0] = false;
            resultado[1] = false;
            resultado[2] = false;
        }
    }

    return resultado;
}
```

Esta consulta comprueba cada uno de los campos, el correo, la contraseña y el email por separados y si todos los campos son correctos se realiza el login en caso contrario dependiendo de qué campos fallen la aplicación avisará al usuario el por qué está fallando su login

- La segunda parte es el registro, un usuario podrá registrarse en la base de datos pero su cuenta estará desactivada, un administrador, es decir un usuario ya registrado y con la cuenta activada podrá activar su cuenta, el nombre de usuario está pensado para el uso del nombre real de este por lo que un nombre podrá repetirse. Ese no es el caso con el Email, el cual como identificador del usuario que es no podrá repetirse, además la ventana de registro cuenta con una clase para validar emails por lo que si el usuario no escribe un email real el programa informará del error, igual que si no se rellenan todos los campos

Esta imagen muestra la parte del registro tras registrar un usuario:

The image shows a user registration interface. At the top, there is a yellow button labeled "Entrar como invitado". Below it, there are three input fields with green labels: "Usuario" (containing "Usuario Nuevo"), "Email" (containing "nuevoUsuario@email.com"), and "Contraseña" (containing four dots). A yellow "Registrarse" button is positioned below the password field. To the right, a red button labeled "Login" is partially visible. In the foreground, a white dialog box with a close button (X) displays the message: "Cuenta creada, por favor espere a que una administrador la active". At the bottom of the dialog is an "Aceptar" button.

En esta imagen se muestran las consultas necesarias para realizar un registro:

```
public Boolean consultaUsuarioRegistro(String correo)
{
    Boolean resultado = false;
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consulta = from busqueda in baseDeDatos.usuario
                        where busqueda.correoUsuario.Contains(correo)
                        select new { nombreUuario = busqueda.nombreUsuario };

        if (consulta.Count() != 0)
        {
            resultado = true;
        }
    }

    return resultado;
}

public void insertarUsuario(usuario UsuarioAInsertar)
{
    prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB();
    baseDeDatos.usuario.Add(UsuarioAInsertar);
    baseDeDatos.SaveChanges();
}
```

Para poder registrarte la aplicación primero comprueba si el correo existe en de que el correo exista, la aplicación avisa al usuario y no se realiza el registro, en caso contrario se inserta el usuario, el cual se construye en la ventana de registro para después ser insertado

Esta imagen muestra el método para realizar el registro:

```
private void RegistroUsuario(Object sender, ExecutedRoutedEventArgs e)
{
    string email = this.txtEmailRegistro.Text.Trim();
    string usuario = this.txtUsuarioRegistro.Text.Trim();
    string contraseña = this.txtContraseñaRegistro.Password.Trim();
    if (email.Length == 0 | usuario.Length == 0 | contraseña.Length == 0)
    {
        MessageBox.Show("Rellene todos los campos para realizar el registro");
    }
    else
    {
        if (EmailEsValido(email))
        {
            if (this.consulta.consultaUsuarioRegistro(email))
            {
                MessageBox.Show("Email en uso");
            }
            else
            {
                usuario UsuarioAInsertar = new usuario();

                UsuarioAInsertar.activo = "Inactivo";
                UsuarioAInsertar.contrasenaUsuario = contraseña;
                UsuarioAInsertar.correoUsuario = email;
                UsuarioAInsertar.nombreUsuario = usuario;

                this.consulta.insertarUsuario(UsuarioAInsertar);

                MessageBox.Show("Cuenta creada, por favor espere a que una administrador la active");
            }
        }
        else
        {
            MessageBox.Show("El formato de su email no es válido");
        }
    }
}
```

- Menú principal:

La principal función de esta parte de Predicción de incendios es la de acceder al resto de las partes del programa, el menú principal puede variar dependiendo de si se ha realizado el login como usuario o como invitado, en caso de realizar el login como invitado no se tendrá acceso a las partes de administración de la aplicación, solo a los registros del historial de incendios y al cálculo de la probabilidad de incendio

Imagen del menú principal a través de un administrador:



La siguiente imagen muestra el código que evita que un usuario invitado acceda a las partes de administración siendo login un boolean que indica si es un usuario registrado o un invitado

```
ClaseLogin infoUsuario;
Boolean cargado = false;

public VentanaPrincipal(ClaseLogin InfoUsuario)
{
    this.infoUsuario = InfoUsuario;
    InitializeComponent();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    cargado = true;
    if (this.infoUsuario.login)
    {
        this.btnAdministracionUsuarios.Visibility = Visibility.Visible;
        this.btnModificarDatosMeteorológicos.Visibility = Visibility.Visible;
        this.btnModificarHistorialIncendios.Visibility = Visibility.Visible;
        this.btnModificarLocalidades.Visibility = Visibility.Visible;
    }
}
```

- Parte de gestión de usuarios:

En la parte de gestión de usuarios se podrá modificar el estado de las cuentas de otros usuarios, se pueden activar y desactivar y borrar, la tabla de usuarios no mostrará ni dejará modificar al usuario actual por razones obvias, no permite eliminar la cuenta a sí mismo, para activar o borrar un usuario solo hay que seleccionar clicando en la tabla al usuario al cual se desea modificar y después clicar en el botón correspondiente, en caso de no seleccionar ningún usuario, el programa avisará y no realizará ninguna acción y en el caso de borrar, el programa pedirá una confirmación

Imagen de la parte de gestión de usuarios con el usuario anteriormente creado:

Nombre Del Usuario	Correo Del Usuario	Contraseña Del Usuario	Activo
Usuario Nuevo	nuevoUsuario@email.com	1234	Activo

La siguiente imagen muestra la consulta para activar o desactivar un usuario, como podemos ver solo se pasa desde la ventana de

gestión de usuarios el correo del usuario al que se desea activar o desactivar, y después de buscarlo con una consulta este es activado o desactivado dependiendo de su estado anterior:

```
public void modificarUsuario(String correoUsuario)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        usuario UsuarioAModificar = baseDeDatos.usuario.First(x => x.correoUsuario.Equals(correoUsuario));
        String activo = UsuarioAModificar.activo;

        if (activo.Equals("Activo"))
        {
            UsuarioAModificar.activo = "Inactivo";
            baseDeDatos.SaveChanges();
        }
        else
        {
            UsuarioAModificar.activo = "Activo";
            baseDeDatos.SaveChanges();
        }
    }
}
```

Esta última imagen muestra la consulta usada para eliminar un usuario y como podemos ver es similar en funcionamiento a la de activar/desactivar usuario pues, esta recibe el correo del usuario y con este busca al usuario para después borrarlo

```
public void eliminarUsuario(String correoUsuario)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        usuario UsuarioAEliminar = baseDeDatos.usuario.First(x => x.correoUsuario.Equals(correoUsuario));
        baseDeDatos.usuario.Remove(UsuarioAEliminar);
        baseDeDatos.SaveChanges();
    }
}
```

- Gestión de localidades:

En esta ventana el usuario podrá añadir, modificar y borrar localidades, en caso de modificar y borrar se requiere que al igual en la gestión de usuarios se seleccione una localidad ya creada, además añadir añadir y crear disponen de sus propias ventanas

En esta imagen se muestra la ventana de gestión de localidades:



Esta imagen muestra las dos consultas realizadas para eliminar la localidad, la primera comprueba que la localidad no tenga ningún incendio, en caso de tener un incendio o más no se realiza el borrado y en caso contrario se realizaría el borrado a través de la segunda consulta de la localidad y sus datos climatológicos:

```
public Boolean consultaIncendiosEnLocalidad(int idLocalidad)
{
    Boolean resultado = true;
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consultaCorreo = from busqueda in baseDeDatos.Incendio
                             where busqueda.idLocalidad == idLocalidad
                             select new { busqueda.idIncendio };

        if (consultaCorreo.Count() != 0)
        {
            resultado = true;
        }
        else
        {
            resultado = false;
        }
    }
    return resultado;
}

public void eliminarLocalidad(int idLocalidad)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consultaDatosMeteo = from busqueda in baseDeDatos.datosMeteo
                                 where busqueda.idLocalidad == idLocalidad
                                 select new { busqueda.idDatos };

        if (consultaDatosMeteo.Count() != 0)
        {
            var idDatos = consultaDatosMeteo.ToList();
            foreach (var dato in idDatos)
            {
                int idDatos = dato.idDatos;
                datosMeteoEliminar = baseDeDatos.datosMeteo.First(x => x.idDatos == idDatos);
                baseDeDatos.datosMeteo.Remove(datosMeteoEliminar);
                baseDeDatos.SaveChanges();
            }
        }

        localidadAEliminar = baseDeDatos.localidad.First(x => x.idLocalidad == idLocalidad);
        baseDeDatos.localidad.Remove(localidadAEliminar);
        baseDeDatos.SaveChanges();
    }
}
```

La ventana de añadir localidades cuenta con una combo box de provincias, con todas las provincias de España, y en cuanto al nombre de la localidad, este puede repetirse siempre que no exista una con el mismo nombre en la misma provincia

Añadir Localidad

Volver atras

Provincia:

Ciudad Real

Nombre de la localidad:

Puertollano

Temperatura Media Primavera:

20

Humedad Media Primavera:

50

Temperatura Media Verano:

30

Humedad Media Verano:

20

Temperatura Media Otoño:

15

Humedad Media Otoño:

60

Temperatura Media Invierno:

5

Humedad Media Invierno:

80

Añadir

La siguiente imagen muestra el diseño de la consulta que carga las provincias:

```
public List<String> cargarProvincias()
{
    List<String> listaProvincias = new List<string>();
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consultaProvincia = from busqueda in baseDeDatos.provincia
                                select new { busqueda.nombreProvincia };
        var lista = consultaProvincia.ToList();
        foreach (var provincia in lista)
        {
            listaProvincias.Add(provincia.nombreProvincia);
        }
        return listaProvincias;
    }
}
```

```

public Boolean consultarLocalidad(String nombreLocalidad, int IdProvincia)
{
    Boolean resultado = false;
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        var consultaNombre = from busqueda in baseDeDatos.localidad
                             where busqueda.nombreLocalidad.Contains(nombreLocalidad) &&
                                   busqueda.idProvincia == IdProvincia
                             select new { busqueda.idLocalidad };

        var listaDeLocalidades = consultaNombre.ToList();
        if (listaDeLocalidades.Count != 0)
        {
            resultado = true;
        }
    }

    return resultado;
}

public void insertarLocalidad(localidad localidadInsertar)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        baseDeDatos.localidad.Add(localidadInsertar);
        baseDeDatos.SaveChanges();
    }
}

public void modificarLocalidad(localidad localidadModificar)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        baseDeDatos.localidad.Attach(localidadModificar);
        baseDeDatos.Entry(localidadModificar).State = EntityState.Modified;
        baseDeDatos.SaveChanges();
    }
}

```

```
private void Abadir(Objet sender, EventArgs e)
{
    if (this.cmbProvincia.SelectedIndex != -1)
    {
        String nombreProvincia = this.cmbProvincia.SelectedItem.ToString();
        String nombreLocalidad = this.txtNombreLocalidad.Text.ToString().Trim();
        String temperaturaMediayInvierno = this.txtTemperIniv.Text.ToString();
        String temperaturaMediayVerano = this.txtTemperVer.Text.ToString();
        String temperaturaMediayOtoño = this.txtTemperOto.Text.ToString();
        String temperaturaMediayInvierno = this.txtTemperInv.Text.ToString();

        String humedadMediayInvierno = this.txtHumarIniv.Text.ToString();
        String humedadMediayVerano = this.txtHumarVer.Text.ToString();
        String humedadMediayOtoño = this.txtHumarOto.Text.ToString();
        String humedadMediayInvierno = this.txtHumarInv.Text.ToString();

        if (Convert.ToInt32(temperaturaMediayVerano.Trim().Length) == 0 || temperaturaMediayVerano.Trim().Length == 0 || temperaturaMediayOtoño.Trim().Length == 0 || temperaturaMediayInvierno.Trim().Length == 0 || humedadMediayInvierno.Trim().Length == 0 || humedadMediayOtoño.Trim().Length == 0 || humedadMediayVerano.Trim().Length == 0)
        {
            MessageBox.Show("Por favor rellene todos los campos");
        }
        else
        {
            int idProvincia = consultas.buscarIdProvincia(nombreProvincia);
            if (consultas.consultarLocalidad(nombreLocalidad, idProvincia))
            {
                MessageBox.Show("Ya existe una localidad con ese nombre en la provincia seleccionada");
            }
            else
            {
                if (int.TryParse(temperaturaMediayInvierno, out _) && int.TryParse(temperaturaMediayVerano, out _) && int.TryParse(temperaturaMediayOtoño, out _) && int.TryParse(humedadMediayInvierno, out _) && int.TryParse(humedadMediayVerano, out _) && int.TryParse(humedadMediayOtoño, out _) && int.TryParse(humedadMediayInvierno, out _))
                {
                    int tempVerIniv = int.Parse(temperaturaMediayInvierno);
                    int tempVerIniv = int.Parse(temperaturaMediayVerano);
                    int tempOtoIniv = int.Parse(temperaturaMediayOtoño);
                    int tempInivVer = int.Parse(temperaturaMediayInvierno);
                    int humVerIniv = int.Parse(humedadMediayInvierno);
                    int humVerIniv = int.Parse(humedadMediayVerano);
                    int humOtoIniv = int.Parse(humedadMediayOtoño);
                    int humInivVer = int.Parse(humedadMediayInvierno);

                    if ((tempVerIniv <= 60 && tempVerIniv >= -60) && (tempVerIniv <= 60 && tempVerIniv >= -60) && (tempOtoIniv <= 60 && tempOtoIniv >= -60) && (tempInivVer <= 60 && tempInivVer >= -60) && (humVerIniv <= 100 && humVerIniv >= 0) && (humVerIniv <= 100 && humVerIniv >= 0) && (humOtoIniv <= 100 && humOtoIniv >= 0) && (humInivVer <= 100 && humInivVer >= 0))
                    {
                        Localidad localidAdir = new Localidad();
                        Localidad localidProv = idProvincia;
                        Localidad localidAdir.nombreLocalidad = nombreLocalidad;
                        Localidad localidAdir.temperaturaMediayInvierno = int.Parse(temperaturaMediayInvierno);
                        Localidad localidAdir.temperaturaMediayVerano = int.Parse(temperaturaMediayVerano);
                        Localidad localidAdir.temperaturaMediayOtoño = int.Parse(temperaturaMediayOtoño);
                        Localidad localidAdir.temperaturaMediayInvierno = int.Parse(temperaturaMediayInvierno);
                        Localidad localidAdir.humedadMediayInvierno = int.Parse(humedadMediayInvierno);
                        Localidad localidAdir.humedadMediayVerano = int.Parse(humedadMediayVerano);
                        Localidad localidAdir.humedadMediayOtoño = int.Parse(humedadMediayOtoño);
                        Localidad localidAdir.humedadMediayInvierno = int.Parse(humedadMediayInvierno);
                        consultas.InsertarLocalidad(localidAdir);
                        MessageBox.Show("Localidad insertada con éxito");
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("La temperatura no puede ser ni superior a 60 ni inferior a - 60 y la humedad no puede ser negativa ni superior a 100");
                    }
                }
                else
                {
                    MessageBox.Show("Por favor no introduzca letras ni números decimales, la temperatura y la humedad se registran usando enteros");
                }
            }
        }
    }
    else
    {
        MessageBox.Show("Por favor seleccione una provincia");
    }
}
}
```


Por último en cuanto a la ventana de modificar localidades, esta es muy similar a la de añadir mostrando toda la información de la localidad, pero no se puede cambiar ni la provincia ni el nombre de esta, solo las temperaturas y las humedades, además la temperatura no podrá ser ni superior a 60 ni inferior a -60 y la humedad ni superior a 100 ni inferior a 0, esta norma se cumple en todo el programa:

Modificar Localidad Volver atras

Provincia:

Nombre de la localidad:

Temperatura Media Primavera:	<input type="text" value="15"/>	Humedad Media Primavera:	<input type="text" value="20"/>
Temperatura Media Verano:	<input type="text" value="30"/>	Humedad Media Verano:	<input type="text" value="10"/>
Temperatura Media Otoño:	<input type="text" value="10"/>	Humedad Media Otoño:	<input type="text" value="30"/>
Temperatura Media Invierno:	<input type="text" value="0"/>	Humedad Media Invierno:	<input type="text" value="30"/>

Modificar

Esta imagen muestra la consulta usada para modificar una localidad, como podemos ver la consulta recibe la localidad modificada, busca la original y la modifica para luego guardar los cambios:

```
public void modificarLocalidad(localidad localidadModificada)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        localidad localidadOriginal = baseDeDatos.localidad.First(x => x.idLocalidad == localidadModificada.idLocalidad);
        localidadOriginal.temperaturaMediaPrimavera = localidadModificada.temperaturaMediaPrimavera;
        localidadOriginal.temperaturaMediaVerano = localidadModificada.temperaturaMediaVerano;
        localidadOriginal.temperaturaMediaOtoño = localidadModificada.temperaturaMediaOtoño;
        localidadOriginal.temperaturaMediaInvierno = localidadModificada.temperaturaMediaInvierno;
        localidadOriginal.humedadMediaPrimavera = localidadModificada.humedadMediaPrimavera;
        localidadOriginal.humedadMediaVerano = localidadModificada.humedadMediaVerano;
        localidadOriginal.humedadMediaOtoño = localidadModificada.humedadMediaOtoño;
        localidadOriginal.humedadMediaInvierno = localidadModificada.humedadMediaInvierno;
        baseDeDatos.SaveChanges();
    }
}
```

- Gestión de datos climatológicos:

En esta ventana el usuario podrá añadir, modificar y borrar datos climatológicos, siguiendo las mismas normas que en la gestión de localidades

Nombre de la Provincia	Nombre de la Localidad	Temperatura media	Fecha de inicio	Fecha de finalización
Alcalá del Júcar	Albacete	21	06/12/2021	13/12/2021
Puertollano	Ciudad Real	30	09/12/2021	26/12/2021
Palos De Almeria	Almería	50	10/12/2021	30/12/2021
Alcalá del Júcar	Albacete	20	14/12/2021	19/12/2021

Imagen del código de la consulta usada para eliminar los registros climatológico, la cual usa el id del registro para borrarlo:

```
public void eliminarDato(int IdDato)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        datoMeteorologico DatoAEliminar = baseDeDatos.datoMeteorologico.First(x => x.idDato == IdDato);
        baseDeDatos.datoMeteorologico.Remove(DatoAEliminar);
        baseDeDatos.SaveChanges();
    }
}
```

La ventana de añadir registros climatológicos cuenta con una combo box de provincias, el cual cuando su valor es cambiado carga un combobox de localidades con las localidades de esta provincia, en cuanto a las fechas no podrán coincidir con las de otro registro en la misma localidad



Este es el código de las consultas para añadir registros, la primera se encarga de buscar si existe otro registro en la misma localidad que coincida con la fecha del nuevo registro, si existe el usuario es insertado y el registro no se inserta, y la segunda se encarga de añadirlo:

```
public Boolean buscarDataConMismaFecha(int IdLocalidad, DateTime FechaInicio, DateTime FechaFinalizacion)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        Boolean resultado = true;
        var consultaData = from busqueda in baseDeDatos.datoMeteoroologico
                           where busqueda.idLocalidad == IdLocalidad
                              && ((busqueda.fechaDeInicio <= FechaInicio && busqueda.fechaDeFinalizacion >= FechaFinalizacion)
                                  || (busqueda.fechaDeInicio >= FechaInicio && busqueda.fechaDeFinalizacion <= FechaFinalizacion)
                                  || (busqueda.fechaDeInicio <= FechaInicio && busqueda.fechaDeFinalizacion >= FechaInicio)
                                  || (busqueda.fechaDeInicio <= FechaFinalizacion && busqueda.fechaDeFinalizacion >= FechaFinalizacion)
                                  || (busqueda.fechaDeInicio >= FechaFinalizacion && busqueda.fechaDeFinalizacion <= FechaFinalizacion))
                           select new { busqueda.idData };

        if (consultaData.Count() != 0)
        {
            resultado = true;
        }
        else
        {
            resultado = false;
        }

        return resultado;
    }
}

public void insertarData(datoMeteoroologico datoInsertar)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        baseDeDatos.datoMeteoroologico.Add(datoInsertar);
        baseDeDatos.SaveChanges();
    }
}
```

Por último en cuanto la ventana de modificar, permitirá modificar todos los campos menos la localidad y la provincia



Las consultas para modificar el dato climatológico son similares a las de añadirlo, pues a la hora de comprobar las fechas es igual a este solo que comprobando que el id del dato no coincida con el dato que está siendo modificado:

```
public Boolean BuscarDatoConMismaFechaModificar(int IdDato, int IdLocalidad, DateTime FechaInicio, DateTime FechaFinalizacion)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        Boolean resultado = true;
        var consultaDato = from busqueda in baseDeDatos.datoMeteorologico
                           where busqueda.IdLocalidad == IdLocalidad
                              && ((busqueda.fechaDeInicio <= FechaInicio && busqueda.fechaDeFinalizacion >= FechaFinalizacion)
                              || (busqueda.fechaDeInicio >= FechaInicio && busqueda.fechaDeFinalizacion <= FechaFinalizacion)
                              || (busqueda.fechaDeInicio <= FechaInicio && busqueda.fechaDeFinalizacion >= FechaFinalizacion)
                              || (busqueda.fechaDeInicio <= FechaFinalizacion && busqueda.fechaDeFinalizacion >= FechaFinalizacion)
                              && busqueda.idDato != IdDato
                           select new { busqueda.idDato };

        if (consultaDato.Count() != 0)
        {
            resultado = true;
        }
        else
        {
            resultado = false;
        }

        return resultado;
    }
}

public void modificarDato(datoMeteorologico datoModificado)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        datoMeteorologico datoOriginal = baseDeDatos.datoMeteorologico.First(x => x.idDato == datoModificado.idDato);
        datoOriginal.temperaturaMedia = datoModificado.temperaturaMedia;
        datoOriginal.humedadMedia = datoModificado.humedadMedia;
        datoOriginal.fechaDeInicio = datoModificado.fechaDeInicio;
        datoOriginal.fechaDeFinalizacion = datoModificado.fechaDeFinalizacion;
        baseDeDatos.SaveChanges();
    }
}
```

- Gestión de datos Incendios:

En esta ventana el usuario podrá añadir, modificar y borrar incendios, siguiendo las mismas normas que en la gestión de localidades

Registros de incendios

[Volver atras](#)

Provincia	Localidad	Temperatura Media	Humedad Media	Hectareas	Fecha de inicio	Fecha de extinción
Denia	Alicante	17	22	2000	18/06/2012	18/06/2012
Alcalá del Júcar	Albacete	17	22	2000	18/06/2012	18/06/2012
Palos De Almeria	Almería	17	22	2000	18/06/2012	18/06/2012
Puertollano	Ciudad Real	17	22	2000	18/06/2012	18/06/2012
Níjar	Almería	50	80	1000	02/12/2021	18/12/2021

[Añadir Incendio](#)[Modificar Incendio](#)[Eliminar Incendio](#)

El código de la consulta usada para eliminar incendios es muy similar al de las otras consultas, simplemente eliminando el incendio cuyo id coincida con el del incendio seleccionado:

```
public void eliminarIncendio(int IdIncendio)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        Incendio IncendioAEliminar = baseDeDatos.Incendio.First(x => x.idIncendio == IdIncendio);
        baseDeDatos.Incendio.Remove(IncendioAEliminar);
        baseDeDatos.SaveChanges();
    }
}
```

La ventana de añadir registros de incendios cuenta con una combo box de provincias, el cual cuando su valor es cambiado carga un combobox de localidades con las localidades de esta provincia, en cuanto a las fechas se considera que han podido coexistir dos incendios en la misma localidad coincidiendo las fechas



The screenshot shows a web form titled "Añadir incendio" on a red background. At the top right is a "Volver atras" button. The form contains the following fields: "Provincia:" with a dropdown menu showing "Ciudad Real"; "Localidad:" with a dropdown menu showing "Puertollano"; "Hectáreas quemadas:" with a text input containing "1000"; "Temperatura media:" with a text input containing "40"; "HumedadMedia:" with a text input containing "70"; "Fecha de inicio:" with a date picker showing "02/12/2021"; and "Fecha de extinción:" with a date picker showing "18/12/2021". At the bottom center is an "Añadir" button.

Por último en cuanto la ventana de modificar, permitirá modificar todos los campos menos la localidad y la provincia



The screenshot shows a web form titled "Modificar Incendio" on a red background. At the top right is a "Volver atras" button. The form contains the following fields: "Provincia:" with a text input containing "Ciudad Real"; "Localidad:" with a text input containing "Puertollano"; "Hectáreas quemadas:" with a text input containing "1000"; "Temperatura media:" with a text input containing "40"; "HumedadMedia:" with a text input containing "20"; "Fecha de inicio:" with a date picker showing "03/12/2021"; and "Fecha de extinción:" with a date picker showing "18/12/2021". At the bottom center is a "Modificar" button.

Estas son las consultas usadas para añadir y modificar un incendio respectivamente:

```
public void insertarIncendio(Incendio incendioInsertar)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        baseDeDatos.Incendio.Add(incendioInsertar);
        baseDeDatos.SaveChanges();
    }
}

public void modificarIncendio(Incendio incendioModificado)
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        Incendio incendioOriginal = baseDeDatos.Incendio.First(x => x.idIncendio == incendioModificado.idIncendio);
        incendioOriginal.temperaturaMedia = incendioModificado.temperaturaMedia;
        incendioOriginal.humedadMedia = incendioModificado.humedadMedia;
        incendioOriginal.hectareasQuemadas = incendioModificado.hectareasQuemadas;
        incendioOriginal.fechaDeInicio = incendioModificado.fechaDeInicio;
        incendioOriginal.fechaDeExtinción = incendioModificado.fechaDeExtinción;
        baseDeDatos.SaveChanges();
    }
}
```

- Historial de incendios:

Esta al igual que la ventana del cálculo de las probabilidades de incendio, son las dos únicas ventanas a las que pueden acceder los usuarios invitados, en esta ventana se cargarán los incendios en una tabla central, estos se pueden ordenar y cargar dependiendo de la provincia, localidad y fecha, no es necesario rellenar todos los campos, pues se realiza una consulta por partes dependiendo de qué campos contengan información, además existe un botón para borrar la información de estos campos y cargar la tabla original

Historial de registros de incendios

Volver atras

Provincia	Localidad	Temperatura Media	Humedad Media	Hectareas	Fecha de inicio	Fecha de extinción
Denia	Alicante	17	22	2000	18/06/2012	18/06/2012
Alcalá del Júcar	Albacete	17	22	2000	18/06/2012	18/06/2012
Palos De Almeria	Almería	17	22	2000	18/06/2012	18/06/2012
Puertollano	Ciudad Real	17	22	2000	18/06/2012	18/06/2012
Níjar	Almería	50	80	1000	02/12/2021	18/12/2021

Provincia

Desde

Seleccione una fecha

15

Buscar

Localidad

Hasta

Seleccione una fecha

15

Limpiar

Código usado para consultar incendios en el historial, para cargar los incendios realiza una consulta que busque todos los incendios y dependiendo de qué campos tengan información realiza subconsultas sobre la consulta original:

```
private void Buscar(object sender, EventArgs e)
{
    AgrIncendios.ItemsSource = null;
    using (prediccion_incendioUtilitarios = new prediccion_incendioUtilitarios())
    {
        var incendios = from búsqueda in BaseDatos.Incendio
                        select new { búsqueda.idLocalidad, búsqueda.idIncendio, búsqueda.fechaDefinición, búsqueda.fechaExtinción };

        if (comboBoxLocalidades.Text.Equals(""))
        {
            string Provincia = comboBoxProvincias.Text;
            int idProvincia = consultar.BuscarIdProvincia(Provincia);
            string Localidad = comboBoxLocalidades.Text;
            int idLocalidad = consultar.BuscarIdLocalidad(idProvincia, Localidad);
            incendios = incendios.Where(x => x.idLocalidad == idLocalidad);
        }
        if (dateTimeDesde.Text.Equals(""))
        {
            DateTime desde = Convert.ToDateTime(dateTimeDesde.SelectedData.Value.Date.ToShortDateString());
            incendios = incendios.Where(x => x.fechaDefinición >= desde);
        }
        if (dateTimeHasta.Text.Equals(""))
        {
            DateTime hasta = Convert.ToDateTime(dateTimeHasta.SelectedData.Value.Date.ToShortDateString());
            incendios = incendios.Where(x => x.fechaDefinición <= hasta);
        }

        this.listaOGIncendios.Clear();
        if (comboBoxProvincias.Text.Equals("")) && comboBoxLocalidades.Text.Equals("")
        {
            string Provincia = comboBoxProvincias.Text;
            foreach (var incendio in incendios)
            {
                int idLocalidad = incendio.idLocalidad;
                string[] nombres = consultar.BuscarNombresLocalidadProvincia(idLocalidad);
                IncendioUtilitarios incendioNuevo = new IncendioUtilitarios(nombres[0], nombres[1], incendio.fechaDefinición.ToString().Substring(0, 10), incendio.fechaExtinción.ToString().Substring(0, 10));
                if (incendioNuevo.nombreProvincia == Provincia)
                {
                    this.listaOGIncendios.Add(incendioNuevo);
                }
            }
        }
        else
        {
            foreach (var incendio in incendios)
            {
                int idLocalidad = incendio.idLocalidad;
                string[] nombres = consultar.BuscarNombresLocalidadProvincia(idLocalidad);
                IncendioUtilitarios incendioNuevo = new IncendioUtilitarios(nombres[0], nombres[1], incendio.fechaDefinición.ToString().Substring(0, 10), incendio.fechaExtinción.ToString().Substring(0, 10));
                this.listaOGIncendios.Add(incendioNuevo);
            }
        }

        this.AgrIncendios.ItemsSource = this.listaOGIncendios;
        AgrIncendios.Columns[0].Header = "Nombre de la Provincia";
        AgrIncendios.Columns[1].Header = "Nombre de la Localidad";
        AgrIncendios.Columns[2].Header = "Fecha de inicio";
        AgrIncendios.Columns[3].Header = "Fecha de extinción";
    }
}
```

- Cálculo de la probabilidad de incendio

Esta es la última ventana del programa en ella se puede calcular las probabilidades de un incendio en una localidad durante un día específico, el programa cuenta con una compleja consulta que recoge la información de otros incendios en esa localidad durante la misma época del año e información climatológica para calcular las probabilidades de incendio, pudiendo ser: baja, media, alta y muy alta

Ejemplo de cálculo de la probabilidad de incendio:

Consultar probabilidades de un incendio

Volver atras

Provincia:

Ciudad Real

Localidad:

Puertollano

Fecha:

17/12/2021

19

Riesgo de incendio

Medio

Calcular riesgo de incendio

Esta es la consulta realizada para calcular las probabilidades de un incendio:

La primera parte de la consulta identifica cuál es la estación de la fecha en la que se quiere calcular la probabilidad:

```
{
    using (prediccion_incendiosEntitiesDB baseDeDatos = new prediccion_incendiosEntitiesDB())
    {
        int mes = fecha.Month;
        DateTime mesInferior = fecha;
        DateTime mesSuperior = fecha;
        int estacion = 0;
        if (mes >= 1 && mes <= 3)
        {
            if (mes == 1)
            {
                mesSuperior.AddMonths(2);
            }
            else if (mes == 2)
            {
                mesInferior.AddMonths(-1);
                mesSuperior.AddMonths(1);
            }
            else if (mes == 3)
            {
                mesInferior.AddMonths(-2);
            }

            estacion = 4;
        }
        else if (mes >= 4 && mes <= 6)
        {
            if (mes == 4)
            {
                mesSuperior.AddMonths(2);
            }
            else if (mes == 5)
            {
                mesInferior.AddMonths(-1);
                mesSuperior.AddMonths(1);
            }
            else if (mes == 6)
            {
                mesInferior.AddMonths(-2);
            }
            estacion = 4;
        }
        else if (mes >= 7 && mes <= 9)
        {
            if (mes == 7)
            {
                mesSuperior.AddMonths(2);
            }
            else if (mes == 8)
            {
                mesInferior.AddMonths(-1);
                mesSuperior.AddMonths(1);
            }
            else if (mes == 9)
            {
                mesInferior.AddMonths(-2);
            }
            estacion = 4;
        }
        else if (mes >= 10 && mes <= 12)
        {
            if (mes == 10)
            {
                mesSuperior.AddMonths(2);
            }
            else if (mes == 11)
            {
                mesInferior.AddMonths(-1);
                mesSuperior.AddMonths(1);
            }
            else if (mes == 12)
            {
                mesInferior.AddMonths(-2);
            }
            estacion = 4;
        }
    }
}
```

La segunda parte de la consulta obtiene la cantidad de incendios ocurridos en la localidad en la misma estación que la fecha dada y después obtiene la temperatura y la humedad media que se espera en esas fechas, en caso de no existir un dato climatológico con tal información se obtiene de las medias de la localidad:

```
String probabilidad = "";
int incendios = 0;
int temperatura = 0;
int humedad = 0;

var consultaIncendio = from busqueda in baseDatos.Incendio
    where busqueda.idLocalidad == IdLocalidad
    && (busqueda.fechaDeInicio.Value.Month <= mesInferior.Month && busqueda.fechaDeInicio.Value.Month >= mesSuperior.Month)
    select new { busqueda.idIncendio };

if (consultaIncendio.Count() != 0)
{
    incendios = consultaIncendio.Count();
}

var consultaData = from busqueda in baseDatos.datoMeteorologico
    where busqueda.idLocalidad == IdLocalidad
    && (busqueda.fechaDeInicio <= fecha && busqueda.fechaDeFinalizacion >= fecha)
    select new { busqueda.temperaturaMedia, busqueda.humedadMedia };

if (consultaData.Count() != 0)
{
    temperatura = consultaData.First().temperaturaMedia.Value;
    humedad = consultaData.First().humedadMedia.Value;
}
else
{
    var consultaLocalidad = from busqueda in baseDatos.localidad
        where busqueda.idLocalidad == IdLocalidad
        select new { busqueda.temperaturaMediaPrimavera, busqueda.temperaturaMediaVerano, busqueda.temperaturaMediaOtoño, busqueda.temperaturaMediaInvierno,
            busqueda.humedadMediaPrimavera, busqueda.humedadMediaVerano, busqueda.humedadMediaOtoño, busqueda.humedadMediaInvierno };

    switch (estacion)
    {
        case 1:
            temperatura = consultaLocalidad.First().temperaturaMediaPrimavera.Value;
            humedad = consultaLocalidad.First().humedadMediaPrimavera.Value;
            break;

        case 2:
            temperatura = consultaLocalidad.First().temperaturaMediaVerano.Value;
            humedad = consultaLocalidad.First().humedadMediaVerano.Value;
            break;

        case 3:
            temperatura = consultaLocalidad.First().temperaturaMediaOtoño.Value;
            humedad = consultaLocalidad.First().humedadMediaOtoño.Value;
            break;

        case 4:
            temperatura = consultaLocalidad.First().temperaturaMediaInvierno.Value;
            humedad = consultaLocalidad.First().humedadMediaInvierno.Value;
            break;
    }
}
```

Por último la tercera parte realiza los cálculos y dependiendo del resultado obtenido devuelve un valor, estos cálculos son realizados dependiendo de la temperatura, la humedad y el número de incendios: si la temperatura es muy elevada el número que indica la probabilidad aumenta, y si es baja este disminuye, ocurre lo mismo con la humedad, si esta es muy baja las posibilidades aumentan y si es alta disminuyen, en el caso de los incendios sirven para indicar si la zona es propensa a incendios, en caso de serlo, también aumentan las probabilidades de que ocurra otro:

```
int probabilidadInt = 0;

if (incendios <= 1)
{
    probabilidadInt = probabilidadInt + 1;
}
else if (incendios >= 2 && incendios <= 5)
{
    probabilidadInt = probabilidadInt + 2;
}
else if (incendios > 5)
{
    probabilidadInt = probabilidadInt + 3;
}

if (temperatura <= 0 && temperatura >= -20)
{
    probabilidadInt = probabilidadInt - 1;
}
else if (temperatura < -20)
{
    probabilidadInt = probabilidadInt - 2;
}
else if (temperatura >= 15 && temperatura <= 30)
{
    probabilidadInt = probabilidadInt + 1;
}
else if (temperatura > 30)
{
    probabilidadInt = probabilidadInt + 2;
}

if (humedad >= 0 && humedad <= 20)
{
    probabilidadInt = probabilidadInt + 2;
}
else if (humedad > 20 && humedad <= 40)
{
    probabilidadInt = probabilidadInt + 1;
}
else if (humedad >= 60 && humedad <= 80)
{
    probabilidadInt = probabilidadInt - 1;
}
else if (humedad > 80)
{
    probabilidadInt = probabilidadInt - 2;
}

if (probabilidadInt >= 7)
{
    probabilidad = ("Muy alto");
}
else if (probabilidadInt <= 6 && probabilidadInt >= 4)
{
    probabilidad = ("Alto");
}
else if (probabilidadInt <= 3 && probabilidadInt >= 1)
{
    probabilidad = ("Medio");
}
else if (probabilidadInt <= 0)
{
    probabilidad = ("Bajo");
}
return probabilidad;
}
```

Pruebas realizadas en la aplicación:

- Primera prueba: registro y login

Esta primera prueba comprobará el correcto funcionamiento del login y el registro en la aplicación

Registro:

La primera parte de la prueba será un registro con una cuenta que ya existe

The image shows a user interface for a login and registration system. It consists of two main panels: a blue one for login and a red one for registration. The blue panel has a yellow button at the top labeled 'Entrar como invitado'. Below it are three green input fields: 'Usuario' (containing 'Jorge'), 'Email' (containing 'jorge@hotmail.com'), and 'Contraseña' (containing four dots). A yellow 'Registrarse' button is at the bottom of the blue panel. The red panel has a yellow 'Login' button. An error message box is overlaid on the right side of the blue panel, displaying 'Email en uso' with a close button (X) and an 'Aceptar' button.

Como podemos ver el programa nos avisa de que el email ya está en uso

Ahora en la segunda parte de la prueba nos registramos con una cuenta que no existe:

Entrar como invitado

Usuario Jorge

Email jorge@email.com

Contraseña ••••

Registrarse

Cuenta creada, por favor espere a que una administrador la active

Aceptar

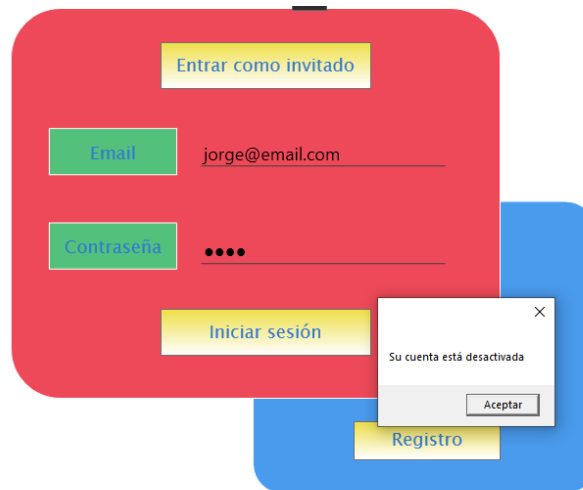
Como podemos ver el programa nos informa de que la cuenta ha sido creada pero no está activada, si realizamos una consulta la podemos ver en la base de datos:

```
select * from usuario where correoUsuario = 'jorge@email.com';
```

Results				
idUsuario	nombreUsuario	correoUsuario	contrasenaUsuario	activo
15	Jorge	jorge@email.com	1234	Inactivo

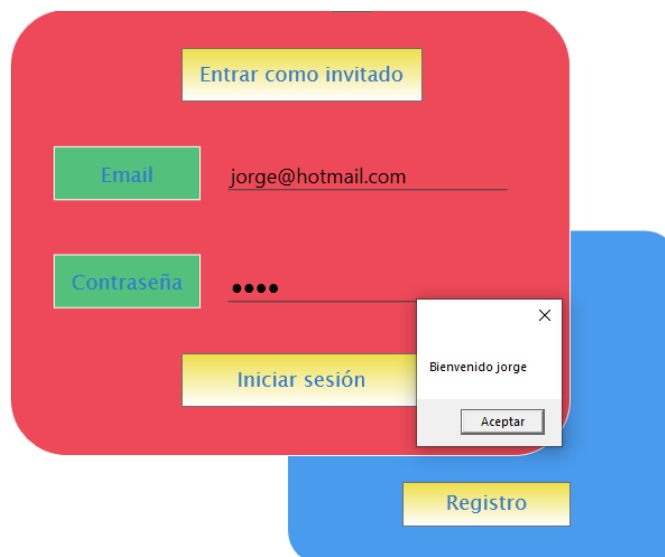
Login:

La primera prueba que realizaré es intentar iniciar sesión con la cuenta creada en las pruebas anteriores



Como podemos ver el programa nos avisa de que la cuenta está desactivada

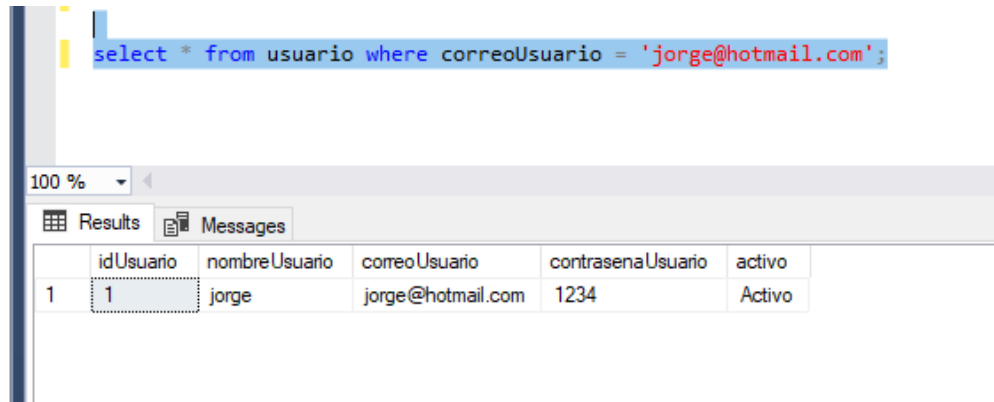
Por último como segunda prueba realizaré el login con una cuenta activada:



Como podemos ver hemos podido iniciar sesión y nos han dado la bienvenida con el nombre de usuario

Consulta del usuario:

```
select * from usuario where correoUsuario = 'jorge@hotmail.com';
```



The screenshot shows a database query result in a web application. The query is `select * from usuario where correoUsuario = 'jorge@hotmail.com';`. The result is displayed in a table with the following columns: `idUsuario`, `nombreUsuario`, `correoUsuario`, `contrasenaUsuario`, and `activo`. The first row contains the values: 1, jorge, jorge@hotmail.com, 1234, and Activo.

	idUsuario	nombreUsuario	correoUsuario	contrasenaUsuario	activo
1	1	jorge	jorge@hotmail.com	1234	Activo

- Segunda prueba: administración de usuarios

Esta segunda prueba sirve para comprobar el funcionamiento de la activación/desactivación de usuarios y el borrado de usuarios

Activación de usuarios:

Aquí podemos ver la tabla de usuarios, con el usuario creado en las pruebas anteriores desactivado además el usuario con el que hemos iniciado sesión no se encuentra en la tabla por lo que el código que impide que un usuario se pueda modificar a sí mismo funciona:

Gestión de usuarios

[Volver atras](#)

Nombre Del Usuario	Correo Del Usuario	Contraseña Del Usuario	Activo
admin	admin@email.com	1234	Inactivo
adminume	ume@email.com	1234	Inactivo
bombero	bombero@email.com	1234	Inactivo
Jorge	jorge@email.com	1234	Inactivo
Jorge Prieto	jorge@gmail.com	1234	Activo
Usuario Nuevo	nuevoUsuario@email.com	1234	Activo

[Activar/desactivar usuario](#) [Eliminar usuario](#)

En estas imágenes podemos ver que tras activar el usuario este ha sido modificado tanto en la tabla como en la base de datos:

Gestión de usuarios

[Volver atras](#)

Nombre Del Usuario	Correo Del Usuario	Contraseña Del Usuario	Activo
admin	admin@email.com	1234	Inactivo
adminume	ume@email.com	1234	Inactivo
bombero	bombero@email.com	1234	Inactivo
Jorge	jorge@email.com	1234	Activo
Jorge Prieto	jorge@gmail.com	1234	Activo
Usuario Nuevo	nuevoUsuario@email.com	1234	Activo

[Activar/desactivar usuario](#)[Eliminar usuario](#)

```
select * from usuario where correoUsuario = 'jorge@email.com';
```

100 %

ResultsMessages

	idUsuario	nombreUsuario	correoUsuario	contrasenaUsuario	activo
1	15	Jorge	jorge@email.com	1234	Activo

La segunda parte comprueba la eliminación de usuarios:

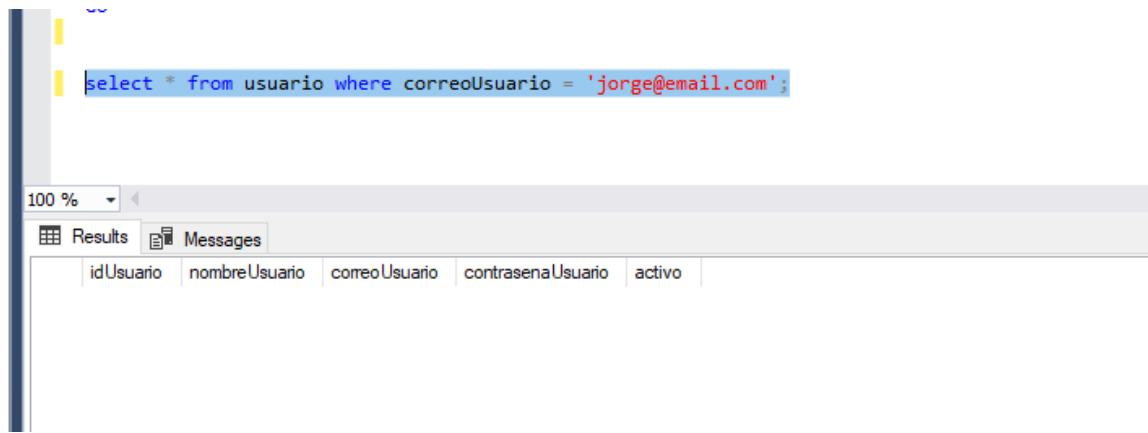
Podemos ver en las imágenes que tras eliminar al usuario anterior, este es eliminado de la tabla y de la base de datos

Gestión de usuarios

[Volver atras](#)

Nombre Del Usuario	Correo Del Usuario	Contraseña Del Usuario	Activo
admin	admin@email.com	1234	Inactivo
adminume	ume@email.com	1234	Inactivo
bombero	bombero@email.com	1234	Inactivo
Jorge Prieto	jorge@gmail.com	1234	Activo
Usuario Nuevo	nuevoUsuario@email.com	1234	Activo

[Activar/desactivar usuario](#)[Eliminar usuario](#)



- Tercera prueba: administración de localidades

Esta tercera prueba se encarga de demostrar el funcionamiento de la eliminación, inserción y modificación de localidades

Creación de localidades:

En esta parte de la prueba creo una localidad llamada Prueba:

The screenshot shows a web form titled "Añadir Localidad" on a blue background. In the top right corner, there is a yellow button labeled "Volver atras". The form contains the following fields:

- Provincia:** A dropdown menu with "Almería" selected.
- Nombre de la localidad:** A text input field containing "Prueba".
- Temperatura Media Primavera:** A text input field containing "20".
- Humedad Media Primavera:** A text input field containing "60".
- Temperatura Media Verano:** A text input field containing "10".
- Humedad Media Verano:** A text input field containing "100".
- Temperatura Media Otoño:** A text input field containing "30".
- Humedad Media Otoño:** A text input field containing "30".
- Temperatura Media Invierno:** A text input field containing "20".
- Humedad Media Invierno:** A text input field containing "60".

At the bottom center of the form is a yellow button labeled "Añadir".

Tras crearla podemos ver que aparece en la tabla cargada y en la base de datos con los mismos valores que introducimos:

Gestión de localidades

Volver atrás

Nombre de la Localidad	Nombre de la Provincia
Alcalá del Júcar	Albacete
Ayala	Álava
Denia	Alicante
Lastres	Asturias
Mérida	Badajoz
Níjar	Almería
Prueba	Almería
Puertollano	Ciudad Real

Añadir localidad Modificar localidad Eliminar localidad

```
select * from localidad where nombreLocalidad = 'Prueba';
```

100 %

Results

Messages

	idLocalidad	idProvincia	nombreLocalidad	temperaturaMediaPrimavera	humedadMediaPrimavera	temperaturaMediaVerano	humedadMediaVerano	temperaturaMediaOtoño	humedadMediaOtoño	temperaturaMediaInvierno	humedadMediaInvierno
1	15	3	Prueba	20	60	10	100	30	30	20	60

Modificación de localidades:

Esta es la segunda parte de la prueba, y en ella modificaré la localidad creada en la primera parte y la mostraré en la base de datos

Modificar Localidad

Volver atrás

Provincia: Almería

Nombre de la localidad: Prueba

Temperatura Media Primavera: 50 Humedad Media Primavera: 60

Temperatura Media Verano: 2 Humedad Media Verano: 0

Temperatura Media Otoño: 50 Humedad Media Otoño: 30

Temperatura Media Invierno: 20 Humedad Media Invierno: 60

Modificar

Como podemos ver en la siguiente imagen la modificación ha sido exitosa pues la localidad tiene los valores modificados:

```
select * from localidad where nombreLocalidad = 'Prueba';
```

	idLocalidad	idProvincia	nombreLocalidad	temperaturaMediaPrimavera	humedadMediaPrimavera	temperaturaMediaVerano	humedadMediaVerano	temperaturaMediaOtoño	humedadMediaOtoño	temperaturaMediaInvierno	humedadMediaInvierno
1	15	3	Prueba	50	60	2	0	50	30	20	60

Eliminación de localidades:

Por último borraré la localidad prueba:

Gestión de localidades

Volver atrás

Nombre de la Localidad	Nombre de la Provincia
Alcalá del Júcar	Albacete
Ayala	Álava
Denia	Alicante
Lastres	Asturias
Mérida	Badajoz
Níjar	Almería
Prueba	Almería
Puertollano	Ciudad Real

Localidad y datos meteorológicos borrados
Aceptar

Añadir localidad Modificar localidad Eliminar localidad

Como podemos ver tras eliminarla ya no se encuentra ni en la tabla ni en la base de datos

```
select * from localidad where nombreLocalidad = 'Prueba';
```

idLocalidad	idProvincia	nombreLocalidad	temperaturaMediaPrimavera	humedadMediaPrimavera	temperaturaMediaVerano	humedadMediaVerano	temperaturaMediaOtoño	humedadMediaOtoño	temperaturaMediaInvierno	humedadMediaInvierno
-------------	-------------	-----------------	---------------------------	-----------------------	------------------------	--------------------	-----------------------	-------------------	--------------------------	----------------------

Nombre de la Localidad	Nombre de la Provincia
Alcalá del Júcar	Albacete
Ayala	Álava
Denia	Alicante
Lastres	Asturias
Mérida	Badajoz
Níjar	Almería
Puertollano	Ciudad Real

- Cuarta prueba: administración de datos climatológicos

Esta es la cuarta prueba que voy a realizar, la cual se encarga de comprobar el correcto funcionamiento de la inserción, modificación y eliminación de registros climatológicos

Creación de registros:

En esta parte de la prueba voy a crear un registro en la localidad:

En esta primera imagen podemos ver que he creado un nuevo registro en la localidad de Mérida en Badajoz:

Provincia: Badajoz

Localidad: Mérida

Temperatura media: 50

HumedadMedia: 30

Fecha de inicio: 10/09/2021

Fecha de finalización: 03/12/2021

Añadir

Tras crearlo aparece en la tabla de registros climatológicos y en la base de datos confirmándose que la inserción funciona:

Modificación de registros climatológicos					Volver atras
Nombre de la Provincia	Nombre de la Localidad	Temperatura media	Fecha de inicio	Fecha de finalización	
Ciudad Real	Puertollano	17	18/06/2012	18/06/2012	
Albacete	Alcalá del Júcar	17	18/06/2012	30/06/2012	
Alicante	Denia	17	18/06/2012	18/06/2012	
Badajoz	Mérida	50	10/09/2021	03/12/2021	
Albacete	Alcalá del Júcar	30	01/12/2021	26/12/2021	

Añadir Registro

Modificar Registro

Eliminar Registro

```
select * from datoMeteorologico where idLocalidad = 10;
```

100 %

Results		Messages				
	idDato	idLocalidad	temperaturaMedia	humedadMedia	fechaDeInicio	fechaDeFinalizacion
1	31	10	50	30	2021-09-10	2021-12-03

Ahora voy a intentar insertar un dato climatológico en la misma localidad durante una fecha que esté ocupada por el dato anterior:

Como podemos ver el programa no nos deja insertar el nuevo dato y nos avisa de que ya existe otro dato en la misma localidad y que la fecha coincide

Añadir Registro

Volver atras

Provincia: Badajoz

Localidad: Mérida

Temperatura media: 24

HumedadMedia: 12

Fecha de inicio: 01/12/2021

Fecha de finalización: 18/12/2021

Ya existe un dato climatologico en esa localidad en esa fecha

Aceptar

Podemos también observar que no se ha insertado nada en la base de datos:

```
select * from datoMeteo where nombreLocalidad = 'Mérida';
```

```
select * from datoMeteo where idLocalidad = 10;
```

100 %

Results Messages

	idDato	idLocalidad	temperaturaMedia	humedadMedia	fechaDeInicio	fechaDeFinalizacion
1	31	10	50	30	2021-09-10	2021-12-03

A continuación voy a insertarlo cambiando la fecha por una que no coincida en ningún día con la del dato anterior y como podemos ver esta vez se ha insertado con éxito en la base de datos:

Añadir Registro

[Volver atrás](#)

Provincia:

Localidad:

Temperatura media:

HumedadMedia:

Fecha de inicio:

Fecha de finalización:

Dato climatológico insertado con éxito

[Aceptar](#)

```
select * from datoMeteo where idLocalidad = 10;
```

100 %

Results Messages

	idDato	idLocalidad	temperaturaMedia	humedadMedia	fechaDeInicio	fechaDeFinalizacion
1	31	10	50	30	2021-09-10	2021-12-03
2	32	10	24	12	2021-12-16	2021-12-25

Después, en esta parte de la prueba voy a modificar el segundo dato creado, seleccionandolo en la tabla y pulsando modificar:

Modificación de registros climatológicos

Volver atras

Nombre de la Provincia	Nombre de la Localidad	Temperatura media	Fecha de inicio	Fecha de finalización
Albacete	Alcalá del Júcar	17	18/06/2012	30/06/2012
Albacete	Alcalá del Júcar	30	01/12/2021	26/12/2021
Alicante	Denia	17	18/06/2012	18/06/2012
Badajoz	Mérida	50	10/09/2021	03/12/2021
Badajoz	Mérida	24	16/12/2021	25/12/2021
Ciudad Real	Puertollano	17	18/06/2012	18/06/2012

Añadir Registro

Modificar Registro

Eliminar Registro

Modificar registro

Volver atras

Provincia:

Badajoz

Localidad:

Mérida

Temperatura media:

30

HumedadMedia:

40

Fecha de inicio:

17/12/2021

15

Fecha de finalización:

31/12/2021

15

Modificar

```
select * from datoMeteorologico where idLocalidad = 10;
```

100 %

Results Messages

	idData	idLocalidad	temperaturaMedia	humedadMedia	fechaDelInicio	fechaDeFinalizacion
1	31	10	50	30	2021-09-10	2021-12-03
2	32	10	30	40	2021-12-17	2021-12-31

Modificación de registros climatológicos

[Volver atras](#)

Nombre de la Provincia	Nombre de la Localidad	Temperatura media	Fecha de inicio	Fecha de finalización
Ciudad Real	Puertollano	17	18/06/2012	18/06/2012
Albacete	Alcalá del Júcar	17	18/06/2012	30/06/2012
Alicante	Denia	17	18/06/2012	18/06/2012
Badajoz	Mérida	50	10/09/2021	03/12/2021
Albacete	Alcalá del Júcar	30	01/12/2021	26/12/2021

Añadir Registro

Modificar Registro

Eliminar Registro

select * from datoMeteorologico where idLocalidad = 10;

100 %

Results Messages

	idData	idLocalidad	temperaturaMedia	humedadMedia	fechaDelInicio	fechaDeFinalizacion
1	31	10	50	30	2021-09-10	2021-12-03

- Quinta prueba: administración de incendios

Esta es la quinta prueba que voy a realizar y en ella compruebo el funcionamiento de la inserción, modificación y borrado de los incendios

Creación de incendios:

En esta parte de la prueba creo un incendio en la localidad de Mérida:

Formulario para añadir un incendio. El fondo es rojo. En la parte superior izquierda está el título 'Añadir incendio' en amarillo, y a la derecha un botón 'Volver atras' en un recuadro azul. Los campos de entrada son:

- Provincia: Badajoz (seleccionado en un menú desplegable)
- Localidad: Mérida (seleccionado en un menú desplegable)
- Hectáreas quemadas: 1000 (campo de texto)
- Temperatura media: 20 (campo de texto)
- Humedad Media: 40 (campo de texto)
- Fecha de inicio: 02/12/2021 (campo de texto con icono de calendario)
- Fecha de extinción: 26/12/2021 (campo de texto con icono de calendario)

En la parte inferior hay un botón 'Añadir' en un recuadro azul.

Como podemos ver el incendio creado se muestra tanto en la tabla de incendios como en la base de datos confirmándose que la creación de incendios funciona:

Pantalla de 'Registros de incendios' con fondo amarillo. En la parte superior está el título 'Registros de incendios' en rojo, y a la derecha un botón 'Volver atras' en un recuadro azul. Debajo de la barra de título hay una tabla con los siguientes datos:

Provincia	Localidad	Temperatura Media	Humedad Media	Hectareas	Fecha de inicio	Fecha de extinción
Alicante	Denia	17	22	2000	18/06/2012	18/06/2012
Albacete	Alcalá del Júcar	17	22	2000	18/06/2012	18/06/2012
Ciudad Real	Puertollano	17	22	2000	18/06/2012	18/06/2012
Almería	Níjar	50	80	1000	02/12/2021	18/12/2021
Badajoz	Mérida	20	40	1000	02/12/2021	26/12/2021

Debajo de la tabla hay un recuadro gris vacío. En la parte inferior de la pantalla hay tres botones: 'Añadir Incendio', 'Modificar Incendio' y 'Eliminar Incendio'.

```
select * from incendio where idLocalidad = 10;
```

100 %							
Results Messages							
	idIncendio	idLocalidad	hectareasQuemadas	temperaturaMedia	humedadMedia	fechaDelInicio	fechaDeExtinción
1	15	10	1000	20	40	2021-12-02	2021-12-26

La segunda prueba es la de modificación de los datos del incendio, y como podemos ver los datos modificados se guardan en la base de datos, confirmando que la modificación de registros de incendios funciona correctamente:

Modificar Incendio

[Volver atras](#)

Provincia:

Localidad:

Hectáreas quemadas:

Temperatura media:

HumedadMedia:

Fecha de inicio:

Fecha de extinción:

[Modificar](#)

```
select * from incendio where idLocalidad = 10;
```

100 %							
Results Messages							
	idIncendio	idLocalidad	hectareasQuemadas	temperaturaMedia	humedadMedia	fechaDelInicio	fechaDeExtinción
1	15	10	200	10	29	2021-12-03	2021-12-26

Registros de incendios

[Volver atras](#)

Provincia	Localidad	Temperatura Media	Humedad Media	Hectareas	Fecha de inicio	Fecha de extinción
Alicante	Denia	17	22	2000	18/06/2012	18/06/2012
Albacete	Alcalá del Júcar	17	22	2000	18/06/2012	18/06/2012
Ciudad Real	Puertollano	17	22	2000	18/06/2012	18/06/2012
Almería	Níjar	50	80	1000	02/12/2021	18/12/2021
Badajoz	Mérida	10	29	200	03/12/2021	26/12/2021

[Añadir Incendio](#)
[Modificar Incendio](#)
[Eliminar Incendio](#)

Por último probaré la eliminación de incendios con el incendio que hemos creado, tras la eliminación del registro este se elimina de la tabla de incendios y de la base de datos confirmándose su correcto funcionamiento:

Registros de incendios

[Volver atras](#)

Provincia	Localidad	Temperatura Media	Humedad Media	Hectareas	Fecha de inicio	Fecha de extinción
Alicante	Denia	17	22	2000	18/06/2012	18/06/2012
Albacete	Alcalá del Júcar	17	22	2000	18/06/2012	18/06/2012
Ciudad Real	Puertollano	17	22	2000	18/06/2012	18/06/2012
Almería	Níjar	50	80	1000	02/12/2021	18/12/2021

[Añadir Incendio](#)
[Modificar Incendio](#)
[Eliminar Incendio](#)

```
select * from incendio where idLocalidad = 10;
```

100 %
Results Messages

idIncendio	idLocalidad	hectareasQuemadas	temperaturaMedia	humedadMedia	fechaDelInicio	fechaDeExtinción
------------	-------------	-------------------	------------------	--------------	----------------	------------------

- Sexta prueba: predicción de incendios

Esta es el último test realizado sobre la aplicación y este se encarga de comprobar el correcto funcionamiento de el algoritmo de predicción de incendios, para ello voy a realizar un cálculo de la probabilidad de incendio en una localidad:

La prueba es en la localidad de Dénia, donde he añadido un dato climatológico con una temperatura de 60 grados y una humedad del 0% y un incendio durante diciembre:

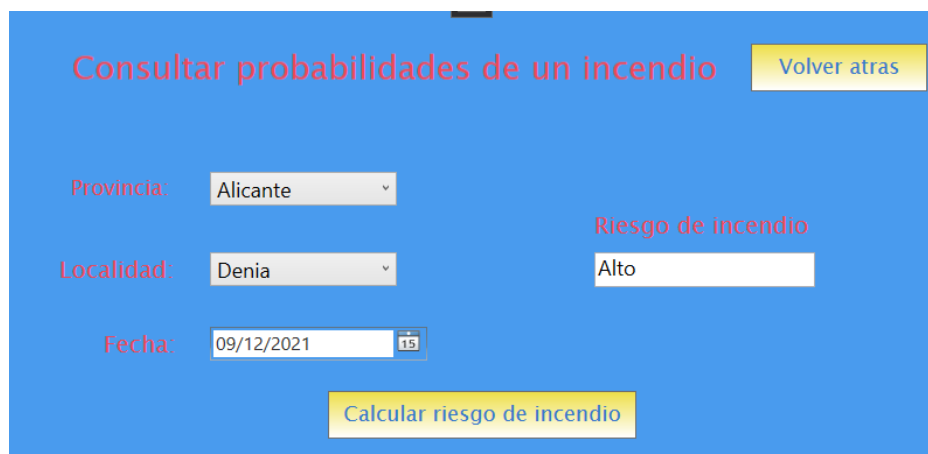
Dato climatológico:

Alicante	Denia	60	03/12/2021	26/12/2021
----------	-------	----	------------	------------

Incendio:

Alicante	Denia	30	20	1000	02/12/2021	19/12/2021
----------	-------	----	----	------	------------	------------

Tras calcular el riesgo de incendio el día 9 de diciembre el riesgo es alto:



Este resultado es correcto porque al ser la cantidad de incendios durante los últimos tres meses del año de tan solo 1 se, suma 1 al número del cálculo de la probabilidad:

```
int probabilidadInt = 0;

if (incendios == 1)
{
    probabilidadInt++;
}
else if (incendios >= 2 && incendios <= 5)
{
    probabilidadInt += 2;
}
else if (incendios > 5)
{
    probabilidadInt += 3;
}
```

Al ser la temperatura superior a 30 se suma 2 en el número del cálculo de la probabilidad:

```
if (temperatura <= 0 && temperatura >= -20)
{
    probabilidadInt--;
}
else if (temperatura < -20)
{
    probabilidadInt -= 2;
}
else if (temperatura >= 15 && temperatura <= 30)
{
    probabilidadInt++;
}
else if (temperatura > 30)
{
    probabilidadInt += 2;
}
```

Y al ser la humedad inferior a 20 también se suma 2 al este número dando como resultado 5 que está dentro del valor de alto

```
if (humedad >= 0 && humedad <= 20)
{
    probabilidadInt += 2;
}
else if (humedad > 20 && humedad <= 40)
{
    probabilidadInt++;
}
else if (humedad >= 60 && humedad <= 80)
{
    probabilidadInt--;
}
else if (humedad > 80)
{
    probabilidadInt -= 2;
}
```

```
if (probabilidadInt >= 7)
{
    probabilidad = ("Muy alto");
}
else if (probabilidadInt <= 6 && probabilidadInt >= 4)
{
    probabilidad = ("Alto");
}
else if (probabilidadInt <= 3 && probabilidadInt >= 1)
{
    probabilidad = ("Medio");
}
else if (probabilidadInt <= 0)
{
    probabilidad = ("Bajo");
}
return probabilidad;
```

Conclusiones:

En general estoy contento con el proyecto realizado, pues originalmente no sabía que proyecto realizar y mi tutora me propuso esta idea de un sistema de predicción de incendios con su propio "Algoritmo" para el cálculo de las probabilidades, y además su realización en wpf de c# para ampliar mis conocimientos sobre este y los de creación de consultas con el Entity Data Model, al principio tuve algunas dificultades, con la animación del login y problemas con las consultas, además de otros problemas con las tablas, pues no podía sacar información de las celdas como si se puede realizar en Forms, solo obtener la fila seleccionada y a través de su índice obtener el elemento en una lista, pero pese a estos problemas estoy satisfecho con el trabajo y lo estudiado en el curso me facilitaron la realización del proyecto, como por ejemplo el uso de GitHub o la creación de diagramas de clases

Información adicional:

- Video en youtube del proyecto: <https://youtu.be/trknoxGAuPTU>
- Video en youtube del prototipo: <https://youtu.be/9EALf6oRvAo>
- Enlace al repositorio de github:
https://github.com/Jorge-Prieto-Medina1/Proyecto_Predicci-n_incenddios_UME

[s_UME](#)