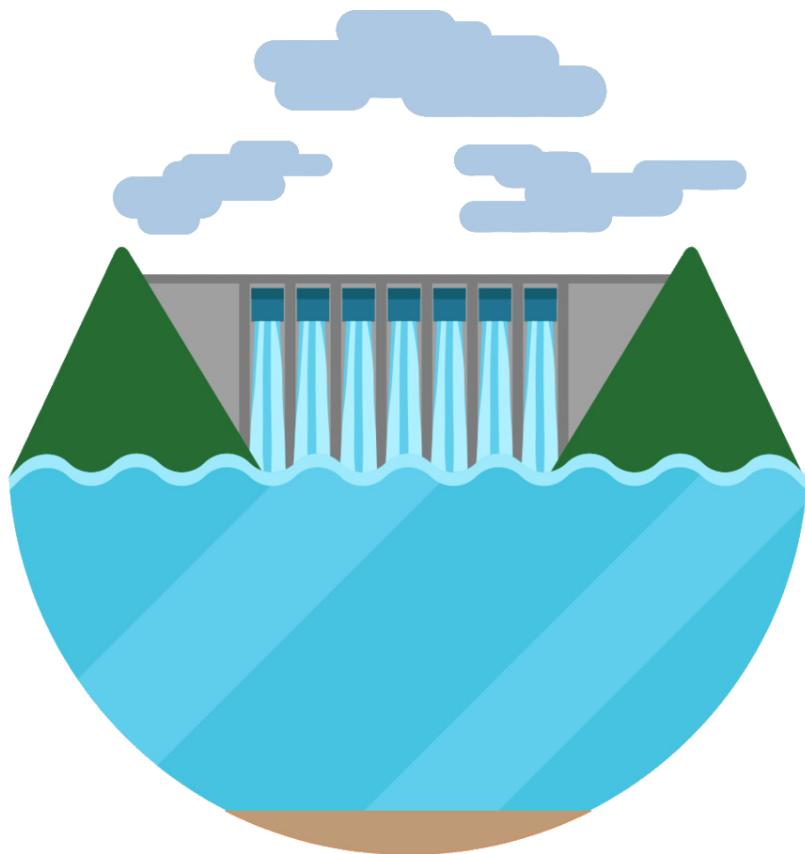


Sistema de presas



PATRONES SOFTWARE

17/01/2020

Grupo: Álvaro de las Heras Fernández a.heras@edu.uah.es 03146833L
José Luis González Fernández-Cid josel.gonzalez@edu.uah.es 03210148S

Índice

Índice	1
1. Enunciado y requisitos	3
1.1. Enunciado	3
1.2. Funcionalidad	3
1.3. Usuarios	3
1.4. Datos	3
1.5. Interfaz	4
2. Manual de usuario	5
2.1. Instalación	5
2.1.1. Prerrequisitos	5
2.1.2. Despliegue	5
2.2. Ejecución	8
2.2.1 Pantalla de inicio	8
2.2.2 Pantalla de privacidad	11
2.2.1 Pantalla configuración	11
2.2.1 Pantallas de gestión de cuentas	13
3. Diseño y patrones	15
3.1. Diagramas de clases	15
3.1.1. Aplicación web	15
3.1.1.1. CLASE ALGORITHM GENETIC Y CHATHUB	15
3.1.1.2. CLASE CONFIGURATIONCONTROLLER	16
3.1.1.3. CLASE HOMECONTROLLER	16
3.1.1.4. PATRÓN PROXY	17
3.1.1.5. PATRÓN CHAIN OF RESPONSABILITY	17
3.1.1.6. PATRÓN ITERATOR	18
3.1.1.7. PATRÓN FACTORY-METHOD/SINGLETON/TEMPLATE METHOD	18
3.1.1.8. OTRAS CLASES	19
3.1.2 Módulo de algoritmos genéticos	20
3.1.2.1. CLASE DNA	20
3.1.2.2. CLASE POBLACIÓN	21
3.1.2.2. PATRÓN BUILDER	22
3.1.2.3. PATRÓN DECORATOR	22
3.1.2.4. PATRÓN STRATEGY	23
3.1.2.5. PATRÓN STATE	23
3.2. Diagramas de casos de uso	24
3.2.1. Aplicación web	24
3.2.1.1. CASO DE USO AUTENTIFICACIÓN	24

3.2.1.2. CASO DE USO CONSULTA DE PRIVACIDAD	24
3.2.1.3. CASO DE USO ITERAR MAPA	25
3.2.2. Módulo de algoritmos genéticos	25
3.2.2.1. Caso de uso búsqueda de punto de presa	25
3.3. Diagramas de despliegue	25
3.4. Patrones	26
3.4.1. Aplicación web	26
3.4.2 Módulo de algoritmos genéticos	31
3.5 Diagrama de secuencia	36
3.5.1 SECUENCIA SIMULACIÓN	36

1. Enunciado y requisitos

1.1. Enunciado

1. Se desarrollará una página web en .NET Core que permitirá:
 - a. Visualizar mapas en formato GeoJSON.
 - b. Dibujar presas y anegar zonas del mapa.
 - c. Generar presas de forma automática y óptima.
 - d. Configurar la visualización y generación.
 - e. Cargar mapas en el sistema.
 - f. Almacenar los mapas en una base de datos.
2. Se desarrollarán algoritmos genéticos que se encargarán de elegir los mejores puntos para la construcción de presas, pudiendo configurarse por parte del usuario.

1.2. Funcionalidad

La página web se encargará principalmente de visualizar los mapas, permitir carga de mapas, guardado de mapas y creación de presas, esta se desarrolla con .NET Core. Además para la agilización de los datos crearemos una base de datos en la que se almacenarán los mapas, con Microsoft SQL Server.

El algoritmo genético recibe un mapa formato GeoJSON, recibe un dato de altura de la presa y recibe un dato de longitud de la presa(opcional), este se desarrollará en C#.

El algoritmo genético irá eligiendo 2 puntos donde colocará el muro de la presa y calculará el volumen de agua de la presa que se generará al añadir el muro en esos puntos.

Este algoritmo genético se podrá personalizar cambiando una serie de parámetros.

Todos estos datos que genera el algoritmo se recibirán en la página web que permitirá la visualización de los mapas GeoJSON donde dibujará encima la presa, y una posible área anegada a causa de ella.

1.3. Usuarios

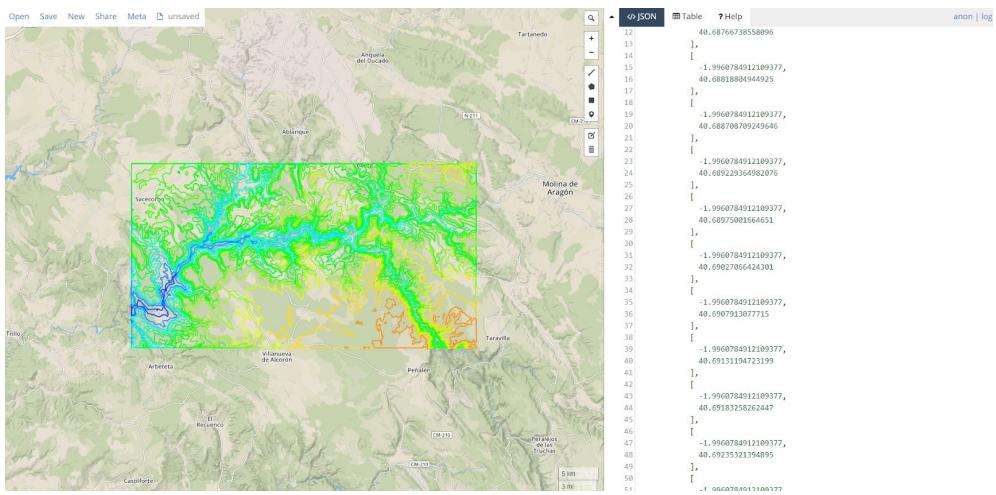
Los usuarios que se encarguen del uso del sistema podrán ser los siguientes:

- **Administrador:** será el encargado de modificar el sistema y parámetros para asegurar el correcto funcionamiento.
- Usuarios que quieran **optimizar**, usará los algoritmos para conseguir dicho objetivo, pudiendo ser o no experto en el tema.
- Usuarios que quieran dibujar presas **sin optimizar**, podrá dibujar la presa donde quiera el mapa sin emplear los algoritmos.

1.4. Datos

Los datos que maneja el programa son:

- **Mapas GeoJSON** son archivos en formato JSON que sirven para representar mapas. Estos serán la base del proyecto.



Mapa GeoJSON generado de prueba de Guadalajara.

- **Presas** (longitud del muro y altura del muro) las definirá el usuario y se mostrarán sobre el GeoJSON.
- **Parámetros del algoritmo genético** permitirán una configuración avanzada del algoritmo, como número de individuos, tipo de algoritmo etc.

1.5. Interfaz

La interfaz que se va a desarrollar va a ser de una web sencilla y básica para hacer un especial énfasis en la visualización de los mapas junto a su manipulación.

GeneticDam Home Privacy

Genetic Dam System

This project finds the best spot in a topographic map to put a dam with genetic algorithms

Configuración

Longitud:
Introduzca la longitud de la presa

Altura:
Introduzca la altura de la presa

Algoritmo genético:
Introduzca los individuos

Aplicar

© 2019 - Genetic Dam by Álvaro de las Heras and Jose Luis Cid - [Privacy](#)

Concepto de interfaz de la web.

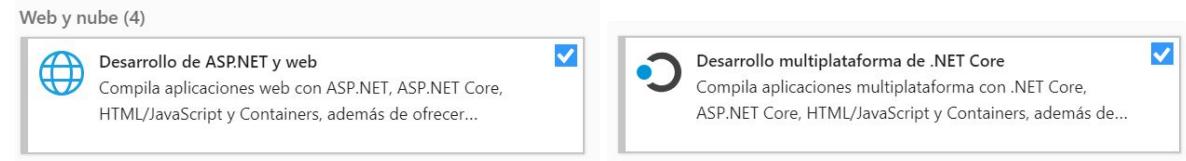
2. Manual de usuario

2.1. Instalación

Puesto que se trata de una web no requiere de instalación pero si de despliegue, para realizar el despliegue se hará de forma sencilla, utilizando la última tecnología puntera para ello como Microsoft Azure (Cloud).

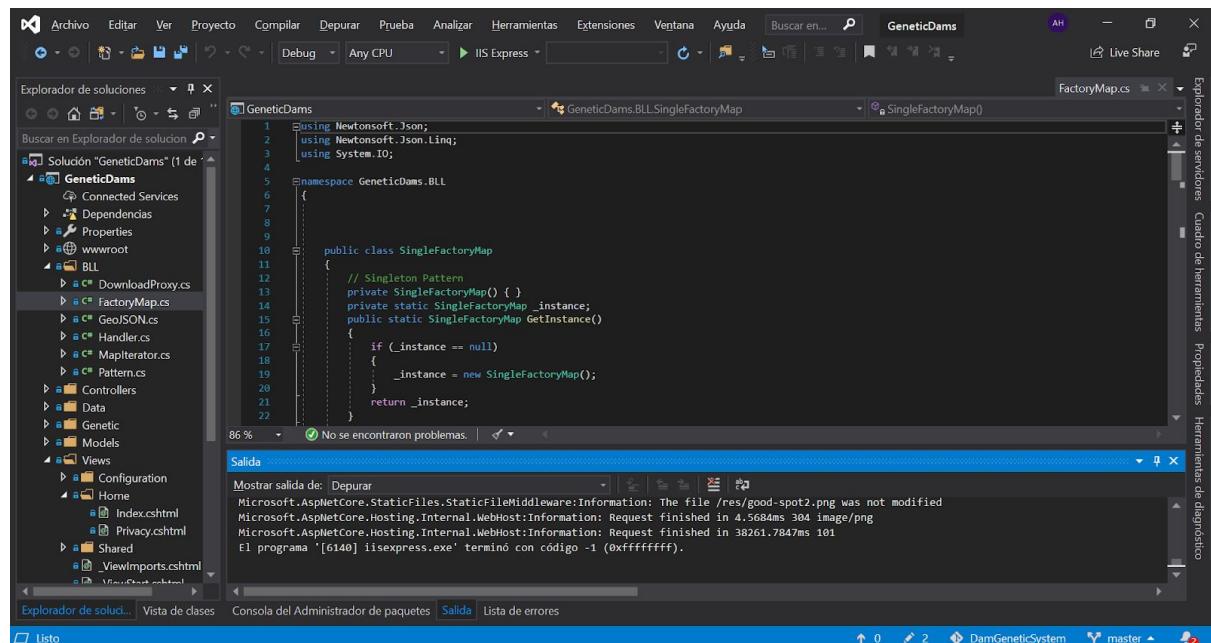
2.1.1. Prerrequisitos

- Disponer de Visual Studio 2019/2017 Instalado
- Disponer de los siguientes módulos para Visual Studio:



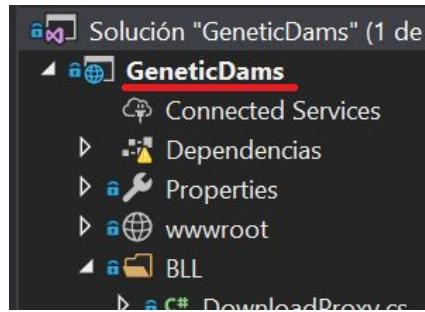
2.1.2. Despliegue

1. Con todos los prerrequisitos satisfechos, el siguiente paso es copiar el proyecto, desde Github, ya sea mediante **descarga** o **git clone**.
2. El siguiente paso es abrir el proyecto, para ello abriendo el archivo **GeneticDams.sln**, que abrirá en Visual Studio la solución completa.



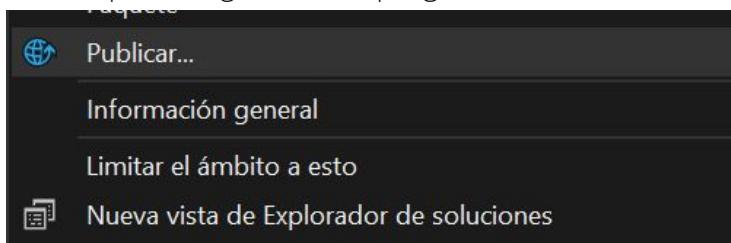
Pantalla que se mostrará al abrir el .sln

3. Una vez está abierto hemos de dirigirnos al proyecto web sobre el que haremos click izquierdo para desplegar el menú de opciones.



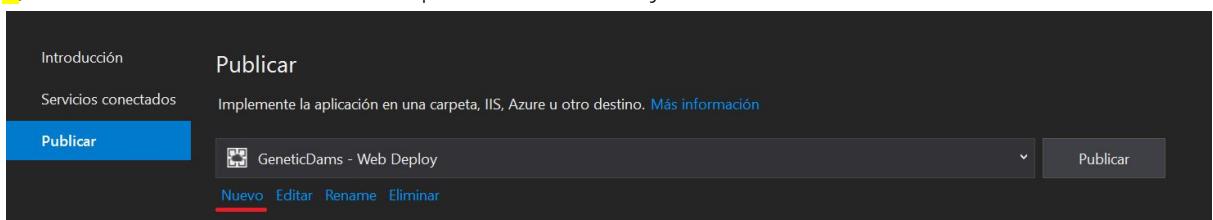
Señalado en rojo el proyecto que hay que elegir.

4. Dentro de las opciones del menú elija publicar que le llevará a una pantalla dónde tendrá que configurar el despliegue.

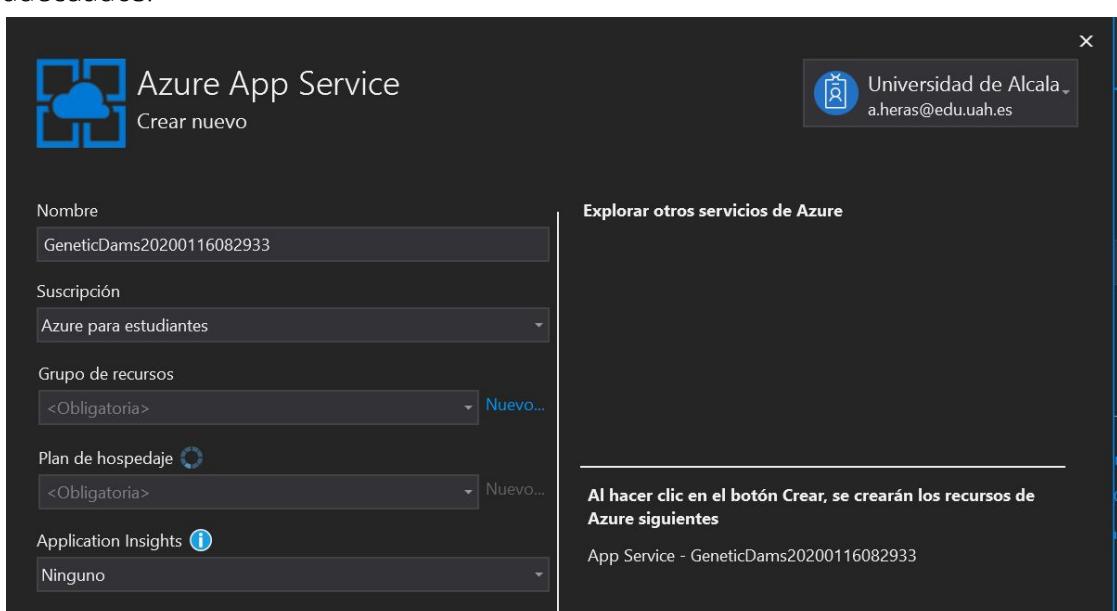


Opción de publicar.

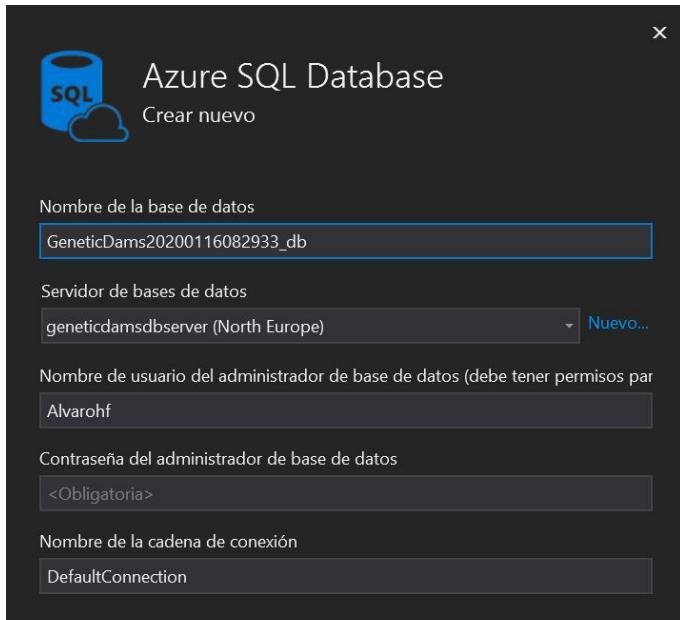
5. Si ya posee un entorno en Azure configurado para aplicaciones web, únicamente tienes pulsar el botón de publicar, de forma que lo hará automáticamente (vea paso X). En caso contrario deberá de pulsar en **nuevo** y **crear una nueva instancia**.



6. A continuación se muestra las opciones que se han de configurar de la nueva aplicación. Para realizarlo ha de **iniciar sesión** con su cuenta de **Azure**. Una vez haya iniciado deberá de **asignar nombres**, si lo desea a los recursos y elegir los adecuados.



7. En este caso además se ha de crear una base de datos, para poder así almacenar los usuarios, por lo que se ha de pulsar en **Crear una base de datos SQL**. Al igual de antes debe de configurarla, según sus necesidades, una vez esté creada deberá cambiar la cadena de conexión de la base de datos de usuarios, que se encuentra dentro del **appsettings.json**.



The screenshot shows the 'Azure SQL Database' creation dialog. It includes fields for 'Nombre de la base de datos' (GeneticDams20200116082933_db), 'Servidor de bases de datos' (geneticdamsdbserver (North Europe)), 'Nombre de usuario del administrador de base de datos' (Alvarohf), 'Contraseña del administrador de base de datos' (<Obligatoria>), and 'Nombre de la cadena de conexión' (DefaultConnection).

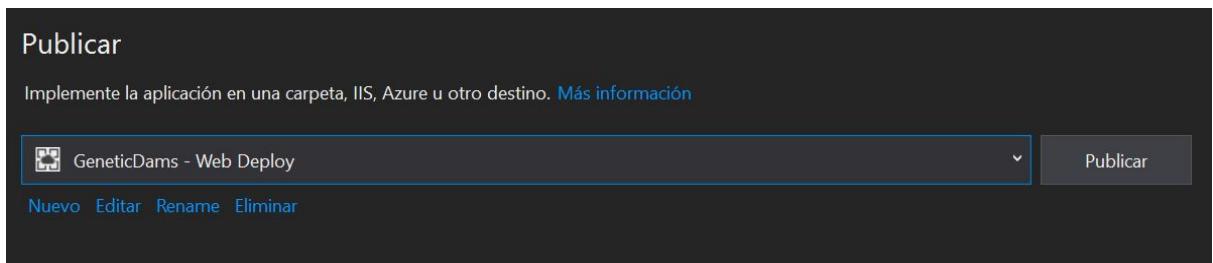


The screenshot shows the 'appsettings.json' file in the Azure portal. The JSON structure is as follows:

```
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "Data S
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Warning"
8      }
9    },
10   "AllowedHosts": "*"
11 }
```

Archivo en el que cambiar la cadena de conexión.

8. Al terminar la configuración de la base de datos estará todo configurado, por lo que se pulsará el botón de crear para tener ya la nueva instancia para utilizar.
9. Finalmente seleccione la instancia que se ha creado y pulse el botón de publicar, para así poder acceder a la web en la dirección que nos proporciona Azure.



The screenshot shows the 'Publicar' (Publish) dialog. It includes a description of implementing the application in a folder, IIS, or Azure, a dropdown menu for 'GeneticDams - Web Deploy', and buttons for 'Nuevo' (New), 'Editar' (Edit), 'Rename', 'Eliminar' (Delete), and 'Publicar' (Publish).

Selección de la instancia y publicación.

Resumen

URL del sitio: <https://geneticdams.azurewebsites.net>

Grupo de recursos: GeneticDams20200110011606ResourceGroup

Configuración: Release

Marco de trabajo de destino: netcoreapp2.2

Modo de implementación: Dependiente de marco de trabajo

Acciones

Obtener vista previa de cambios

Administrar en Cloud Explorer

Editar configuración de Azure App Service

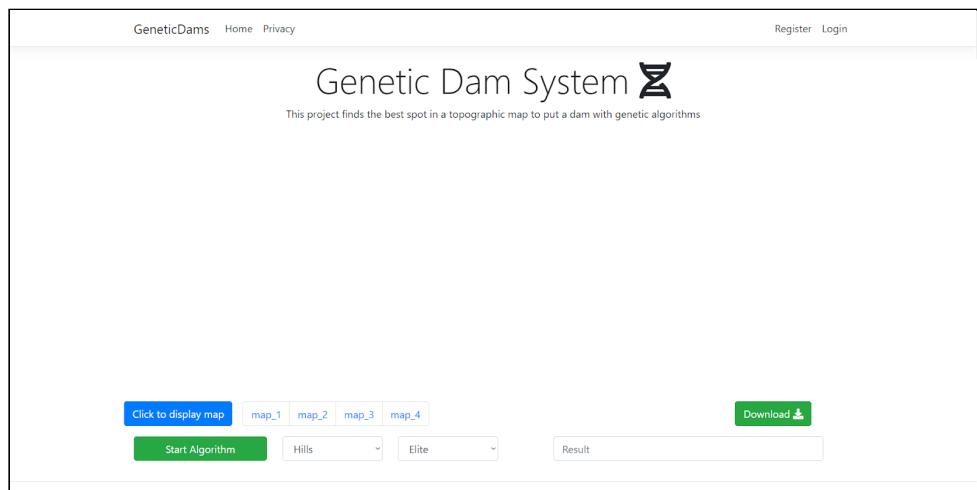
Abrir la Guía de solución de problemas

Resumen con los datos del despliegue, como la url de acceso.

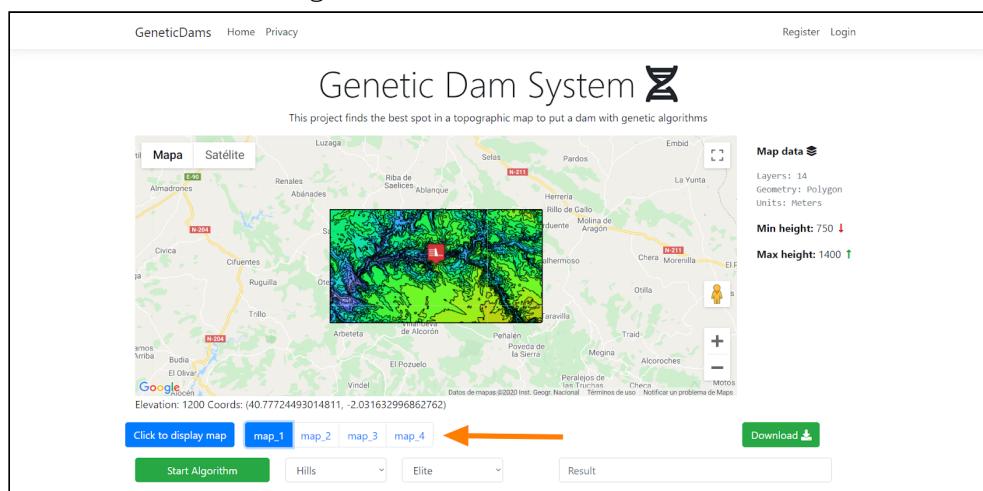
2.2. Ejecución

2.2.1 Pantalla de inicio

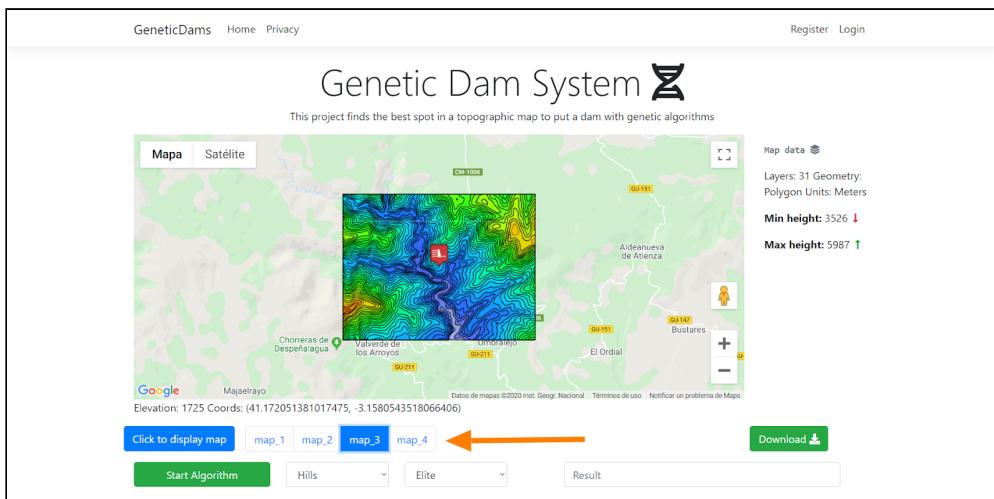
Para la ejecución de la aplicación deberá de abrir en primer lugar la dirección de página web que tenga asignada en este caso <https://geneticdams.azurewebsites.net>. La primera pantalla que se mostrará es la siguiente:



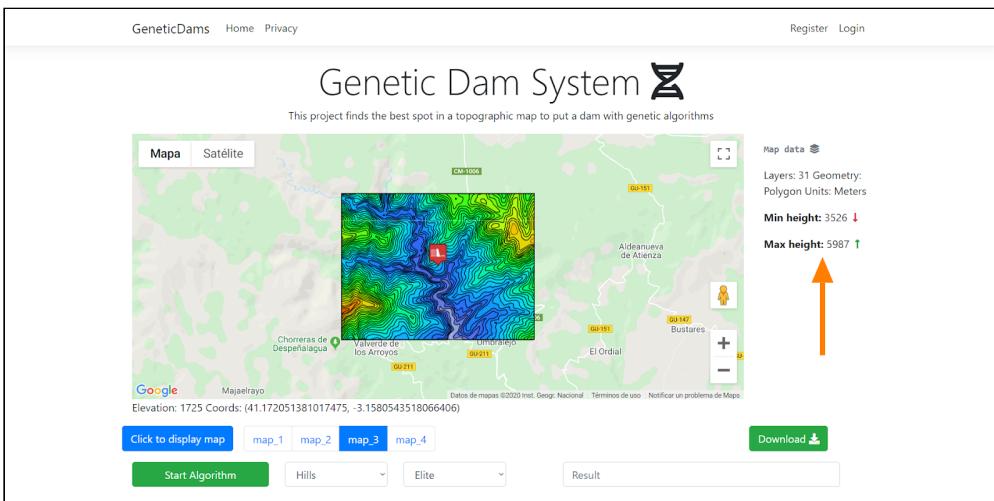
En lo que no habrá ningún mapa ni estará iniciada la sesión. Para mostrar un mapa basta con pulsar el botón de Click to display map o cualquiera de los 4 mapas disponibles. Mostrando una vista como la siguiente.



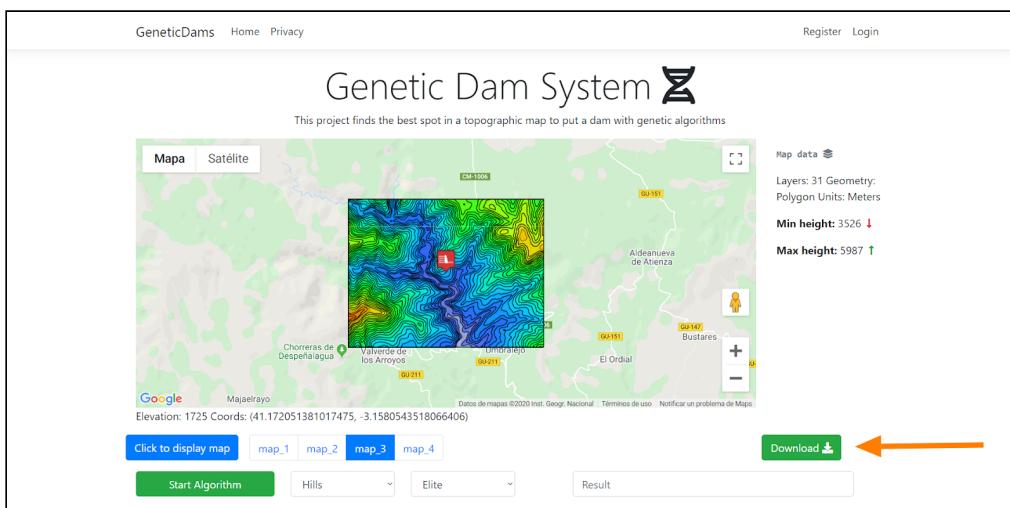
Se puede ir alternando entre mapas pulsando entre los disponibles como se ve en la imagen a continuación.

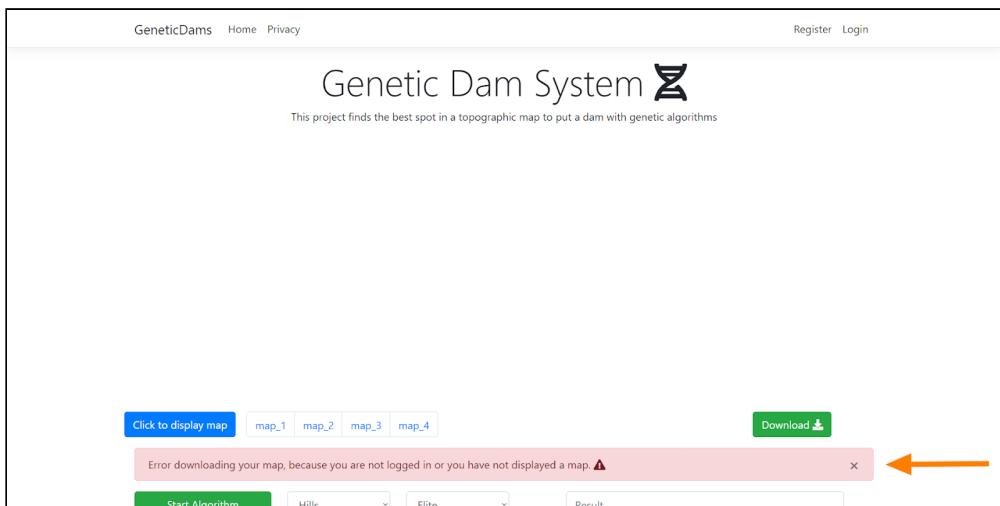


En la pantalla además de mostrarse el mapa junto con un marcador en el medio, se muestra información del mapa, como las capas, alturas o unidades. Además al mover el ratón por encima del mapa se muestra información en la parte inferior del mapa.

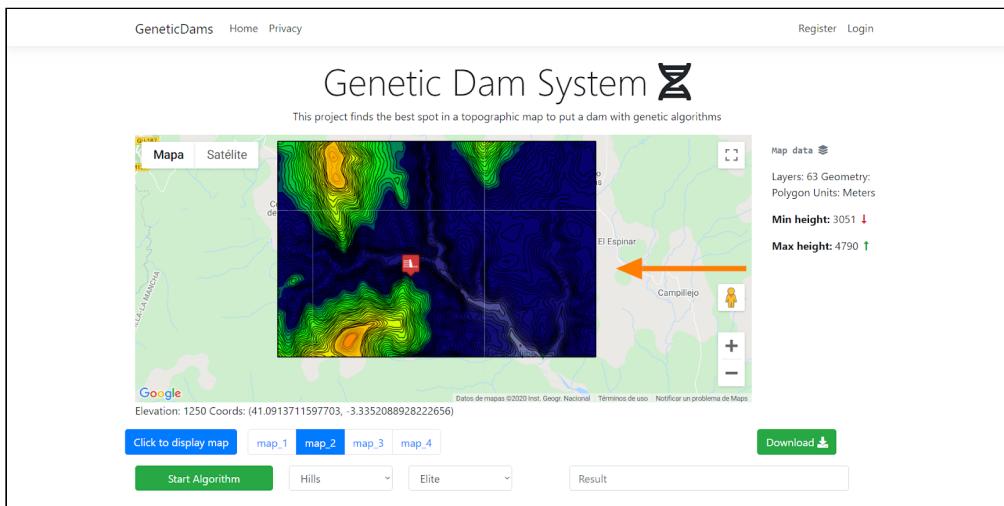


Si se quiere es posible descargar el GeoJSON del mapa, para ello bastará con pulsar el botón de Download, aunque implica que se ha de iniciar sesión, en caso contrario mostrará un mensaje de error.

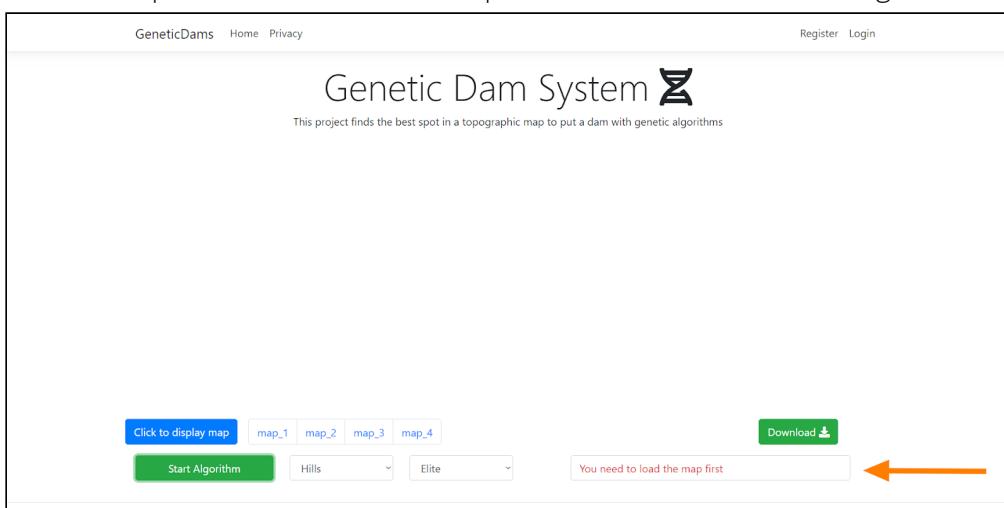




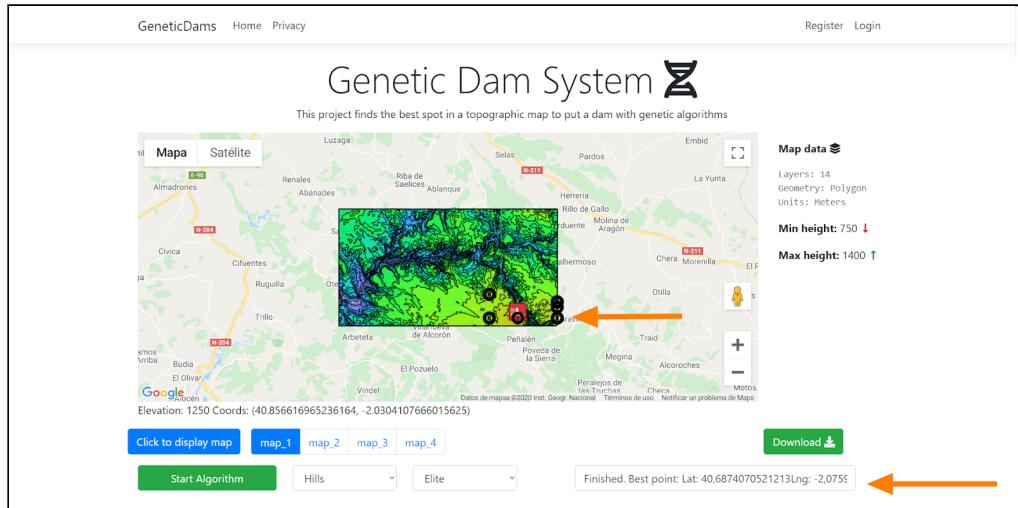
Para poder interactuar con el mapa hay dos formas una manual en la que colocas el punto en el mapa y otra automática que usa los algoritmos genéticos.
En el caso de la forma manual al colocar el punto en una capa se procederá a inundarlo con agua que colorea las capas inferiores de azul.



Para hacerlo con algoritmos genéticos hay que pulsar el botón Start Algorithm, con la configuración que deseemos, pudiendo buscar montañas o valles y usar selección por ruleta o élite. Aunque debe de haber un mapa abierto si no mostrará el siguiente error.



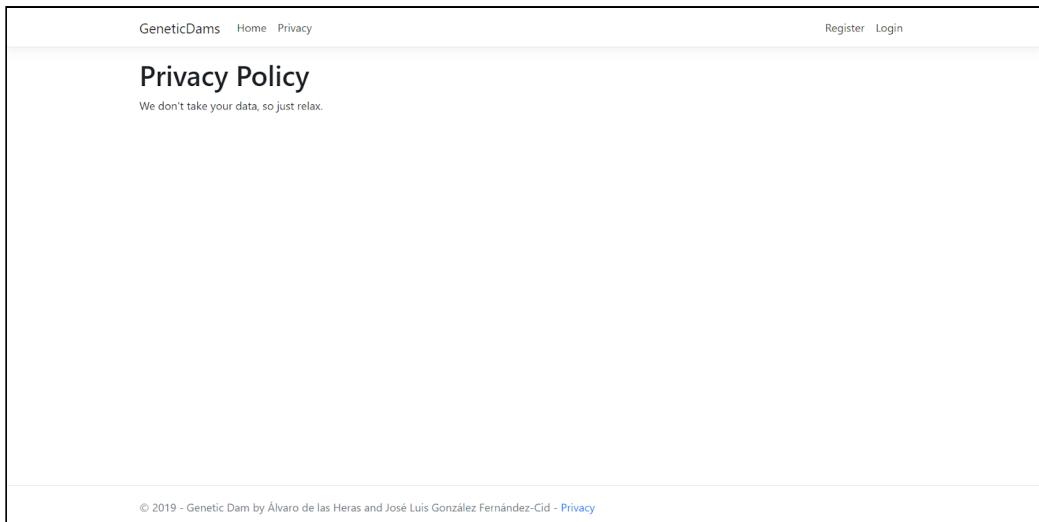
Si había un mapa cargado entonces se ejecutará de forma normal, mostrando los puntos que tiene como bolas 8 negras, y el mejor de todos con el icono rojo. Además se podrán ver las coordenadas dentro de la casilla Result, señalada con la flecha en la imagen.



Se puede acceder a otras pantallas de esta haciendo click en las respectivas secciones como Privacy, Register, Login, Configuration o tu nombre de usuario.

2.2.2 Pantalla de privacidad

Si se pulsa sobre privacy nos lleva a la pantalla de privacidad donde se explica la política de privacidad de la web, que en este caso es ninguna. Además si es la primera vez que se entra a la web saldrá un aviso de cookies, puesto que se usan para mantener la sesión de los usuarios.



2.2.1 Pantalla configuración

La pantalla de configuración sólo aparece para seleccionar en el menú una vez has iniciado sesión, esta permitirá configurar la página web a los distintos roles existentes que hay. En este momento solo hay 3 roles para los que se muestran mensajes personalizados. Estos roles son usuario, especialista y administrador. Correspondiéndose los dos últimos a los

usuarios con correo spec@uah.es y admin@uah.es, respectivamente con contraseña:
Contraseña1.

GeneticDams Home Privacy Hello spec@uah.es! Configuration Logout

Hello spec@uah.es!
Role: Specialist
You can configure the map options

© 2019 - Genetic Dam by Álvaro de las Heras and José Luis González Fernández-Cid - [Privacy](#)

GeneticDams Home Privacy Hello admin@uah.es! Configuration Logout

Hello admin@uah.es!
Role: Admin
You can configure the map options and system options

© 2019 - Genetic Dam by Álvaro de las Heras and José Luis González Fernández-Cid - [Privacy](#)

GeneticDams Home Privacy Hello test@mail.es! Configuration Logout

Hello test@mail.es!
Role: User
You can't configure the system or model, try to log with your specialist or admin account

© 2019 - Genetic Dam by Álvaro de las Heras and José Luis González Fernández-Cid - [Privacy](#)

2.2.1 Pantallas de gestión de cuentas

Si hacemos click en Login nos redirige a la ventana de iniciar sesión dentro de la cual se puede iniciar sesión introduciendo nuestros datos, recordar contraseña (todavía no implementado) o registrar. Si los datos son correctos se procede a iniciar si no se muestran mensajes de error.

The screenshot shows the login page for the GeneticDams application. At the top, there is a navigation bar with the site name "GeneticDams" on the left and "Register" and "Login" links on the right. Below the navigation bar, the main content area has a title "Log in" and two options: "Use a local account to log in." and "Use another service to log in.". A note states: "There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services." The "Email" field contains "test@mail.com" and the "Password" field contains a masked password. There is a "Remember me?" checkbox, which is unchecked. A blue "Log in" button is centered below the fields. Below the button are links for "Forgot your password?" and "Register as a new user". At the bottom of the page, a copyright notice reads "© 2020 - GeneticDams - [Privacy](#)".

Si se pulsa en Register se lleva a la siguiente ventana en el que se crea una cuenta al introducir correctamente los datos en el formulario. Si el usuario ya está registrado o hay algún campo erróneo muestra el mensaje de error correspondiente como se ve en la imagen.

The screenshot shows the registration page for the GeneticDams application. At the top, there is a navigation bar with the site name "GeneticDams" on the left and "Register" and "Login" links on the right. Below the navigation bar, the main content area has a title "Register" and a subtitle "Create a new account.". The form requires three inputs: "Email" (containing "test@mail.com"), "Password" (containing a masked password), and "Confirm password" (containing a masked password). Below these fields is a blue "Register" button. At the bottom of the page, a copyright notice reads "© 2020 - GeneticDams - [Privacy](#)".

GeneticDams

Register

Create a new account.

Email

 The Email field is required.

Password

 The Password field is required.

Confirm password

Register

© 2020 - GeneticDams - [Privacy](#)

Si una vez iniciada sesión haces click sobre tu nombre de usuario te redirige a un panel en el que te permite realizar gestiones con tus datos entre estas gestiones se puede consultar tu perfil, cambiar tu contraseña y gestionar tus datos personales. Sin embargo habría que implementar la verificación en dos pasos.

GeneticDams

Hello admin@uah.es! Configuration Logout

Manage your account
 Change your account settings

Profile

Profile	Profile
Password	Username <input type="text" value="admin@uah.es"/>
Two-factor authentication	
Personal data	Email <input type="text" value="admin@uah.es"/> Send verification email
	Phone number <input type="text"/>

Save

© 2020 - GeneticDams - [Privacy](#)

GeneticDams

Hello admin@uah.es! Configuration Logout

Manage your account
 Change your account settings

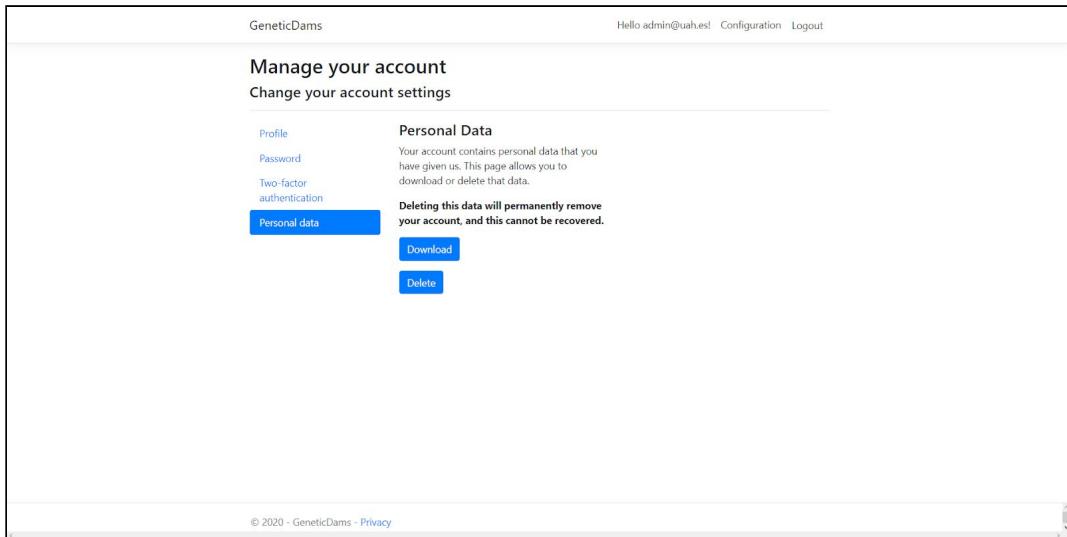
Profile

Change password

Profile	Current password <input type="password"/>
Password	New password <input type="password"/>
	Confirm new password <input type="password"/>

Update password

© 2020 - GeneticDams - [Privacy](#)



3. Diseño y patrones

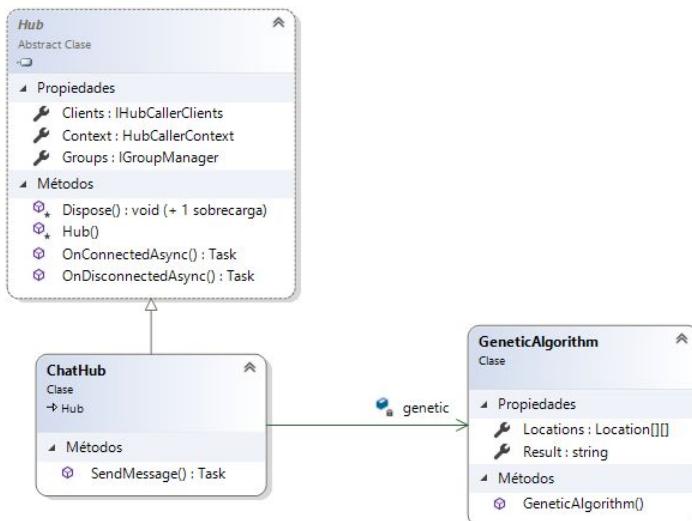
3.1. Diagramas de clases

Para los diagramas de clases utilizaremos la herramienta de la que dispone Visual Studio (a partir de VS 2017). Aunque no es compatible con proyectos en .NET Core se puede solventar mediante la creación manual del diagrama.

3.1.1. Aplicación web

3.1.1.1. CLASE ALGORITHM GENETIC Y CHATHUB

La clase **GeneticAlgorithm** es la encargada de gestionar la simulación de los algoritmos genéticos, encargándose de gestionar el tipo de búsqueda y la selección que se va a realizar. A su vez esta se relaciona con **ChatHub**, esta clase se encarga de crear un websocket para tener comunicación a tiempo real entre cliente y servidor, pasando los resultados del algoritmo, al igual que recibiendo las configuraciones.



GeneticAlgorithm posee los atributos:

- Locations: son arrays de arrays de localizaciones, guardando los resultados de cada iteración para enviarlos de golpe, ahorrando consultas.
- Result: es un string que contiene la mejor posición encontrada de entre todos los resultados.

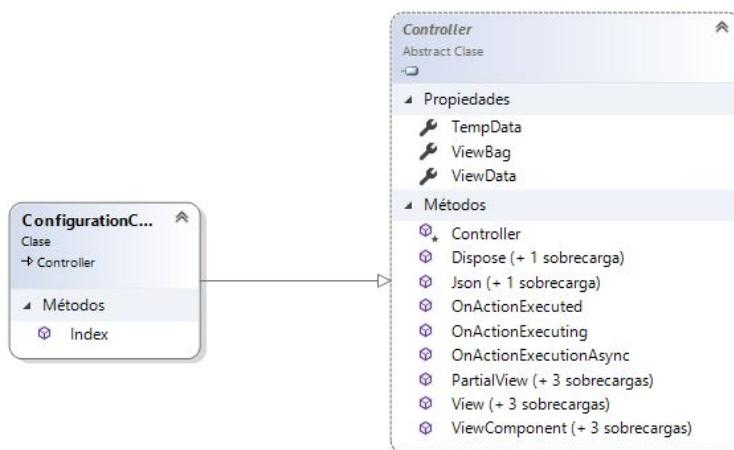
En los métodos disponemos del constructor **GeneticAlgorithm()**, que es el encargado de realizar llamar a los algoritmos genéticos con la configuración que recibe.

ChatHub posee el atributo **genetic** que es una instancia de GeneticAlgorithm.

El método que posee es SendMessage(), que se encarga de recibir la configuración del los genéticos, obtener los resultados usando genetic, y enviarlo de vuelta al cliente.

3.1.1.2. CLASE CONFIGURATIONCONTROLLER

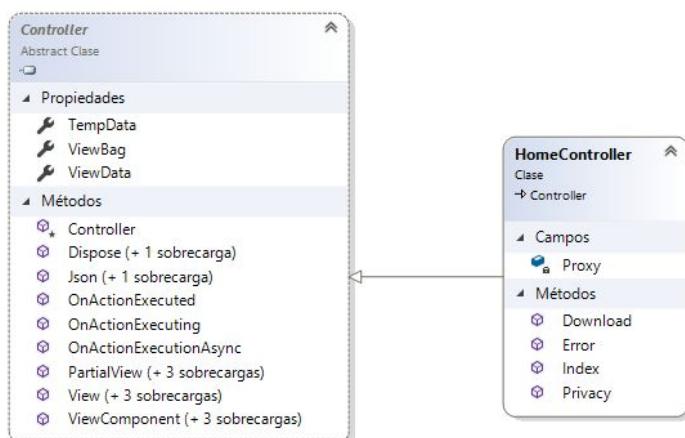
Esta clase se encarga de controlar la vista para la configuración aceptando las solicitudes y enviando las respuestas. Para ello hereda de la clase abstracta Controller.



Cuenta con el método **Index()**, que se encarga de devolver la vista al usuario.

3.1.1.3. CLASE HOMECONTROLLER

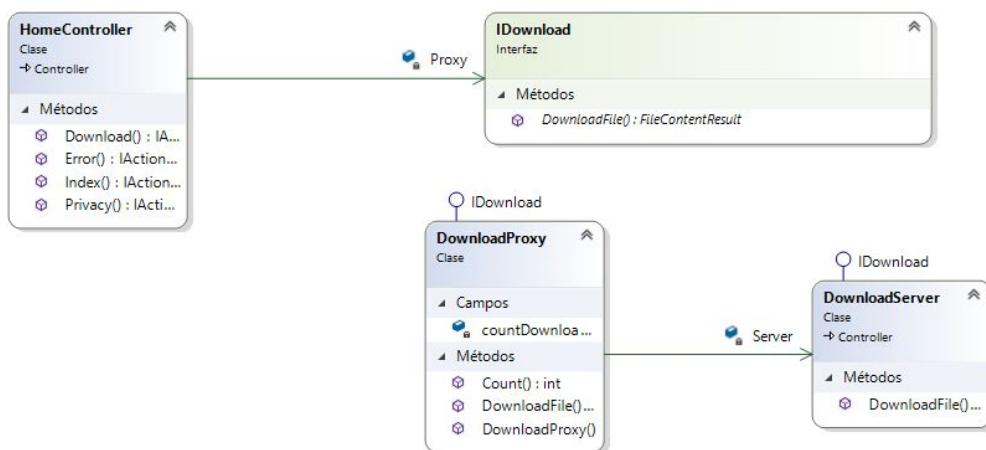
Esta clase se encarga de controlar la página principal, que es la que se muestra al inicio en la web. Para ello hereda de la clase abstracta Controller.



Cuenta con los métodos:

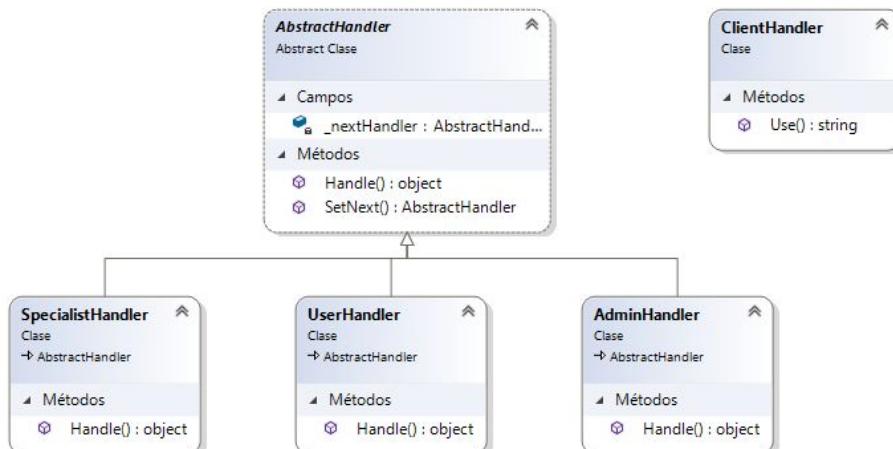
- Index: muestra la vista de la página principal al usuario.
- Download: es una acción que permite descargar un archivo.
- Error: muestra una vista de error en el caso de un fallo en la web.
- Privacy: muestra la vista que contiene el texto de privacidad.

3.1.1.4. PATRÓN PROXY



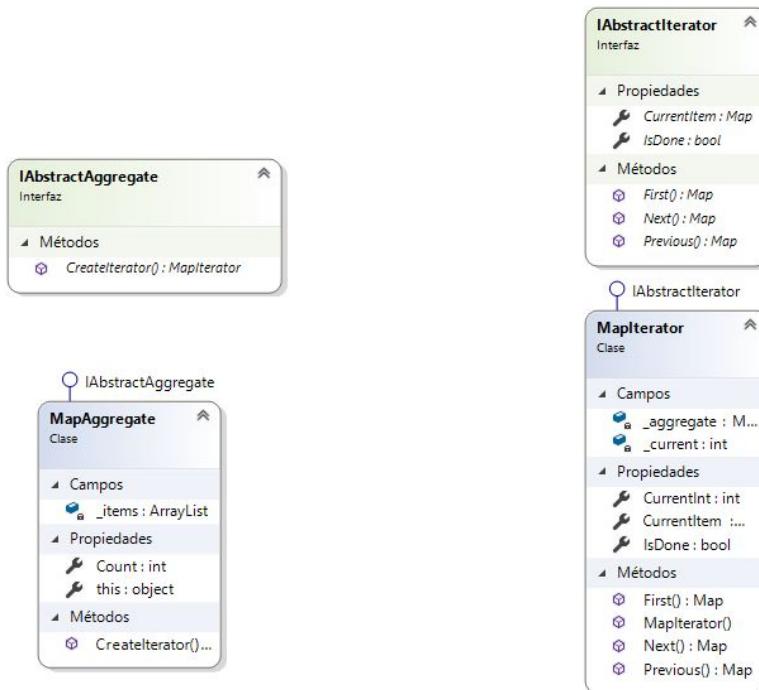
- Sujeto: **IDownload**.
- Proxy: **DownloadProxy**.
- Sujeto Real: **DownloadServer**.
- Cliente: **HomeController**.

3.1.1.5. PATRÓN CHAIN OF RESPONSABILITY



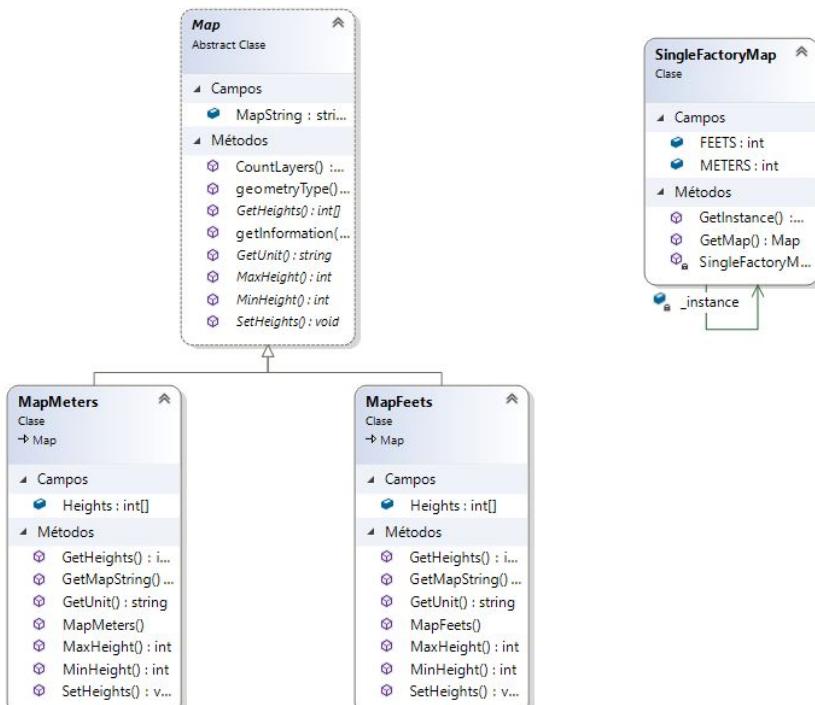
- Manejador: **AbstractHandler**
- ManejadorConcreto: **SpecialistHandler**, **UserHandler**, **AdminHandler**
- Cliente: **ClientHandler**

3.1.1.6. PATRÓN ITERATOR



- Iterador: **IAbstractIterator**
- IteradorConcreto: **MapIterator**
- Agregado: **IAbstractAggregate**
- AgregadoConcreto: **MapAggregate**

3.1.1.7. PATRÓN FACTORY-METHOD/SINGLETON/TEMPLATE METHOD



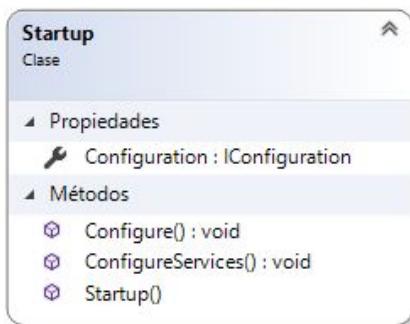
- Singleton
 - Singleton: SingleFactoryMap
- Factory
 - Producto: Map
 - ProductoConcreto: MapMeters, MapFeets
 - Creador: SingleFactoryMap
- Template method
 - Plantilla: Map
 - PlantillaConcreta: MapMeters, MapFeets

3.1.1.8. OTRAS CLASES

Estas clases no siguen un esquema normal, puesto que son clases encargadas de controlar la base de datos, la web y la aplicación en sí. Por lo que las trataremos de forma especial.

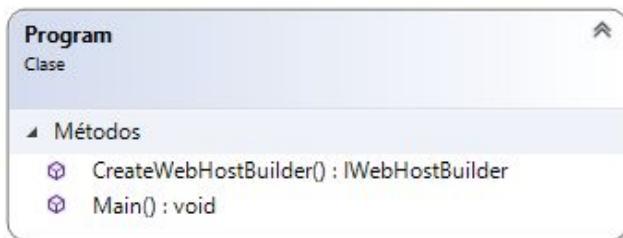
Startup

Esta clase se encarga de configurar los servicios del back-end y de la web en general, para ello dispone de los métodos **Configure()**, **ConfigureServices()** y el constructor **Startup()**.



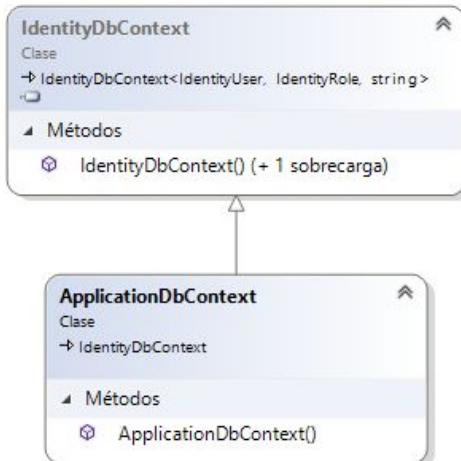
Program

Esta clase es el punto de entrada al proyecto, dispone del método **Main()** y de otro método llamado **CreateWebHostBuilder()**, que llama a la clase Startup para configurar y ejecutar la web.



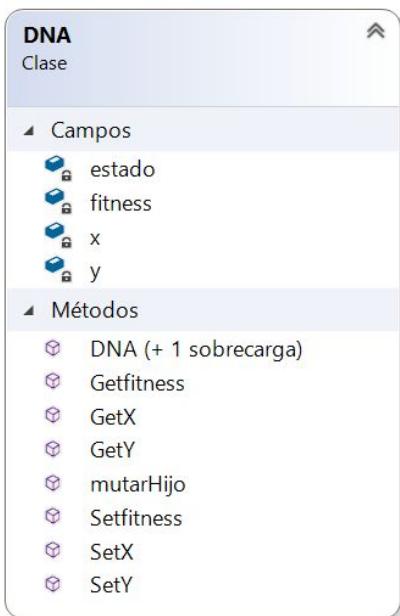
ApplicationDbContext

Esta clase gestiona el contexto de la base de datos, de tal forma que permite conectarse a la base de datos, obtener los datos y mapear dichos datos en las clases que define para los usuarios, de tal forma que puedan ser usados por el resto de clases. Se trata de una clase generada con Scaffolding y que hereda de la clase IdentityDbContext de la que obtiene sus opciones.



3.1.2 Módulo de algoritmos genéticos

3.1.2.1. CLASE DNA



Clase que representa un individuo de la población

Cuenta con los atributos:

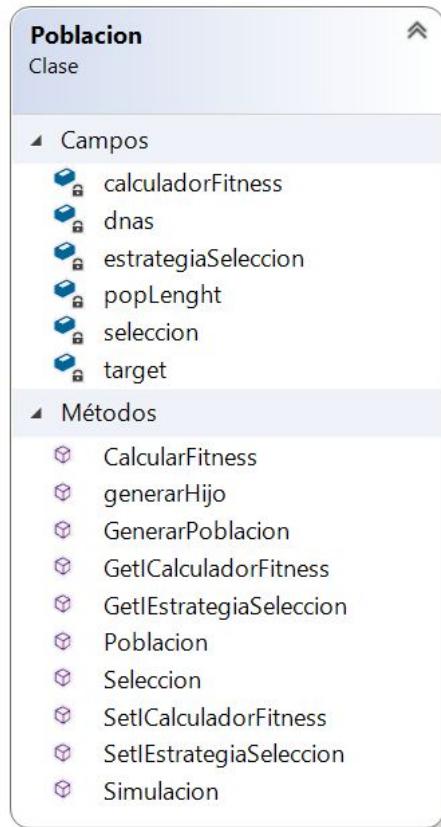
- X: indica la coordenada x en el geojson
- Y: indica la coordenada y en el geojson
- Fitness: indica el grado de aptitud del individuo

Cuenta con los métodos:

- `mutarHijo`: modificar los parametros x e y de manera aleatoria en un rango.

3.1.2.2. CLASE POBLACIÓN

Clase que representa un conjunto de individuos y realiza la simulación del paso del tiempo en el algoritmo genético. Esta clase ademas sirve como fachada para poder acceder a todo el subsistema de los algoritmos genéticos



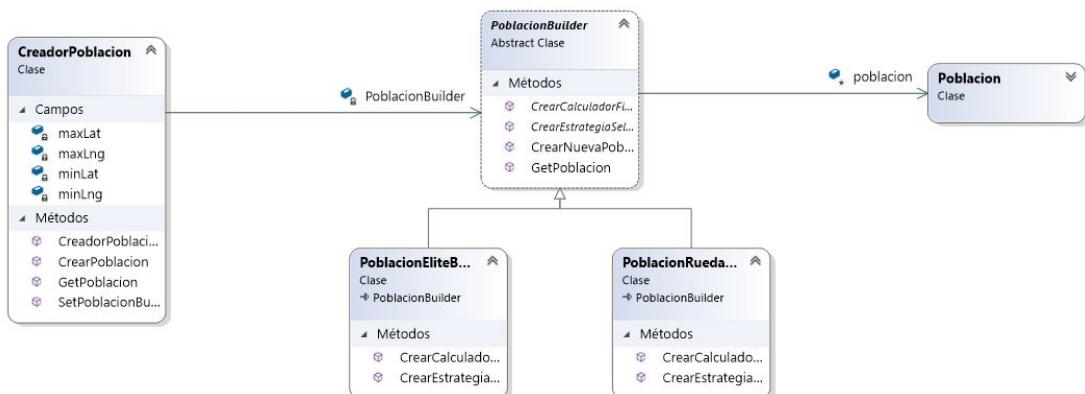
Cuenta con los atributos:

- Dnas: array con los individuos de la actual generación
- seleccion: array con los individuos que han sido seleccionados para reproducirse en la siguiente generación
- calculadorFitness: Objeto que se encarga de calcular la fitness de cada individuo
- estrategiaSeleccion: Objeto que se encarga de crear la array de selección utilizando una de las distintas estrategias
- PopLenght: el número de individuos que contendrá el array de Dnas

Cuenta con los métodos:

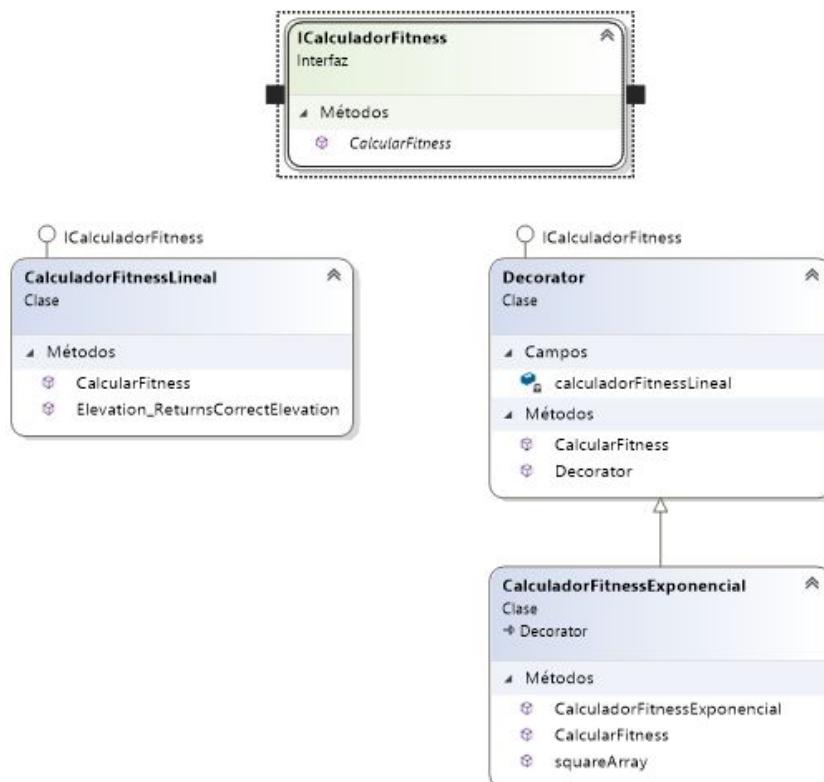
- Simulación: Proceso para simular una generación siguiendo estos pasos:
 - Calcular la fitness de los individuos mediante el proceso calcularFitness()
 - Crear una matriz de reproducción mediante el proceso Seleccion()
 - crear la nueva población combinando y mutando individuos de la matriz de selección mediante el proceso GenerarPoblacion();

3.1.2.2. PATRÓN BUILDER



- Constructor: PoblacionBuilder
- Constructor concreto: PoblacionEliteBuilder, PoblacionRuedaRuletaBuilder
- Producto: Poblacion
- Director: CreadorPoblacion

3.1.2.3. PATRÓN DECORATOR



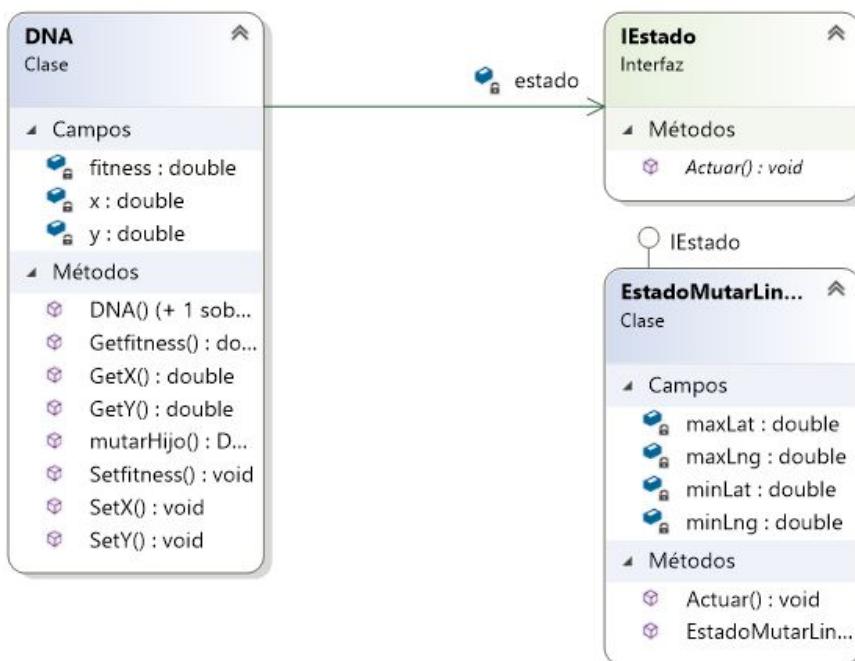
- Componente: ICalculadorFitness
- Componente concreto: CalculadorFitnessLineal
- Decorador: Decorator
- Decorador Concreto: CalculadorFitnessExponencial

3.1.2.4. PATRÓN STRATEGY



- Contexto: Poblacion
- Estrategia: ICalculadorFitness
- EstrategiaConcreta: EstrategiaSeleccionElite, EstrategiaSeleccionRuedaRuleta

3.1.2.5. PATRÓN STATE

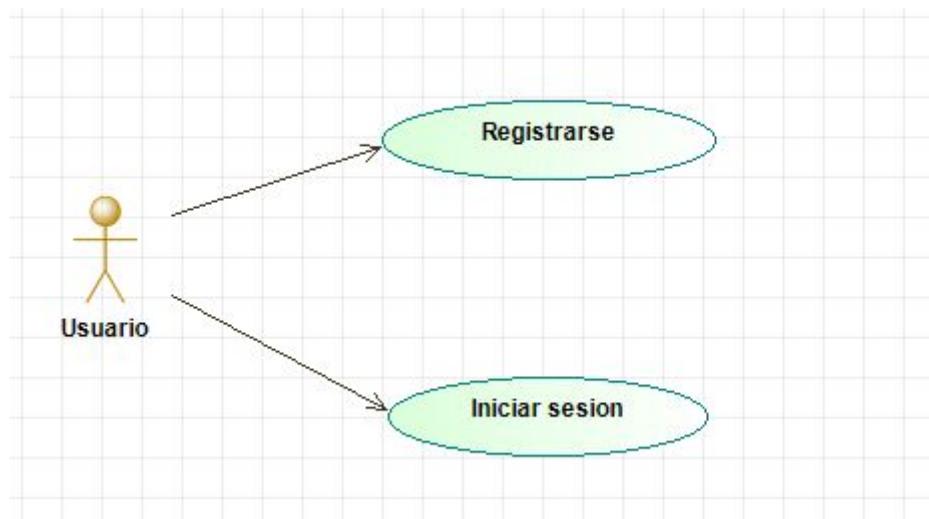


- Contexto: DNA
- Estado: IEstado
- EstadoConcreto: EstadoMutarLineal

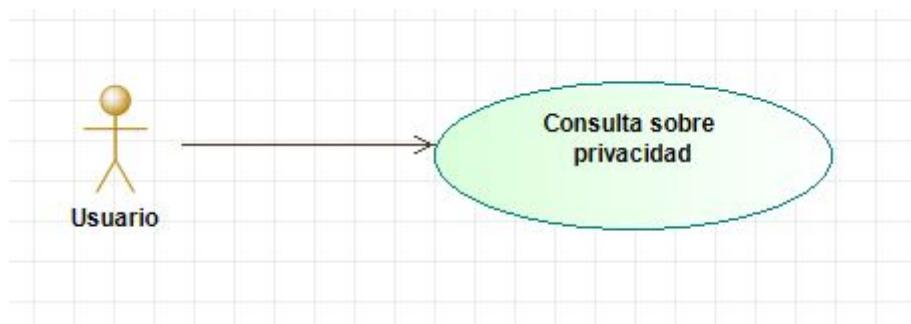
3.2. Diagramas de casos de uso

3.2.1. Aplicación web

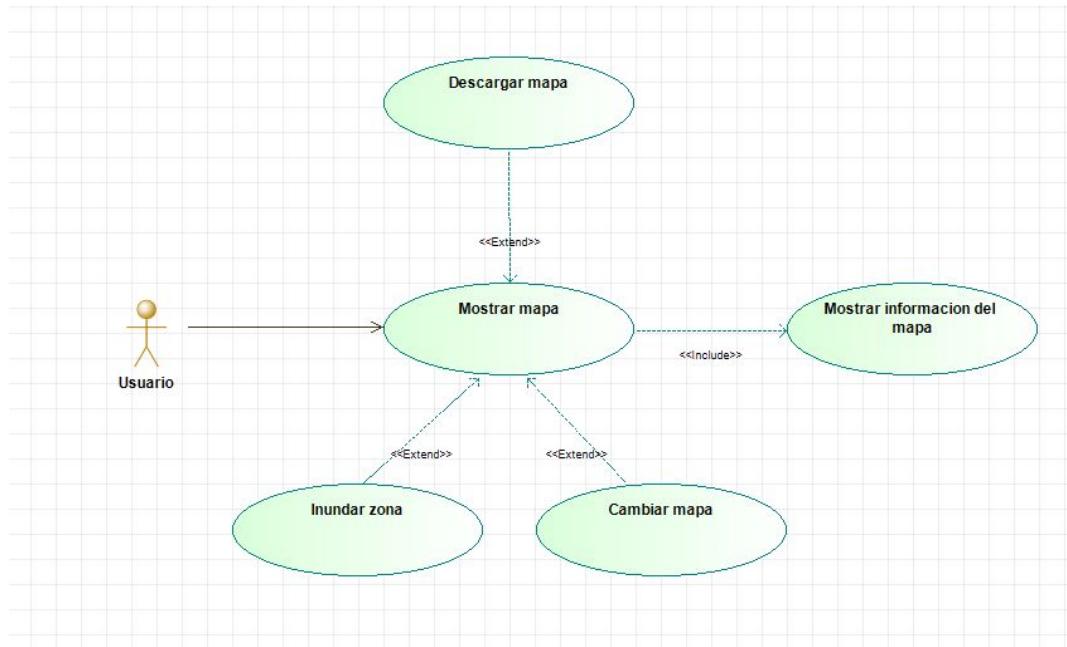
3.2.1.1. CASO DE USO AUTENTIFICACIÓN



3.2.1.2. CASO DE USO CONSULTA DE PRIVACIDAD

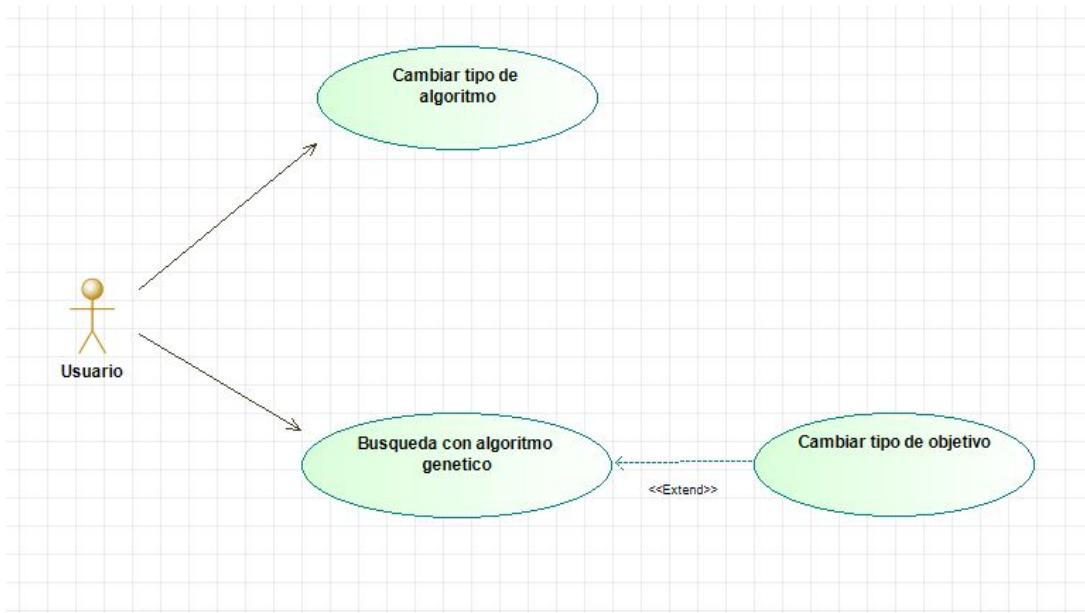


3.2.1.3. CASO DE USO ITERAR MAPA



3.2.2. Módulo de algoritmos genéticos

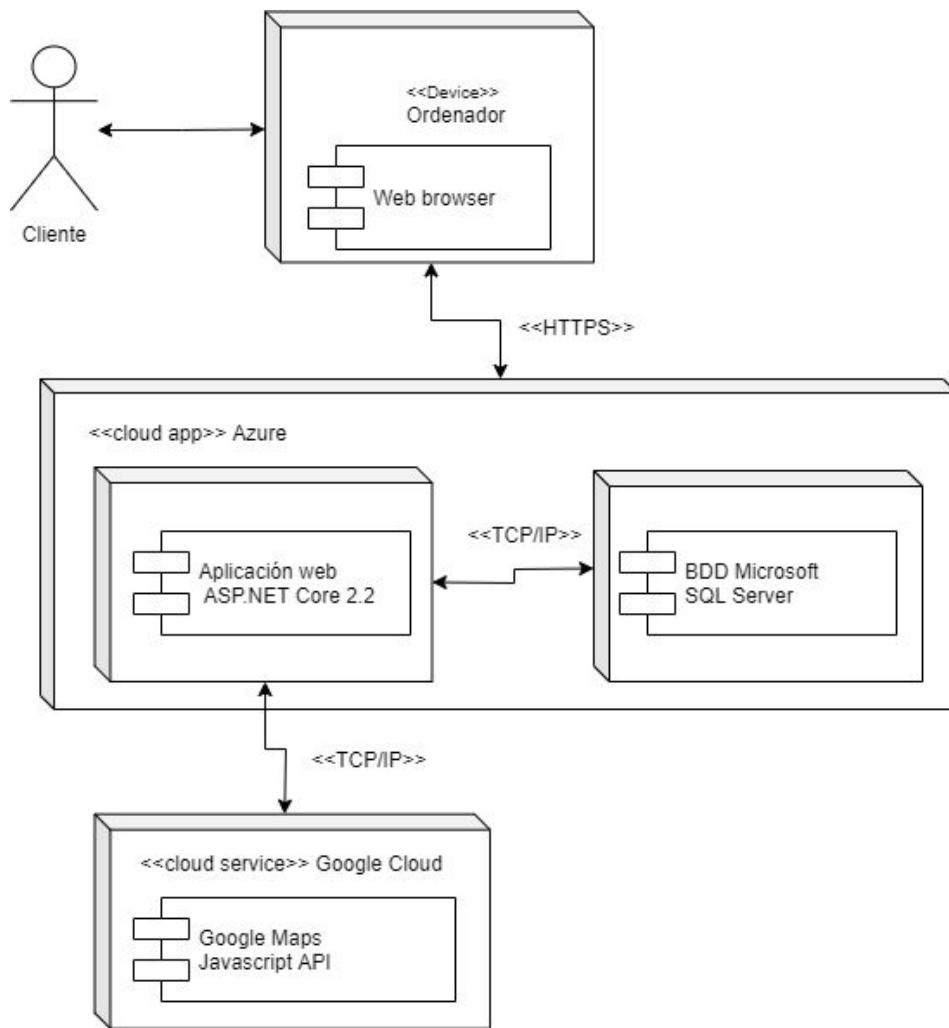
3.2.2.1. Caso de uso búsqueda de punto de presa



3.3. Diagramas de despliegue

La aplicación web se caracteriza por tener todo su despliegue en la nube. Esto sigue el motivo de la escalabilidad de la misma, de tal manera que al hacer el despliegue usando servicios en la nube es muy sencillo escalarla pagando únicamente por lo que se consume, siendo mucho más barato que si tuviéramos que mantener nuestros propios servidores. La

distribución que sigue la aplicación, como se puede observar en el diagrama, sigue un esquema basado en módulos. El principal módulo es la aplicación web desarrollada con ASP.NET Core 2.2 empleando dentro MVC, este módulo es el nexo al que se conectan los clientes mediante su navegador, la base de datos y otros servicios externos que se emplean. Dentro de la nube de Azure además de tener la aplicación web, se encuentran las bases de datos de Microsoft SQL Server , que almacenan los datos de registro, conectadas a dicha aplicación. Por último, se encuentra el servicio externo que nos proporciona los mapas que es la API de Google Maps para Javascript, siendo una parte importante del sistema.

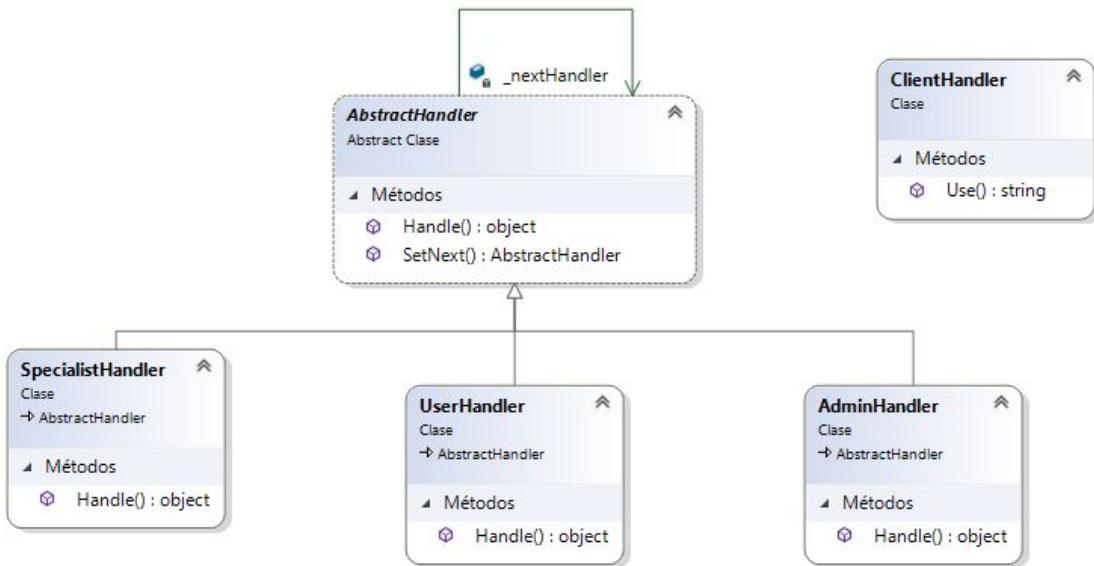


3.4. Patrones

3.4.1. Aplicación web

Chain of responsibility:

Desacoplar el emisor y el mensaje del objeto que presta el servicio. Para conseguir esto se establece una cadena en el sistema de forma que el emisor envía un mensaje al primer elemento de la cadena, este lo procesa si puede, y si no, lo redirige a otro objeto de la cadena hasta que uno pueda procesarlo



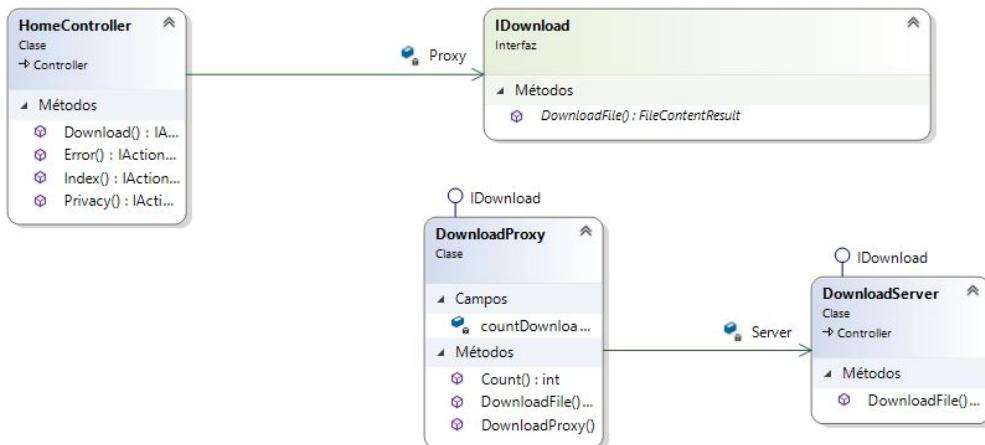
Implementación:

En este caso se implementan 3 clases correspondientes a manejadores concretos, de tal forma que cada uno define su propio método **Handle**, para determinar qué acciones puede realizar el usuario. Estos 3 manejadores heredan de la clase abstracta **AbstractHandler** que aquí contiene el método abstracto **Handle()** y un método **setNext()** que sirve para ir encadenando manejadores. Finalmente el cliente llama a los manejadores con el método **Use()**.

Razones de uso del patrón MVC:

En el sistema se tienen peticiones que deben de ser tratadas de forma independiente sin saber el usuario que lo recibe hasta el momento de ejecución, es por eso que es necesario aplicar esta cadena para saber si el usuario receptor es el adecuado.

Proxy:



MVC:

Se trata de un patrón que nos permite separar la lógica de control , la lógica de negocio de la aplicación y la lógica de presentación.

Implementación:

La aplicación completa se basa en el uso de este patrón de tal forma que se organizan las distintas lógicas de forma modular. En este caso tenemos **HomeController** y

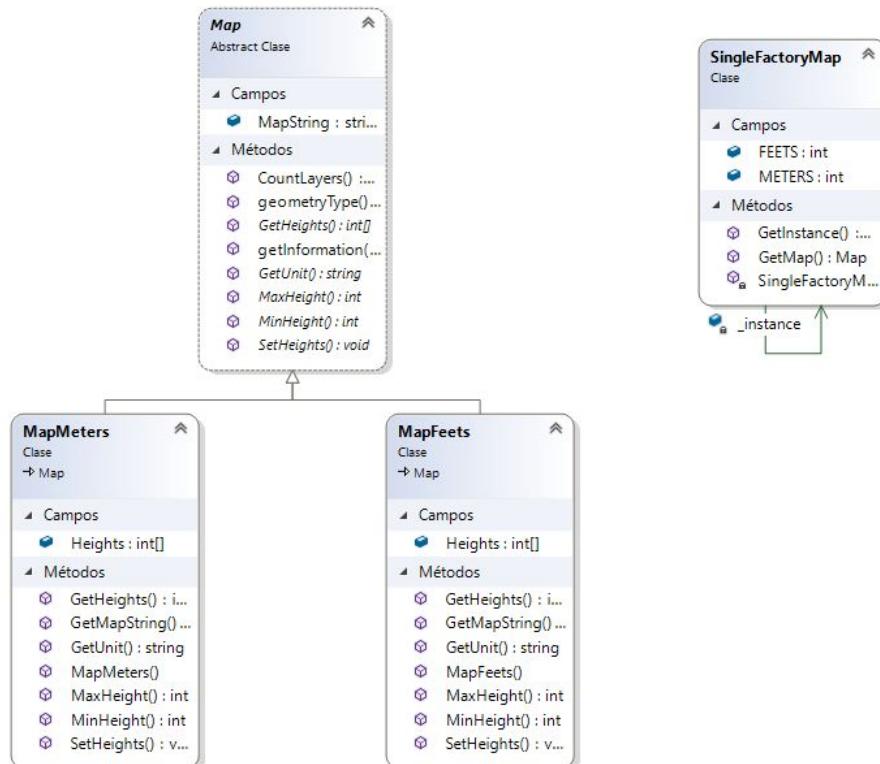
ConfigurationController que se encargan de la lógica de control. Después tenemos la lógica de negocio dentro de la carpeta BLL y Genetic, que se encargan de el funcionamiento de la aplicación y por últimos las vistas que se guardan en Views como index.cshtml, estas contienen funcionalidades que se declaran en la misma vista para agregar funcionalidad a la hora de mostrar los datos como pueden ser los mapas en Javascript.

Razones de uso del patrón MVC:

El uso de este patrón es que en este caso es ideal para gestionar la aplicación de la que disponemos, al tener una complejidad alta en la lógica de negocio al igual que en la de presentación, consiguiendo así un código mantenable, reusable y bien separado.

Factory:

Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación del objeto



Implementación:

En nuestro sistema existen 2 tipos de mapas: Mapas cuyas unidades están en metros y mapas cuyas unidades están en pies, estas clases heredan una serie de métodos y atributos de la clase abstracta Map. El método getMap() nos devolverá una instancia de estas subclases, este método lo implementará la clase creadora SingleFactoryMap

Razones de uso del patrón Factory:

Se desea crear un framework extensible, ya que queremos aumentar el tipo de mapas que queremos implementar

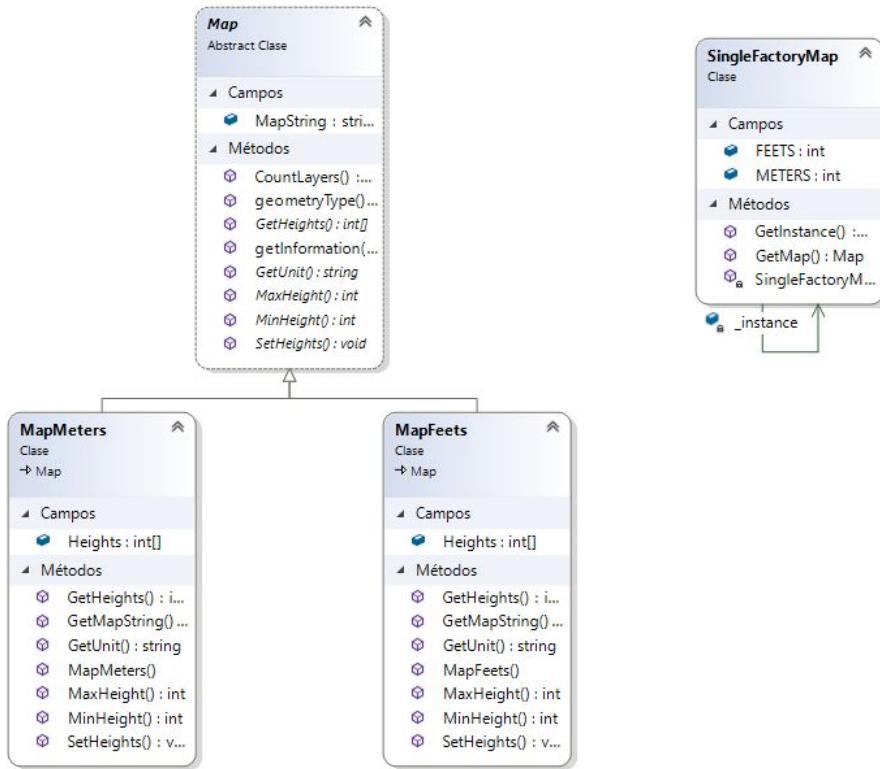
Relaciones con otros patrones:

El creador SingleFactoryMap es una clase que implementa el patrón singleton

El producto map implementa el patrón template method

Template Method:

Proporcionar un método que permita que las subclases redefinan partes del método sin reescribirlo.



Implementación:

La clase **map** implementa el método `getInformation()` que muestra un mensaje estándar (`geometryType()`), al que luego las subclases agregan más información (`GetUnit()`).

Razones de uso del patrón Template Method:

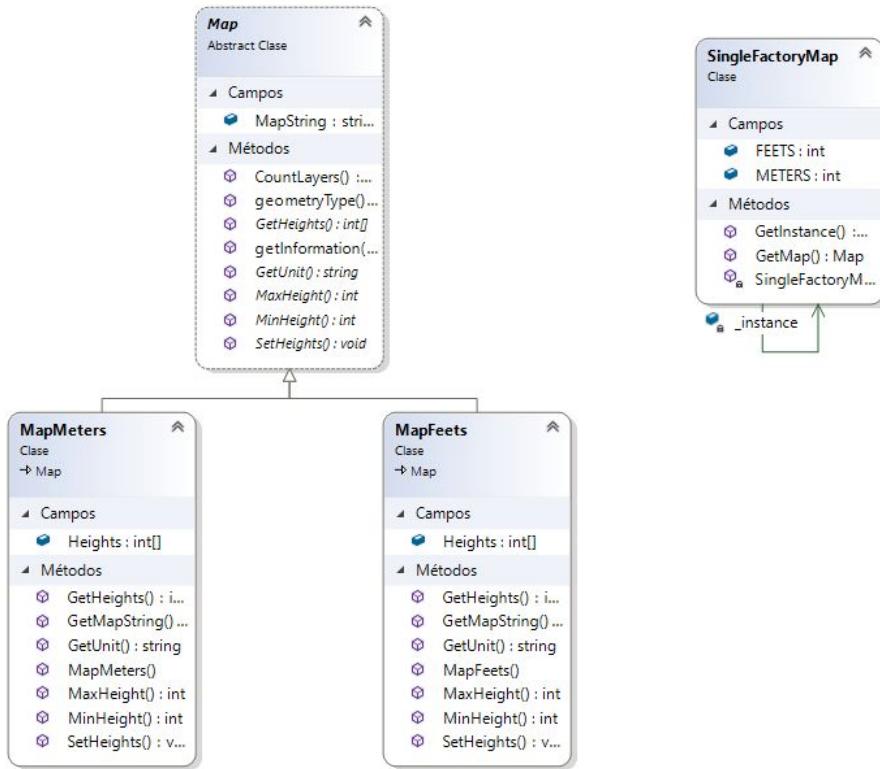
Queremos evitar la duplicación de código entre clases de una jerarquía.

Relaciones con otros patrones:

Estas clases se crean mediante un patrón factory

Singleton:

Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Todos los objetos que utilizan una instancia de esa clase usan la misma instancia.



Implementación:

Creamos un constructor privado y un método `getInstance()` que devuelve la única instancia que puede haber de la clase

Razones de uso del patrón Singleton:

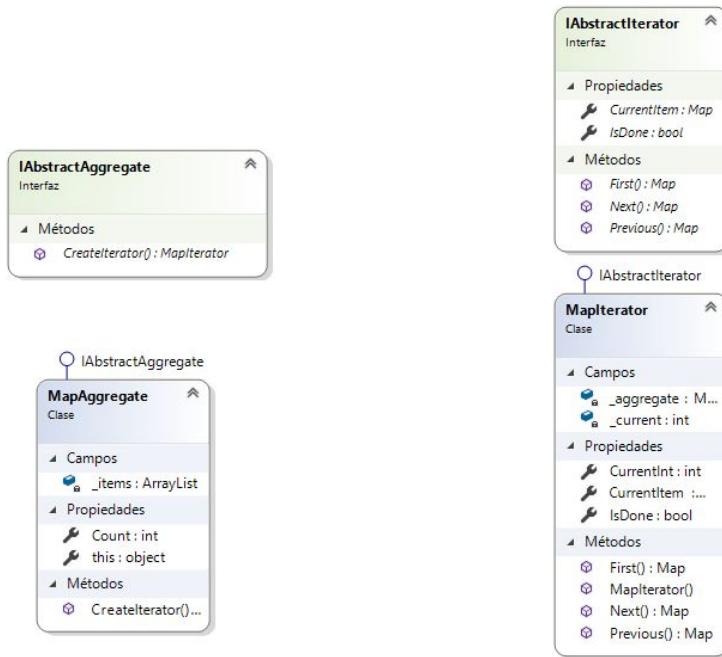
Sólo puede haber una instancia de una clase, y debe ser accesible a los clientes desde un punto de acceso bien conocido. Puesto que va a ser nuestro creador de mapas debe ser accesible para los clientes

Relaciones con otros patrones:

Esta clase singleton es el creador del patrón Factory

Iterator:

Proporciona una forma coherente de acceder secuencialmente a los datos de una colección, independientemente del tipo de colección.



Implementación:

Para implementar este patrón se han creado dos interfaces correspondientes al agregado y al iterador, que son **IAbstractAggregate** e **IAbstractIterator**. Estas son implementadas por las clases concreta **MapAggregate** y **MapIterator** respectivamente. El agregado implementa un método para crear iteradores, y propiedades para obtener el agregado y contar los elementos. En cuanto a los iteradores contienen métodos para moverse en las colecciones como **First()**, **Next()** o **Previous()**, además de propiedades y campos que contienen los mapas o indica si ha finalizado.

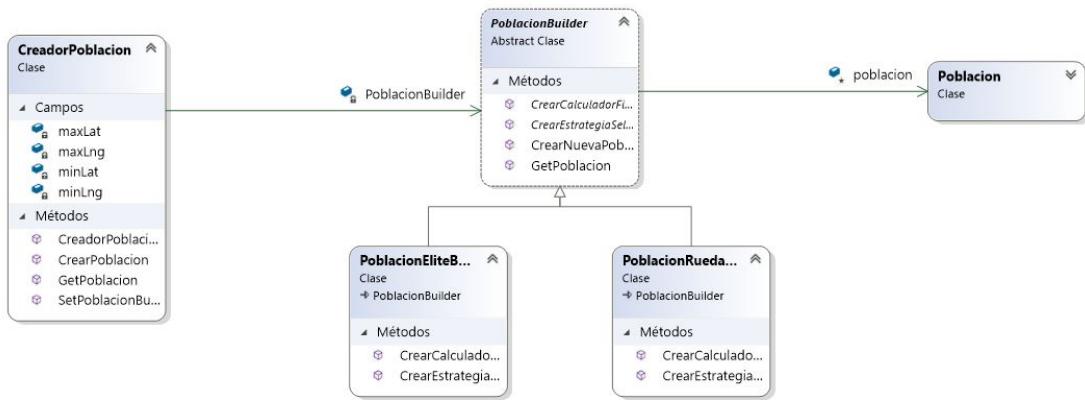
Razones de uso del patrón Iterator:

Se ha empleado para tener una forma coherente de recorrer los distintos mapas que hay disponibles de tal forma que se oculte la implementación al cliente, para que no se preocupe. Además queríamos dejarlo abierto a añadir futuras nuevas colecciones de mapas a recorrer u otros elementos.

3.4.2 Módulo de algoritmos genéticos

Builder:

Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción puede crear diferentes representaciones.



Implementación:

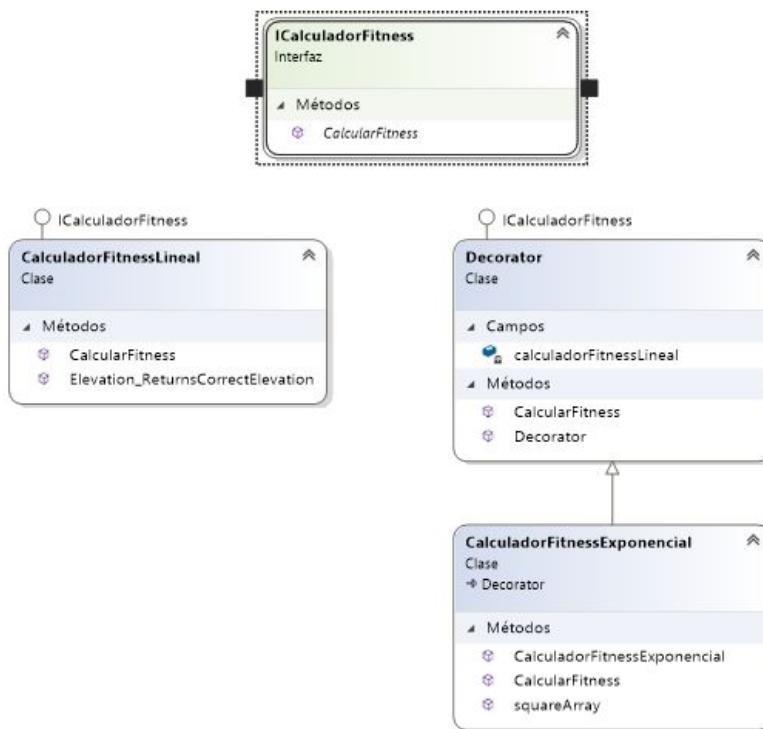
La creación de la clase población está dividida en 2 etapas, la creación del calculador de fitness y la creación de la estrategia de selección. Existen 2 builders; un builder para poblaciones Élite, que al tomar siempre la mitad de la población con la fitness más alta. La formas de modificar la fitness no afectan a este tipo de poblaciones por tanto utilizamos una fitness lineal. El otro builder utiliza poblaciones Rueda Ruleta, que está implementación utiliza probabilidades por lo tanto intentar acentuar los cambios de la fitness puede dar resultados muy positivos por tanto utilizamos el decorator fitness exponencial.

Razones de uso del patrón Builder:

La clase población tiene una estructura compleja, que está dividido en 2 etapas, la creación del calculador de fitness y la creación de la estrategia de selección.

Decorator:

Añadir nuevas responsabilidades dinámicamente a un objeto sin modificar su apariencia externa o su función, es una alternativa a crear demasiadas subclases por herencia.



Implementación:

En el sistema existen dos tipos de formas de calcular la fitness: un calculador de fitness lineal, el cual calcula la fitness de un punto sin añadir ninguna modificación; y un calculador de fitness exponencial, la cual eleva al cuadrado la fitness(esto puede ser muy útil para hacer que cualquier mejoría se incremente de manera exponencial).

La clase ICalculadorFitness es una interfaz que actúa como componente, la clase CalculadorFitnessLineal actúa como componente concreto y la clase CalculadorFitnessExponencial actúa como el decorador concreto.

Razones de uso del patrón Decorator:

Para poder añadir funcionalidad a una clase de forma dinámica en tiempo de ejecución.

Strategy:

Definir un grupo de clases que representan un conjunto de posibles comportamientos. Estos comportamientos pueden ser fácilmente intercambiados en un aplicación, modificando la funcionalidad en cualquier instante.



Implementación:

se proporcionan dos implementaciones de la interfaz Estrategia: EstrategiaSeleccionElite, incluye la implementación del método abstracto Seleccion, que dado el parámetro población(que incluye todos los individuos de la población), selecciona que individuos podrán reproducirse para la siguiente generación usando una estrategia de Élite. Esto consiste en añadir a la matriz de reproducción solamente la mitad de los individuos cuya fitness sea más alta.

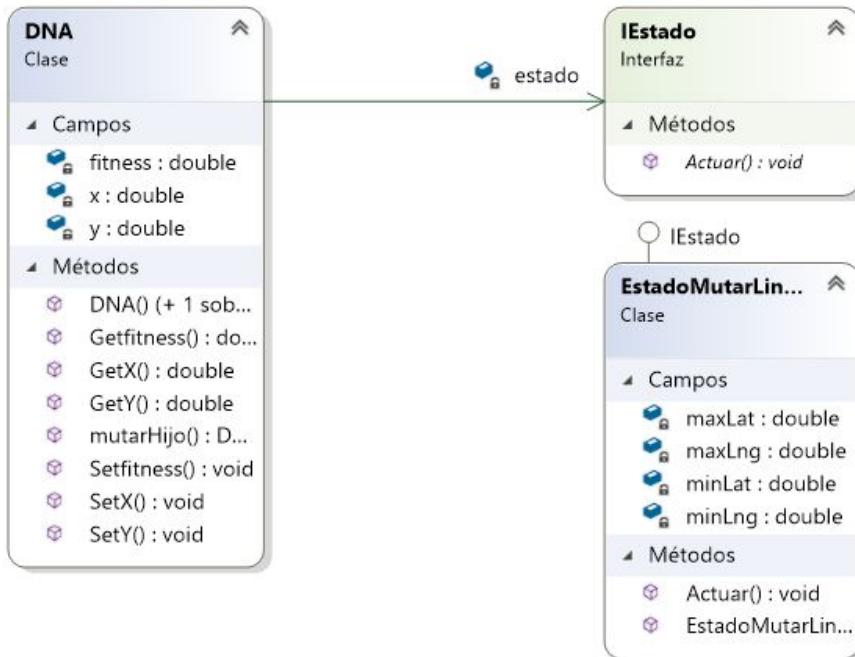
EstrategiaSeleccionRuedaRuleta, selecciona que individuos podrán reproducirse para la siguiente generación usando una estrategia de Rueda ruleta. Esto consiste en cada individuo será añadido a la matriz de reproducción un cierto número de veces. A mayor fitness tendrás más elementos en la matriz de reproducción y por tanto más posibilidades de ser elegido para reproducirse.

Razones de uso del patrón Strategy:

Como se puede observar estas clases solamente se diferencian en la manera que generan la matriz de reproducción, utilizando distintos algoritmos para obtener la misma funcionalidad.

State

Permitir que un objeto se comporte de distinta forma dependiendo de su estado interno, como si cambiase la clase a la que pertenece. Permite cambiar fácilmente el comportamiento de un objeto en tiempo de ejecución.

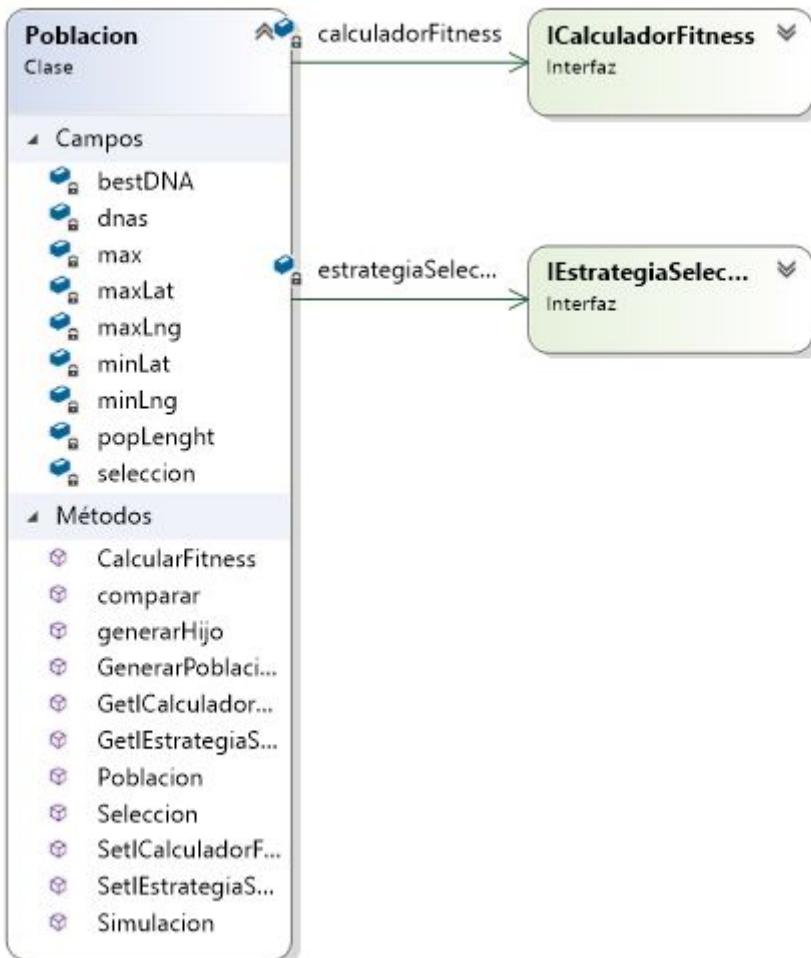


Razones de uso del patrón State:

El comportamiento del objeto DNA depende de la etapa (vida, selección, reproducción, mutación) en la que se encuentra y necesitamos cambiar su comportamiento en cada etapa.

Facade:

Simplifica el acceso a un conjunto de subsistemas o un sistema complejo. Representa una única interfaz unificada, que envuelve el subsistema, y es responsable de colaborar con los componentes del subsistema.



Implementación:

la clase Poblacion actúa como fachada, simplificando el acceso a el subsistema de los algoritmos genéticos, mediante el método simulación(), calcularFitness(), generarPoblacion(). Se accede a todas las demás clases asociadas a la clase Poblacion.

Razones de uso del patrón facade:

Se formó de manera natural al crear el código

3.5 Diagrama de secuencia

3.5.1 SECUENCIA SIMULACIÓN

