# Definición del proyecto

## Objetivo, ámbito y límites

El objetivo principal del proyecto es desarrollar una aplicación web que permita buscar cervecerías en la ciudad de Gijón dada la ubicación del usuario. Esta búsqueda podrá ser acotada por él, al que se le permitirá interactuar con diferentes filtros:

* Un slider cuyos extremos sean “calidad de la cerveza” y “cercanía del local”.
* También se ofrecerá la posibilidad de especificar el tipo y marca de las cervezas buscadas.

Además de interactuar con la configuración de búsqueda el usuario podrá valorar un establecimiento, incluyendo una descripción y una nota entre 1 y 5 puntos, siendo 1 la valoración más baja y 5 la valoración más alta.

El usuario también podrá indicar al sistema que quiere regresar a su casa, dándose diferentes comportamientos dependiendo de si el sistema conoce esa información:

* El sistema no conoce donde vive el usuario: en este caso se consultará la parada de taxis más cercana y se le indicará como llegar hasta ella.
* El sistema sabe dónde vive el usuario (el usuario es un usuario registrado y ha incluido donde vive): en este caso se le indicará tanto la parada de taxis más cercana como el autobús que le deje más cerca de su casa.

## Actores del sistema

* Usuario no registrado: podrá valorar un establecimiento y registrarse en el sistema.
* Usuario registrado: podrá valorar un establecimiento, incluir información sobre la ubicación de su vivienda, consultar las valoraciones que ha realizado y editarlas.
* Gerente de un local: podrá incluir una descripción de su establecimiento, el horario en el que permanece abierto y un catálogo con las cervezas que se sirven con sus precios.

## Servicios utilizados

Para el desarrollo del proyecto se utilizarán los siguientes servicios:

* Taxis: para consultar la información de los taxis se utilizará la API <https://observa.gijon.es>, que cuenta con más de 700 conjuntos de datos abiertos
* Servicio propio: se creará un servicio propio con un base de datos local donde se mantendrá la información referente a las cervecerías y las votaciones.
* Zonas verdes: como servicio WMS externo se han escogido las zonas verdes de Gijón (las diferentes capas que ofrece el servicio se pueden consultar en la siguiente dirección http://ide.gijon.es/visor/)
* Autobús urbano: en este caso se utilizará la combinación de los siguientes servicios, <http://docs.gijon.es/sw/busgijon.asmx> para la información sobre las paradas y <http://datos.gijon.es/doc/transporte/busgijontr.json> para conocer las posiciones de los autobuses en tiempo real.

# Prototipo

## Funcionalidades implementadas

Pero todo esto tan solo ha sido la idea original, que se va a simplificar para el desarrollo del prototipo:

* El usuario no registrado podrá valorar una cervecería, pero no registrarse, y utilizar el slider de preferencia de calidad vs cercanía.
* En cuanto a los servicios utilizados tan solo se va a excluir el referente a los autobuses urbanos, por lo que se desarrollará el servicio propio, el servicio de taxis consultando la API externa y el servicio WMS de las zonas verdes de Gijón.

## Diseño

### Arquitectura general

El desarrollo del proyecto se va a dividir en dos servicios separados:

* Una página web que se conectará con la API de Google Maps, cuyo mapa vamos a utilizar como elemento base del proyecto y sobre el que se colocarán la capa recibida por el servicio WMS y los marcadores, los cuales nos serán proporcionados por el otro servicio
* Un servicio propio, encargado de realizar las conexiones a la base de datos para actualizar los componentes y a los servicio externos para informar al servicio web.

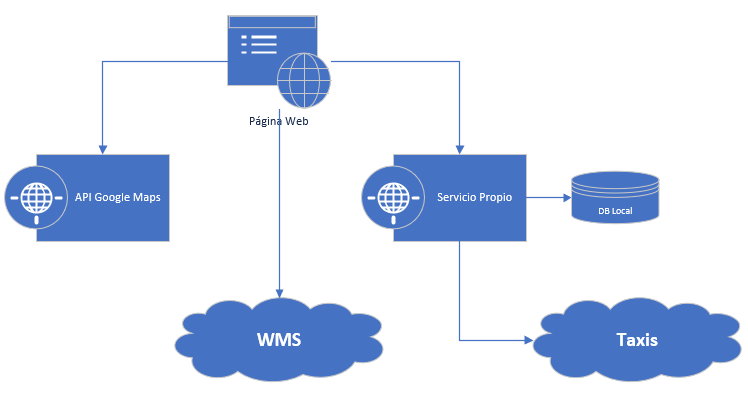


Ilustración 1 Diagrama de componentes

### Arquitectura página web

La aplicación principal que consume los servicios se ha realizado en NodeJS, con el motor de platillas ejs.

La aplicación está formada por una única vista donde encontraremos las diferentes opciones: mostrar taxis y filtrar los bares en función de distancia o calidad de cerveza.

Para mostrar las funcionalidades hemos utiliza la API de Google Maps, donde junto con el mapa hemos utilizado marcadores para mostrar las ubicaciones de los bares o los taxis.

Toda esta funcionalidad está implementada en un fichero JavaScript donde hemos separado por funciones cada parte. Por un lado, tenemos la inicialización del mapa, crear y eliminar las ubicaciones; y por otro lado tendremos funciones con las llamadas Ajax a los servicios de obtener los json con los taxis o los bares.

### Arquitectura servicio propio

Realizado en C# utilizando un servicio MVC de Microsoft con métodos restfull en .NetFramework, pone a disposición de la interfaz varios métodos que comentaremos más adelante. Como podemos ver en la Ilustración 1 Diagrama de componentes Nuestro servicio web hace de punto intermedio entre la base de datos, la información de las paradas de taxi y nuestra interfaz.

Hablemos primero de la información sobre paradas de taxis. Gracias a la política opendata de Gijón, se puede obtener de la siguiente [URL de taxis](https://observa.gijon.es/api/records/1.0/search/?dataset=paradas-de-taxis&rows=-1&facet=parada) toda la información relevante de las paradas de taxis que servirá para alimentar la base de datos de ubicaciones de nuestra aplicación. La configuración del servicio permite que en el momento del despliegue se configure la base de datos local utilizando **SQL server 2017** para que se inicialice con unos datos de configuración básicos que hemos configurado, entre estos datos se utiliza una conexión http para realizar una petición sobre la url antes nombrada para así obtener toda la información necesaria sobre paradas. Esta actualización puede configurarse para realizarse automáticamente en busca de modificaciones.

Referente a la parte de gestión de Pubs, para el prototipo ya disponemos de un servicio restfull que permite tanto la obtención de todos los marcadores disponibles como la interfaz de modificación de votaciones. Estos servicios quedarán expuestos y preparados para que la interfaz pueda consumirlos una vez desplegado en la máquina destino.

#### Métodos expuestos

En primer lugar, explicaremos los métodos Get disponibles para la obtención de marcas, estos serán accedidos mediante las siguientes URL:

* */api/Pub*
* */api/Taxi*

Para la inclusión de nuevas peticiones disponemos de un método POST que se atiende en */api/Pub/<id del bar>* enviando en el body en formato JSON el voto que se desea subir.

De cara al despliegue, como los proyectos desarrollados en visual studio, permiten publicar y dar como resultado una carpeta que desplegaremos sobre el IIS destino. En la estructura de ficheros, dispondremos de un *web.config* donde podremos realizar ciertas configuraciones, como por ejemplo cambiando la cadena de conexión, podríamos utilizar una base de datos propia a diferencia de la base de datos local MDF preparada dentro de la carpeta App\_Data del proyecto. Como se ha utilizado EntityFramework como interfaz ORM, una vez comenzado el despliegue se realizarán las tareas iniciales de configuración de la base de datos, tales como la inicialización de la semilla y la descarga de Taxis de Gijón.

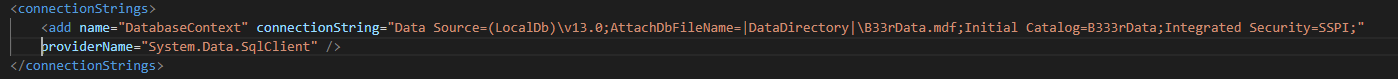


Ilustración 2. Cadena de conexión

### Formato de los marcadores

A continuación, se define el formato JSON que van a seguir los marcadores con los que van a trabajar los dos servicios. Cabe destacar que no se han utilizado todos los elementos proporcionados por el servicio de taxis, ya que tan solo nos hemos quedado con los elementos que tenían más representación para el proyecto.

|  |  |
| --- | --- |
| Bar | Taxi |
| {  "guid": "Guid",  "name": "Nombre del bar",  "location": {  "latitud": 12341234,  "longitud": 123412341234  },  "imageUrl": "image location",  "votes": [  {  "value": 4,  "msg": "Mensaje"  }  ]  } | {  "recordId": "Guid",  "fields": {  "latitud": 12341234,  "longitud": 123412341234,  "parada": "Calle de la parada"  }  } |

### Pantallas

Para el análisis del prototipo se han diseñado tres prototipos de pantalla: la pantalla principal con las cervecerías marcadas en el mapa, la pantalla de “llévame a la parada de taxis más cercana” con la ruta a seguir y el popup para votar y comentar al seleccionar una cervecería.

|  |  |  |
| --- | --- | --- |
| Ilustración 3 Principal | Ilustración 4 taxi | Ilustración 5 vota |

## Uso de la API

### Taxis

Para poder acceder a toda la información de los taxis de Gijón se utilizará la página <https://observa.gijon.es/explore/dataset/paradas-de-taxis/api/>, la cual nos proporcionará los datos de configuración de la consulta que queremos realizar para consumir los datos de la API.

Según están los parámetros nos saldrían 10 paradas de taxis, pero como queremos la información de todas las paradas tendremos que modificar el parámetro rows para que sea -1.

Al enviar la petición nos llevará a la siguiente página https://observa. gijon.es/ api/records/1.0/search/?dataset=paradas-de-taxis&rows=-1&facet=parada en la que se podrán consultar todos los datos.



Ilustración 6 JSON DEVUELTO POR LA API DE PARADAS DE TAXIS

### Google Maps

Para mostrar los datos obtenidos se ha utilizado la API de Google Maps sin KEY. Para ello, hemos incrustado en la vista el script de la api y hemos inicializado el mapa en una función en nuestro fichero JavaScript.

Una vez echo esto, se ha modificado los estilos del mapa de forma que casi ocuparan el 100% de esta.

Por otro lado, se ha utilizado Markers para añadir elementos al mapa como por ejemplo la ubicación de los bares o de las paradas de taxis, siendo estas últimas en la cuales se ha realizado un icono personalizado para ellas.

### WMS

Para incluir la capa de información proveniente de un servicio WMS estándar externo se ha escogido el servicio <http://ide.gijon.es/geoserver/gwc/service/wms>?, y más concretamente la capa de zonas verdes de Gijón. La información referente a la ubicación en el mapa devuelta por este sistema se encuentra en formato UTM, por lo que ha sido necesario transformar los elementos que se iban a incluir en el BBOX a este formato.

## Lenguajes y metodologías utilizadas

### C#

|  |  |
| --- | --- |
| Ilustración 7 logo c# | Es un lenguaje de programación orientado a objetos desarrollado por Microsoft y que apareció por primera vez en el 2000. Su sintaxis fue creada siguiendo el modelo de C y C++ para que fuera superior a ellos y, aunque utiliza el modelo de objetos de .NET, se le considera un lenguaje de programación independiente que puede generar aplicaciones en múltiples plataformas. Aunque la última versión de este lenguaje sea la 7.2, de noviembre de 2017 se ha utilizado la Microsoft.CSharp/X.X.X |

### CSS y HTML

|  |  |
| --- | --- |
| CSS, o Cascading Style Sheets, es un lenguaje que se utiliza en páginas web para definir estilos. La principal ventaja de este lenguaje es que se puede incluir mediante ficheros externos y referenciarlo al html mediante etiquetas style.  HTML, también conocido como HyperText Markup Language es un elemento de construcción de páginas web para crearlas y representarlas visualmente, determinando el contenido, pero no la funcionalidad. | Ilustración 8adsf  Ilustración 9 logo css y html5 |

### JavaScript

|  |  |
| --- | --- |
| Ilustración 10 logo javascript | JavaScript es un lenguaje de programación interpretado utilizado en el lado del cliente, que permite realizar actividades complejas en páginas web, consiguiendo que éstas puedan cambiar dinámicamente. |

### Node.js

|  |  |
| --- | --- |
| Node.js, es un entorno JavaScript del lado del servidor basado en eventos, utilizando el motor V8 de Google que permite a Node compilar y ejecutar JavaScript a mucha velocidad. Es una librería de código abierto que fue creada para crear programas de red altamente escalables. | Ilustración 11 logo node.js |

### Entity Framework

|  |  |
| --- | --- |
| Ilustración 12 logo entity framework | Es un conjunto de tecnologías para desarrollar aplicaciones orientadas a datos de ADO.NET. Esta tecnología permite unir la gestión de objetos y la gestión de datos, haciendo que se pueda trabajar con datos en forma de objetos, permitiendo al desarrollador trabajar en un nivel mayor de abstracción al no tener que preocuparse por las tablas y transacciones con la base de datos. |

# Despliegue de la aplicación

# Referencias

Autobuses urbanos Gijón: <http://docs.gijon.es/sw/busgijon.asmx>

Autobuses urbanos Gijon, tiempo real: <http://datos.gijon.es/doc/transporte/busgijontr.json>

Balsamiq Mockup: <https://balsamiq.com/>

Json Editor Online: <https://jsoneditoronline.org/>

Maps JavaScript API: <https://developers.google.com/maps/documentation/javascript/tutorial>

Paradas de Taxis de Gijón: <https://observa.gijon.es/explore/dataset/paradas-de-taxis/?flg=es>

Visio: <https://products.office.com/es-es/visio/flowchart-software>

Visor IDE Gijón: <http://ide.gijon.es/visor/>