TengoPlan4U

Autor: Julio Benítez Casas
Máster Universitario en Ingeniería Informática
Desarrollo de aplicaciones web

Consultor: Ignasi Lorente Puchades
Profesor responsable: César Pablo Córcoles Briongos

Fecha de entrega: 9 de enero de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*TengoPlan4U*
Nombre del autor:	*Julio Benítez Casas*
Nombre del consultor/a:	*Ignasi Lorente Puchades*
Nombre del PRA:	*César Pablo Córcoles Briongos*
Fecha de entrega (mm/aaaa):	01/2017
Titulación::	*Máster Universitario en Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo de aplicaciones web*
Idioma del trabajo:	*Español*
Palabras clave	*Plan, Local, Recomendación*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

La finalidad de este trabajo es desarrollar una aplicación web que ofrezca a los usuarios ideas sobre locales de ocio con los que trazar su plan en función de una serie de filtros de búsqueda. Asimismo, el usuario se verá beneficiado de recomendaciones en función de su perfil y podrá aportar sus propias opiniones una vez visite algún de los lugares obtenidos en su búsqueda.

Esta idea surge en un contexto en que cada vez es más difícil tomar decisiones entre la multitud de opciones disponibles y, en el caso concreto de los locales de ocio y gastronómicos, ante la falta de concreción de los sitios existentes en Internet para ofrecer búsquedas específicas.

Para plasmar esta idea, se hace uso del *framework* .NET y el patrón de diseño Modelo Vista Controlador (MVC) de Microsoft, buscando como resultado una aplicación web usable tanto en ordenadores como en dispositivos móviles (diseño *responsive*). La aplicación consumirá además algunos servicios web de Google para obtener de los locales información de localización y otros detalles.

En conclusión, se trata de una aplicación de gran utilidad en los tiempos actuales, especialmente, en ciudades grandes en las que las distancias son mayores y el tiempo escaso para pensar lugares dónde ir en tu tiempo libre o, simplemente, pasear con tranquilidad para descubrir lo que realmente te gusta.

Abstract (in English, 250 words or less):

The goal of this project is to develop a web application that will be able to make a list of places based on a set of search parameters, which will be useful to users to make their own plans. In addition, the user will be notified with

recommendatios based on his profile and he will be able to register his opinions about places once he has visited any of them.

This idea is born in a context of difficult decissions, due to the multiple available choices we have every time we want to decide something. Also, regarding food and leisure places, current Internet sites dedicated to this topic do not offer enough specific results.

Framework .NET and MVC (Model View Controller) desing pattern of Microsoft are used to achieve this objective. The result is a usable web application both in computers and mobile devices (responsive desing). Moreover, the applitacion consumes some Google web services to get location and other details of the searched places.

In conclusion, it is a very useful application nowadays, specially in big cities, where distances are larger and people have less time to search and decide which places should be nice to visit on their free time.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Aunque a lo largo de la historia humana, el hecho de tomar una decisión entra dos o más ítems siempre ha sido un quebradero de cabeza para muchos individuos, podemos afirmar que en los últimos tiempos este hecho se ha acrecentado a niveles estratosféricos, en gran parte debido a la globalización y, sobre todo, a las nuevas tecnologías, las cuales amplían el espectro de decisión enormemente. Por ejemplo, a la hora de querer elegir un nuevo *smartphone*, las posibilidades y variables a evaluar son inacabables, convirtiéndose la elección en algo frustrante.

De igual manera, de un tiempo a esta parte y también debido a la conquista que Internet ha hecho de nuestras vidas, cada vez es más frecuente que previamente a realizar un plan de ocio gastronómico en la calle, ya sea con amigos, con la familia o en pareja, busquemos consejo en la red para decidir qué lugares visitar. En estos casos, la dinámica habitual es hacer uso de tu motor de búsqueda favorito (Google, en la mayoría de los casos, como se aprecia en las estadísticas que ofrece StatCounter[1]), y consultar los resultados que se consideren más interesantes, siendo estos en general blogs personales o webs dedicadas al turismo como Trip Advisor[2].

Respecto a este último ejemplo, es capaz de proveer una lista de restaurantes de todo tipo buscados en una ciudad o cerca de una zona concreta. Sin embargo, falta concreción en el filtrado de resultados que se ajusten a unas necesidades más específicas, dado que la tipología de restaurantes o locales existente es muy variada.

Otros como 11870[3], sólo permiten hacer búsquedas en ciudades enteras o por nombre del sitio en que se tiene interés, filtros que no ayudan a cercar la búsqueda a una zona específica.

Para ayudar en este tipo de decisiones, se desarrolla una aplicación web que permite a los usuarios obtener una lista de recomendaciones personalizada en función del tipo de plan que quieran realizar (ruta de cañas y tapas, vinos, menús del día, desayunos, meriendas, etc.) y de la zona donde quieran moverse. El producto aportará de esta forma valor al satisfacer esa necesidad de concreción que los usuarios demandan a la hora de decidir.

1.2. Objetivos del Trabajo

Los objetivos a alcanzar con este trabajo son los siguientes:

- Ofrecer una lista de recomendaciones ajustada a las preferencias marcadas por el usuario en su búsqueda.

- Mostrar los resultados de manera organizada y atractiva.

- Recabar información de los usuarios tras uso de alguno de los lugares recomendados.

- Realizar recomendaciones de lugares en función de los más valorados por el usuario y afinidades con otros usuarios.

1.3. Enfoque y método seguido

Al iniciar un proyecto web, las posibilidades tecnológicas son muchas, pudiendo variar, por ejemplo, entre distintas plataformas de programación como PHP, Java EE, ASP.NET, Python o Ruby.

Finalmente, se ha decidido hacer uso de ASP.NET. Este *framework* ofrece flexibilidad, estabilidad y robustez, siendo además *open-source*.
Asimismo, cabe destacar una mayor facilidad de programación gracias al IDE de Microsoft, Visual Studio, en conjunción con SQL Server, un sistema de gestión de base de datos rápido y seguro y que permite alojar grandes cantidades de información.

Otra de sus ventajas es la opción de elegir entre varios lenguajes de programación, siendo el escogido C#, beneficiándose así el trabajo de las características de un lenguaje orientado a objetos que ofrece incluso más funcionalidad que Java.

Junto a ASP.NET, se hará uso del patrón de diseño MVC (Modelo Vista Controlador), sobre el cual Microsoft está poniendo cada vez más énfasis en que es la mejor opción para crear una aplicación web.

Entre sus ventajas[4], destaca que ofrece una disminución de la complejidad, reduciendo el volumen de *code-behind*, respecto a, por ejemplo, el diseño basado en Web Forms[5].
A ello hay que unir una mejor gestión del cambio y del mantenimiento de la aplicación, gracias al uso del concepto SoC (*Separation of Concers*) [6], y una perfecta integración con otras tecnologías como jQuery. AngularJS o JSON.

En términos de técnicas de programación, se hacen uso de algoritmos de recomendación para ofrecer una lista de locales ajustada al perfil del usuario. Por otra parte, para obtener la información a mostrar a los usuarios, se plantea hacer uso de técnicas de *web scraping*.

La estrategia elegida es la de elaborar un producto nuevo que satisfaga las necesidades no cubiertas por productos existentes de concepto similar.

El resultado es una aplicación web usable tanto en ordenadores como dispositivos móviles (diseño *responsive*).

1.4. Planificación del Trabajo

Se plantean las tareas indicadas en la tabla siguiente:

Tabla 1. Planificación del Trabajo

Tarea	Fecha de Inicio	Duración	Fecha de Fin
PEC1 - Introducción	21/09/2016	13	04/10/2016
PEC2 - Documentación	05/10/2016	28	02/11/2016
Análisis Funcional	05/10/2016	16	21/10/2016
Diseño Técnico	21/10/2016	12	02/11/2016
Investigación	05/10/2016	14	19/10/2016
Construcción I	20/10/2016	13	02/11/2016
PEC3 - Fase Beta	03/11/2016	31	04/12/2016
Construcción II	03/11/2016	29	02/12/2016
Pruebas I	28/11/2016	4	02/12/2016
Entrega final	05/12/2016	35	09/01/2017
Construcción III	05/12/2016	31	05/01/2017
Pruebas II	05/12/2016	31	05/01/2017
Entregables	05/01/2017	4	09/01/2017

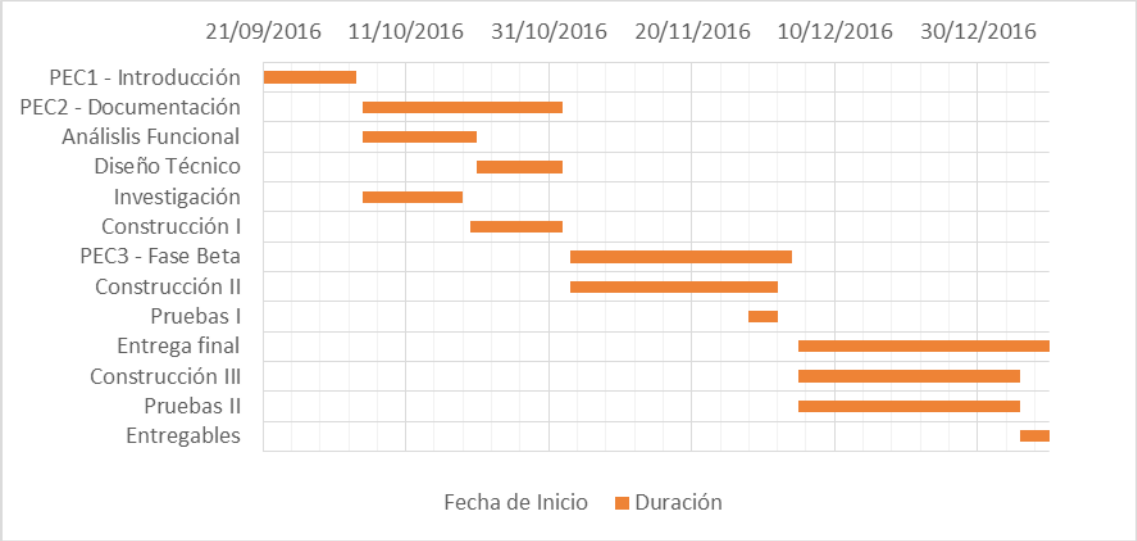
1. PEC1 – Introducción: Definición formal del proyecto y pauta de trabajo a seguir.

2. PEC2 – Documentación y comienzo de desarrollo
   2.1. Análisis Funcional: Análisis y revisión de requisitos y casos de uso de la aplicación.
   2.2. Diseño Técnico: Definición de los aspectos técnicos y de la arquitectura del trabajo
   2.3. Investigación: Búsqueda de tutoriales de apoyo y fuentes de información útiles de cara a dar comienzo a los desarrollos del trabajo.
   2.4. Construcción I: Comienzo de los trabajos de desarrollo.

3. PEC3 – Fase Beta
   3.1. Construcción II: Continuación de trabajos de desarrollo.
   3.2. Pruebas I: Primera batería de pruebas de diversa índole.

4. Entrega final.
   4.1. Construcción III: Continuación de trabajos de desarrollo.
   4.2. Pruebas II: Continuación de trabajos de pruebas sobre la aplicación.
   4.3. Entregables: Preparación de documentos escritos y audiovisuales para la exposición del trabajo.

Figura 1. Diagrama de Gantt



2. Análisis e Implementación

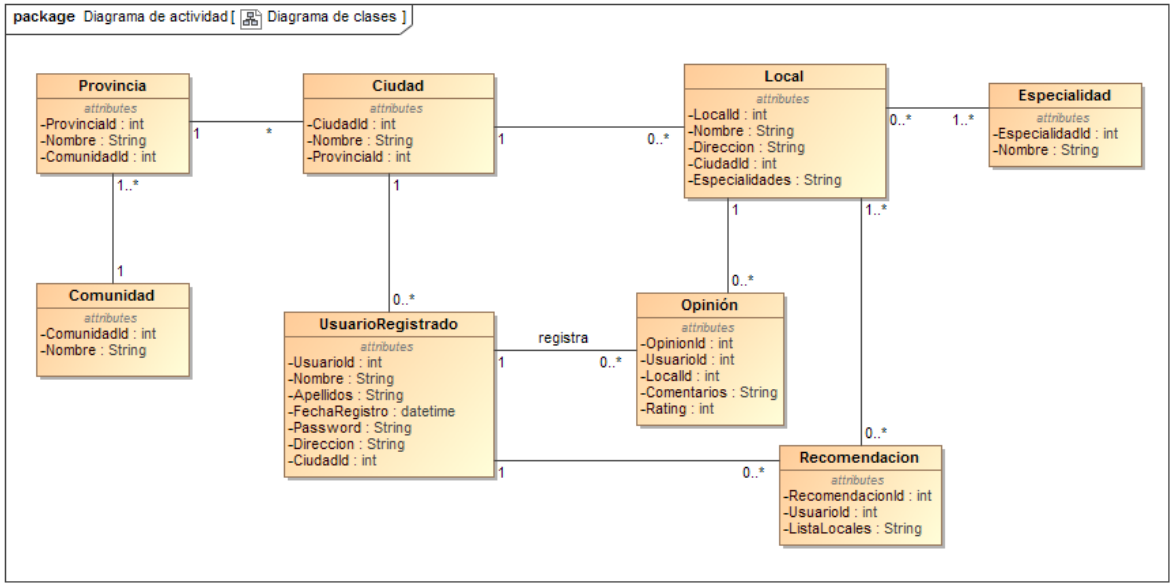
2.1. Diseño de la aplicación

1.1.1. Diagramas UML

A continuación, se presentan una serie de diagramas UML que ayudan a esclarecer la funcionalidad que pretende ofrecer el trabajo. Para ello, se ha utilizado el software de modelado MagicDraw.

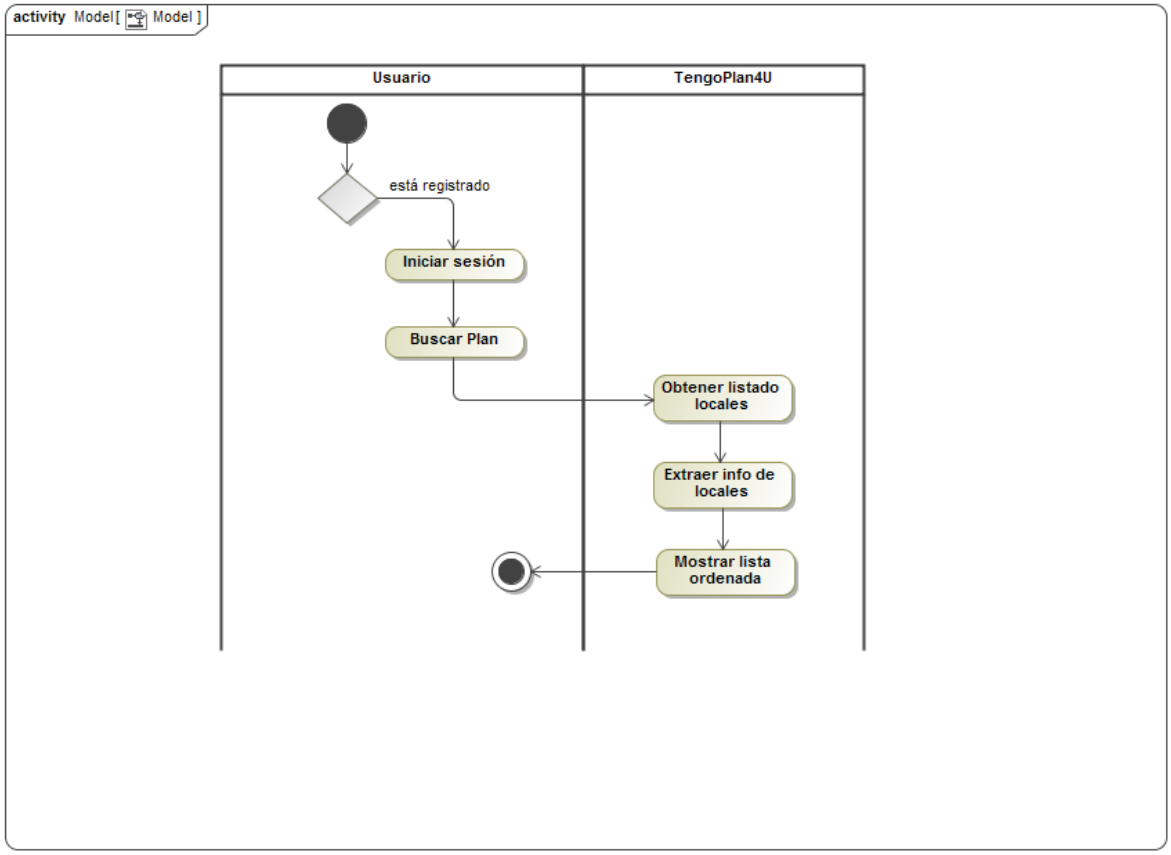
En primer lugar, se expone un diagrama de clases que modela la vista de diseño estática del sistema, mostrando el conjunto de entidades que lo conforman, así como sus atributos y relaciones estructurales:

Figura 2. Diagrama de clases



Para modelar el proceso principal de búsqueda que realiza el sistema, se ha hecho uso de un diagrama de actividades:

Figura 3. Diagrama de actividad



Por último, se define el comportamiento del sistema desde el punto de vista de observador externo mediante un diagrama de casos uso, los cuales serán desglosados seguidamente al diagrama:

Figura 4. Diagrama de casos de uso

Identificador	CU_001	
Nombre	Entrar Home	
Descripción	El usuario accede a la página principal de la aplicación.	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
Postcondición	-	
Excepciones	**Paso**	**Acción**

Identificador	CU_002	
Nombre	Registro de Usuario	
Descripción	El usuario crea su perfil en el sistema	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hace clic en “Registrarse”
	4	Se muestra el formulario de registro
	5	El usuario completa los datos del formulario y pulsa en Guardar.
	6	El usuario inicia sesión en la aplicación.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce algún dato incorrecto en el formulario.
	2	El sistema informa de los campos que debe corregir.

Identificador	CU_003	
Nombre	Búsqueda de plan	
Descripción	El usuario realiza una búsqueda en la aplicación	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**

	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario completa los filtros de búsqueda y pulsa en Buscar.
	4	Se muestra listado de locales obtenidos ordenados por relevancia.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario no introduce algún filtro obligatorio.
	2	El sistema informa de los campos que necesita introducir.

Identificador	CU_004	
Nombre	Consulta de opiniones	
Descripción	El usuario consulta las opiniones que tiene un local.	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario completa los filtros de búsqueda y pulsa en Buscar.
	4	Se muestra listado de locales obtenidos ordenados por relevancia.
	5	El usuario hace clic sobre el link Comentarios de un local del listado.
	6	Se muestran las opiniones registradas en el sistema para el local.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario no introduce algún filtro obligatorio en la búsqueda.
	2	El sistema informa de los campos que necesita introducir.

Identificador	CU_005	
Nombre	Consulta de reseñas de Google	
Descripción	El usuario consulta las reseñas que ofrece Google sobre el local.	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario completa los filtros de búsqueda y pulsa en Buscar.

	4	Se muestra listado de locales obtenidos ordenados por relevancia.
	5	El usuario hace clic sobre el link Reseñas Google de un local del listado.
	6	Se redirige al usuario a las reseñas de Google del local seleccionado.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario no introduce algún filtro obligatorio en la búsqueda.
	2	El sistema informa de los campos que necesita introducir.

Identificador	CU_006	
Nombre	Consulta de web del local	
Descripción	El usuario consulta la web oficial de un local.	
Actores principales	Usuario.	
Actores secundarios		
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario completa los filtros de búsqueda y pulsa en Buscar.
	4	Se muestra listado de locales obtenidos ordenados por relevancia.
	5	El usuario hace clic sobre web del local.
	6	Se redirige al usuario a la web del local seleccionado.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario no introduce algún filtro obligatorio en la búsqueda.
	2	El sistema informa de los campos que necesita introducir.

Identificador	CU_007	
Nombre	Inicio de sesión	
Descripción	El usuario inicia sesión en el sistema.	
Actores principales	Usuario registrado.	
Actores secundarios		
Precondición	El usuario tiene un perfil creado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hace clic en “Iniciar Sesión”.
	4	Se muestra el formulario de inicio de sesión.
	5	El usuario introduce nombre de usuario y contraseña y pulsa en Acceder.

	6	Se muestra la Home de la aplicación.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce un usuario y/o contraseña incorrectos.
	2	El sistema informa de los valores incorrectos.

Identificador	CU_008	
Nombre	Edición de perfil	
Descripción	El usuario edita su perfil	
Actores principales	Usuario registrado.	
Actores secundarios		
Precondición	El usuario tiene un perfil creado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hacer clic sobre su nombre de usuario en la parte superior de la interfaz.
	4	Se muestra la página de gestión de la cuenta de usuario.
	5	Hacer clic sobre el enlace Editar.
	6	Se muestra el formulario de edición de perfil de usuario.
	7	El usuario actualiza datos del formulario y hace clic en Guardar.
	8	Se redirige al usuario a la página de gestión de cuenta de usuario.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce un usuario y/o contraseña incorrectos.
	2	El sistema informa de los valores incorrectos.

Identificador	CU_009	
Nombre	Cambio de contraseña	
Descripción	El usuario modifica su contraseña.	
Actores principales	Usuario registrado.	
Actores secundarios		
Precondición	El usuario tiene un perfil creado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hacer clic sobre su nombre de usuario en la parte superior de la interfaz.
	4	Se muestra la página de gestión de la cuenta de usuario.
	5	Hacer clic sobre el enlace “Cambiar la contraseña”.
	6	Se muestra el formulario para cambio de contraseña.
	7	El usuario actualiza su contraseña y hace clic en el botón “Cambiar contraseña”.

	8	Se redirige al usuario a la página de gestión de cuenta de usuario.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce un usuario y/o contraseña incorrectos.
	2	El sistema informa de los valores incorrectos.

Identificador	CU_010	
Nombre	Registro de opinión	
Descripción	El usuario registra una opinión para un local.	
Actores principales	Usuario registrado.	
Actores secundarios		
Precondición	El usuario tiene un perfil creado en el sistema y ha realizado una búsqueda de locales la última vez que entró.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hace clic en “Iniciar Sesión”.
	4	Se muestra el formulario de inicio de sesión.
	5	El usuario introduce nombre de usuario y contraseña y pulsa en Acceder.
	7	El sistema muestra una página solicitando opiniones sobre los locales recomendados.
	8	El usuario valora y escribe su opinión sobre uno o varios locales y hace clic en Guardar.
	9	Se muestra la Home de la aplicación.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce un usuario y/o contraseña incorrectos.
	2	El sistema informa de los valores incorrectos.

Identificador	CU_011	
Nombre	Consulta de recomendación	
Descripción	El usuario consulta las recomendaciones del sistema.	
Actores principales	Usuario registrado.	
Actores secundarios		
Precondición	El usuario tiene un perfil creado en el sistema y ha realizado búsquedas en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce la URL de la aplicación y pulsa enter.
	2	Se muestra la página Home de la aplicación con la ciudad del usuario cargada.
	3	El usuario hace clic en “Iniciar Sesión”.
	4	Se muestra el formulario de inicio de sesión.
	5	El usuario introduce nombre de usuario y contraseña y pulsa en Acceder.

	6	Se muestra la Home de la aplicación.
	7	El sistema muestra el botón Recomendados para ti.
	8	El usuario hace clic sobre el botón.
	10	Se muestra listado de locales recomendados al usuario.
Postcondición	-	
Excepciones	**Paso**	**Acción**
	1	El usuario introduce un usuario y/o contraseña incorrectos.
	2	El sistema informa de los valores incorrectos.

1.1.2. Usabilidad/UX (DCU)

El diseño se centra en un usuario genérico, el cual puede ser cualquier visitante que acceda a la aplicación web con el objetivo de dilucidar el plan que quiere llevar a cabo en su ciudad. Adicionalmente, existe un perfil de usuario registrado y como tal, contará con funcionalidad extra al interactuar con la aplicación.

Gracias el *framework* Bootstrap[1], el cual ha sido agregado por Visual Studio automáticamente al proyecto, se consigue un diseño *responsive* que será accesible y adaptable a cualquier dispositivo móvil.

Por otra parte, se han recopilado las siguientes historias de usuario[8], entendiendo como tales apuntes de los requisitos del usuario utilizados para planificar, y que pueden ser actualizados, confirmados o eliminados conforme se avanzan en los trabajos del proyecto:

ID	HU_001
Como	Usuario.
Quiero	Saber lugares de tapas por una zona de mi ciudad.
Para	Poder decidir con mis amigos adónde ir.
Criterios de aceptación	• Encontrar zona elegida. • Obtener listado de locales del tipo deseado. • Ordenar listado por valoración de usuarios.

ID	HU_002
Como	Usuario registrado.
Quiero	Recibir recomendaciones de locales que se ajusten a mis gustos.
Para	Conseguir ideas de nuevos sitios que descubrir.
Criterios de aceptación	• Recopilar opiniones del usuario. • Obtener historial de búsquedas del usuario. • Comparar preferencias del usuario con la de otros usuarios para encontrar coincidencias.

ID	**HU_003**

[1] Bootstrap es el *framework* HTML, CSS y JavaScript más utilizado en el mundo para el desarrollo de aplicaciones web con un diseño *responsive*. Es de libre descarga y de código abierto

Como	Usuario.
Quiero	Conocer detalles de lo que ofrece un local.
Para	Para poder tomar una decisión con mayor facilidad.
Criterios de aceptación	• Obtener listado de locales del tipo deseado. • Obtener detalle de los locales y ofrecer acceso a dicha información.

Debido a que son más rápidos de crear, más económicos y de fácil arreglo, se ha decidido presentar una serie de prototipos Lo-Fi (*Low Fidelity*) para representar el diseño general de las interfaces:

Figura 5. Prototipo Home

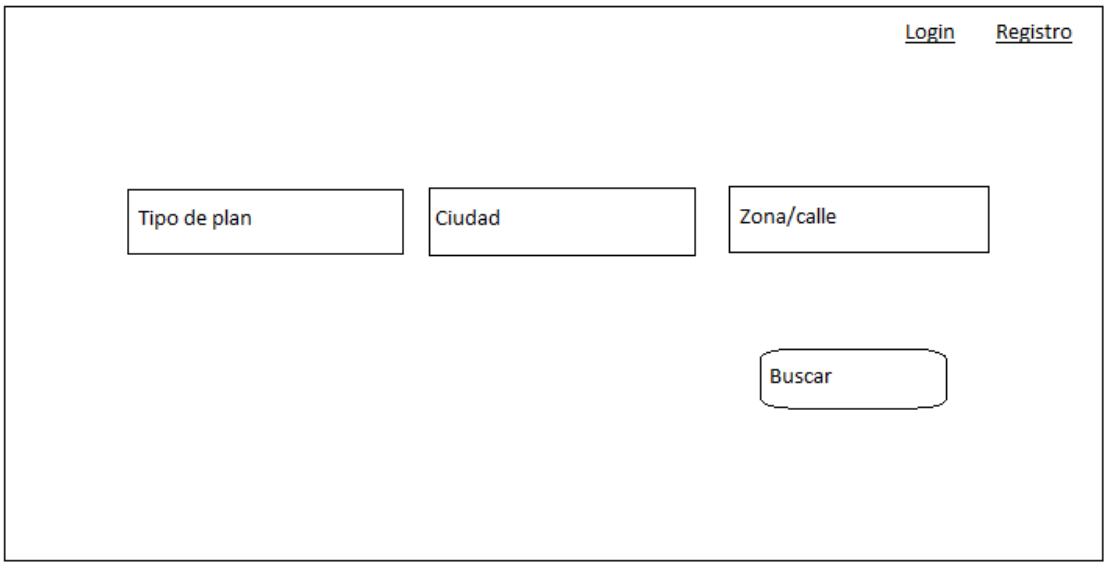


Figura 6. Prototipo Resultados Búsqueda

Figura 7. Prototipo Solicitud Opiniones

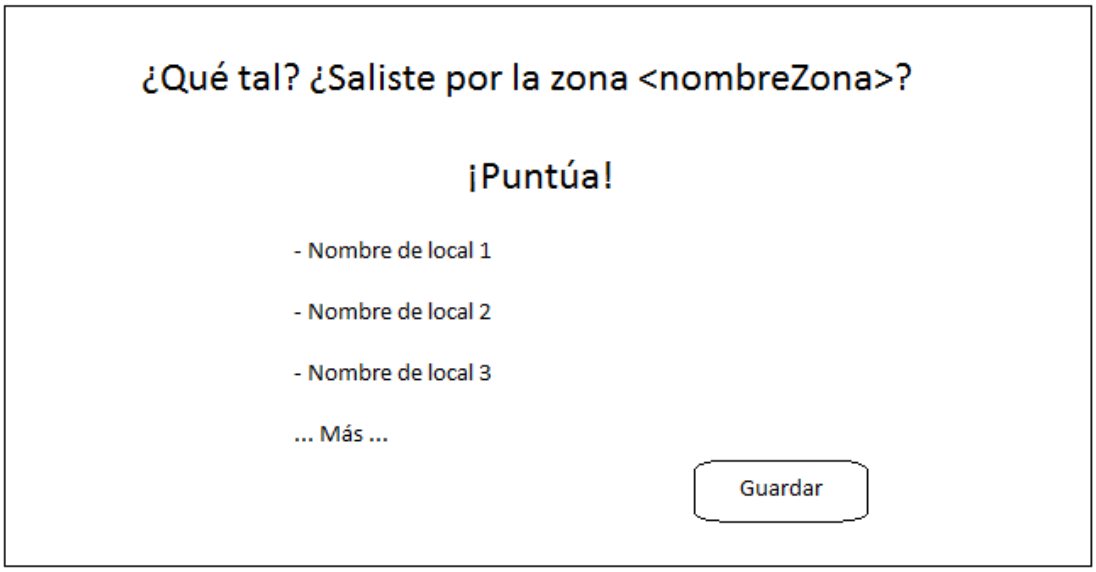
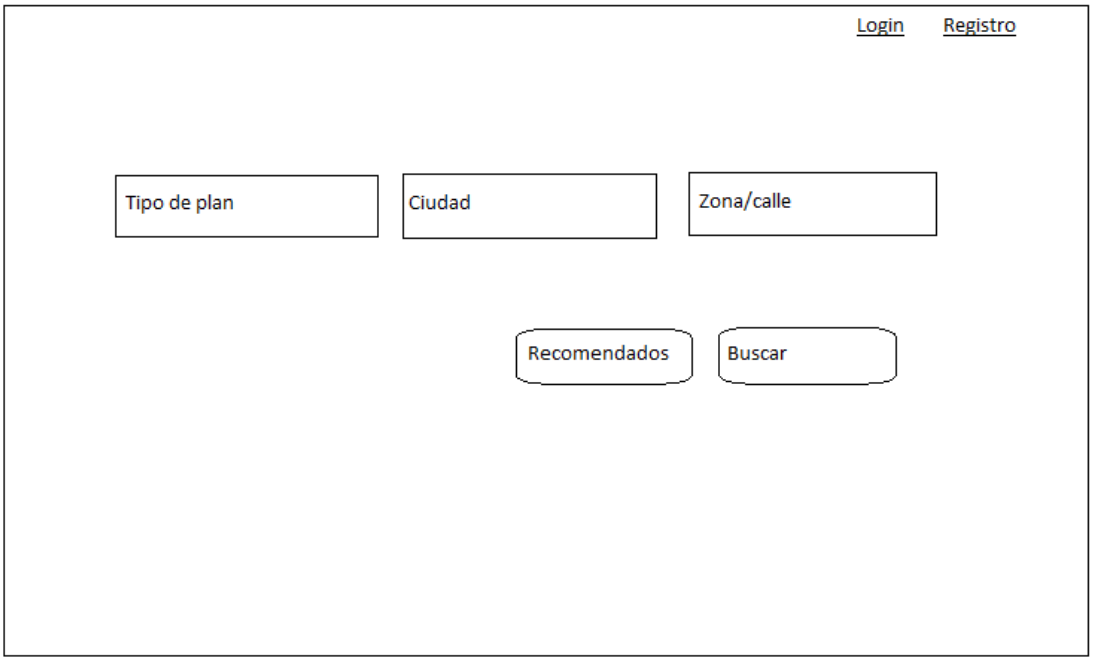


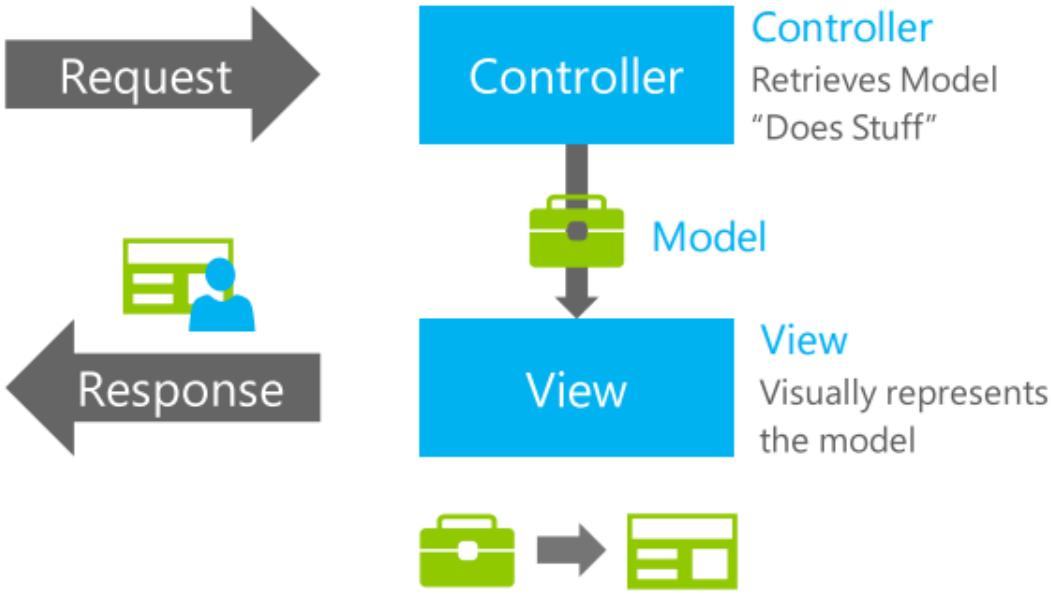
Figura 8. Prototipo Recomendaciones



1.1.3. Arquitectura

El proyecto está construido sobre el patrón de arquitectura software MVC (Modelo Vista Controlador), cuyo funcionamiento es explicado esquemáticamente mediante la siguiente imagen, extraída del curso introductorio a ASP.NET MVC de la Microsoft Virtual Academy[9]

Figura 9. Esquema Estructura MVC

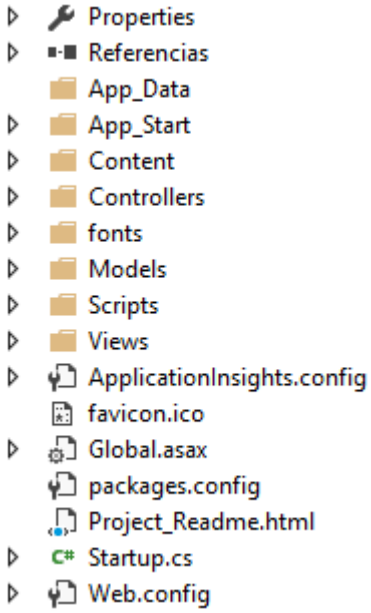


Como se mencionaba en el apartado 1.3 del presente documento, se ha optado por hacer uso del *framework* ASP.NET MVC, el cual aporta ventajas como facilitar la reutilización de código y su mantenimiento gracias a la separación de conceptos.

La figura nos muestra visualmente la división clara e independiente entre modelo de datos, lógica de negocio e interfaces de usuario. En el ciclo de vida de una aplicación web basada en MVC, la solicitud del usuario es recibida por el Controlador, el cual realiza la lógica de negocio necesaria y recupera el Modelo de datos, el cual será enviado a la Vista, quien se encargará de representarlo por pantalla y hacer llegar la respuesta al usuario.

La aplicación ha sido construida mediante una solución de Microsoft Visual Studio Community 2015 y sigue la estructura estándar ofrecida por este IDE para crear aplicaciones web basadas en ASP.NET MVC:

Figura 10. Estructura proyecto en Visual Studio



El modelo de datos utilizado extrae la información de una base de datos integrada en Microsoft SQL Server Express 2014 que se compone de las siguientes tablas:

Nombre	UsuariosRegistrados		
Descripción	Contiene datos de los usuarios registrados en el sistema		
Campos	**Nombre**	**Tipo**	**Nullable**
	UsuarioId	Entero	No
	Nombre	Texto	No
	Apellidos	Texto	No
	FechaRegistro	Fecha	No
	Password	Texto	No
	Direccion	Texto	Sí
	CiudadId	Entero	No

Nombre	Opiniones		
Descripción	Contiene opiniones de locales registradas por usuarios		
Campos	**Nombre**	**Tipo**	**Nullable**
	OpinionId	Entero	No
	UsuarioId	Entero	No
	GooglePlaceId	Texto	Sí
	Rating	Int	No
	Zona	Texto	Sí
	Comentarios	Texto	Sí

Nombre	Ciudades		
Descripción	Contiene información de ciudades		
Campos	**Nombre**	**Tipo**	**Nullable**

	CiudadId	Entero	No
	Nombre	Texto	No
	ProvinciaId	Entero	No

Nombre	Provincias		
Descripción	Contiene información de provincias		
Campos	**Nombre**	**Tipo**	**Nullable**
	ProvinciaId	Entero	No
	Nombre	Texto	No
	ComunidadId	Entero	No

Nombre	Comunidades		
Descripción	Contiene información de comunidades		
Campos	**Nombre**	**Tipo**	**Nullable**
	ComunidadId	Entero	No
	Nombre	Texto	No

Nombre	TiposPlanes		
Descripción	Contiene los tipos de plan disponibles		
Campos	**Nombre**	**Tipo**	**Nullable**
	TipoPlanId	Entero	No
	Nombre	Texto	Sí

La base de datos es construida haciendo uso de Entity Framework Code First[10]. El hecho de elegir el modo Code First quiere decir que las tablas se construyen cuando se lanza la aplicación en Visual Studio, basándose en los distintos modelos (clases) incluidos. Concretamente, se ha configurado de manera que la base de datos sea eliminada y creada de nuevo cada vez que se realice una modificación en el modelo.

Junto a lo referido anteriormente, la aplicación hace uso de APIs de Google para tener acceso a las clases y métodos que permiten obtener información detallada de locales registrados en Google Maps, esto incluye su dirección completa, número de teléfono, calificaciones y reseñas de usuarios, etc. En concreto, se consumen servicios de la siguientes APIS:

- Google Maps Geocoding API: permite obtener la localización y coordenadas de una zona o dirección solicitada por el usuario en los parámetros de búsqueda de locales.

- Google Places API Web Service: con este servicio se obtiene la información detallada de cada lugar a mostrar en el listado fruto de la búsqueda establecida por el usuario.

2.2. Desarrollo

1.1.4. Extractos de código

Para consumir los servicios de Google, es necesario componer las URLs a llamar para obtener información. La estructura de las utilizadas en la aplicación es mostrada seguidamente (en todas ellas el último parámetro se corresponde a la clave del proyecto Google, requisito indispensable para hacer uso de estos servicios):

Figura 11. URLs Servicios Google

```
ServiceGeocode = "https://maps.googleapis.com/maps/api/geocode/json?address={0}&key={1}";
ServicePlacesRadar = "https://maps.googleapis.com/maps/api/place/radarsearch/json?location={0}&radius=400&types={1}&key={2}";
ServicePlaceDetails = "https://maps.googleapis.com/maps/api/place/details/json?placeid={0}&key={1}";
ServicesPhotos = "https://maps.googleapis.com/maps/api/place/photo?maxwidth={0}&photoreference={1}&key={2}";
```

- ServiceGeocode: obtiene las coordenadas de una dirección pasada como parámetro.
- ServicesPlacesRadar: obtiene el listado de locales que cumple con los parámetros pasados en la URL. En este caso, la localización en coordenadas, el radio de acción en metros y los tipos de local. Los tipos deben estar incluidos en la lista de *types* soportados por Google[11] y son combinados con el separados "|". En la versión actual de aplicación, según la elección del usuario en su búsqueda, el parámetro *types* es compuesto de la siguiente manera:

Figura 12. Método GetTypesPlan

```
public static string GetTypesPlan(string tipoPlan)
{
    string types = string.Empty;
    switch (tipoPlan)
    {
        case "Caña + tapas":
            types = "bar";
            break;
        case "Desayunos":
            types = "bar|cafe";
            break;
        case "Menú del día":
            types = "restaurant|food";
            break;
    }
    return types;
}
```

- ServicePlacesDetails: obtiene el detalle de un local cuyo Id es pasado como parámetro.
- ServicePhotos: utilizada para obtener fotos del lugar, pasando como parámetro la máxima amplitud deseada para la foto y el código *photoreference* de la misma.

A continuación, se muestra extracto de la clase "PlansController.cs", el controlador encargado de extraer la información de los locales mostrados por pantalla. Concretamente el método "GetCoordinates() hace uso de la API de Google para obtener las coordenadas de la dirección o zona que ha introducido el usuario en el formulario de búsqueda:

Figura 13. Método GetCoordinates

```
private string GetCoordinates(string direccion)
{
    string googleGeo = string.Format(Estructuras.ServiceGeocode, direccion);
    var result = new System.Net.WebClient().DownloadString(googleGeo);
    GoogleGeocode.Rootobject googleData = JsonConvert.DeserializeObject<GoogleGeocode.Rootobject>(result);
    float latitud = googleData.results[0].geometry.location.lat;
    float longitud = googleData.results[0].geometry.location.lng;
    return latitud.ToString().Replace(",",".") + ", " + longitud.ToString().Replace(",", ".");
}
```

En el extracto anterior, se llama al servicio de Google con la dirección deseada, el cual devuelve la información en formato JSON y es deserializada en un objeto denominado “googleData”, con el que sí se puede interactuar para conseguir los datos buscados.

En la figura 12 mostrada seguidamente, se expone el método GetPlaces(), con el que se obtiene el listado de locales a mostrar al usuario, recibiendo como parámetro las coordenadas que devuelve el método referido anteriormente:

Figura 14. Método GetPlaces

```
public static List<Local> GetPlaces(string coordinates, string tipoPlan)
{
    string API_KEY = string.Empty;
    if (DateTime.Now.Second % 2 == 0)
    {
        API_KEY = Estructuras.API_KEY;
    }
    else API_KEY = Estructuras.API_KEY_2;
    string types = GetTypesPlan(tipoPlan);
    List<Local> listaLocales = new List<Local>();
    string googlePlaces = string.Format(Estructuras.ServicePlacesRadar, coordinates, types, API_KEY);
    WebClient wc = new WebClient();
    wc.Encoding = Encoding.UTF8;
    var result = wc.DownloadString(googlePlaces);
    GooglePlacesRadar.Rootobject googleData = JsonConvert.DeserializeObject<GooglePlacesRadar.Rootobject>(result);
    foreach (var place in googleData.results)
    {
        GooglePlaceDetails.Rootobject placeDetails = GetPlaceDetails(place.place_id);
        if (placeDetails.status.Equals("OK"))
        {
            Local local = new Local();
            local.Nombre = placeDetails.result.name;
            local.GooglePlaceId = placeDetails.result.place_id;
            local.Direccion = placeDetails.result.formatted_address;
            local.Rating = placeDetails.result.rating;
            local.UrlGoogleMaps = placeDetails.result.url;
            if (!string.IsNullOrEmpty(placeDetails.result.website))
            {
                local.Website = placeDetails.result.website;
            }
            else
            {
                local.Website = "sin-web";
        }
        if (placeDetails.result.photos != null)
        {
            local.UrlImagen = string.Format(Estructuras.ServicesPhotos, 300, placeDetails.result.photos[0].photo_reference, API_KEY);
        }
        else
        {
            local.UrlImagen = "../Images/tengoPlan.png";
        }
        if (local.Rating > 0) listaLocales.Add(local);
    }
}
return listaLocales;
```

Como en el método de la figura 11, en este método se llama al servicio de Google que devuelve los listados de locales en una zona determinada, para a continuación llamar al método GetPlacesDetails para cada uno de ellos. La información en formato JSON es igualmente tratada para finalmente devolver un *array* con los distintos locales encontrados.

El métoto GetPlacesDetails mencionado en el párrafo anterior es mostrado en la siguiente figura:

Figura 15. Método GetPlaceDetails

```
public static GooglePlaceDetails.Rootobject GetPlaceDetails(string placeId)
{
    string API_KEY = string.Empty;
    if (DateTime.Now.Second % 2 == 0)
    {
        API_KEY = Estructuras.API_KEY;
    }
    else API_KEY = Estructuras.API_KEY_2;
    string googlePlaceDetails = string.Format(Estructuras.ServicePlaceDetails, placeId, API_KEY);
    WebClient wc = new WebClient();
    wc.Encoding = Encoding.UTF8;
    var result = wc.DownloadString(googlePlaceDetails);
    GooglePlaceDetails.Rootobject googleData = JsonConvert.DeserializeObject<GooglePlaceDetails.Rootobject>(result);
    return googleData;
}
```

Por último, mencionar también como código destacado el método utilizado[12] para obtener recomendaciones de lugares a los usuarios. Se hace uso de la correlación de Pearson[13], medida de similitud que puede tomar un valor en el rango [–1,1]. Si su valor es 1 indica que las dos variables están perfectamente relacionadas; si es 0, no hay relación lineal entre ellas. La idea es utilizar este método para comparar las preferencias del usuario logado con el resto opiniones registradas en la base de datos, de manera que a continuación se pueda conseguir aquel usuario o usuarios con mayor afinidad:

```
public static double CoeficientePearson(Dictionary<string, int> valores1, Dictionary<string, int> valores2)
{
    int[] vector1 = valores1.Values.ToArray();
    int[] vector2 = valores2.Values.ToArray();

    double[] vector_xy = new double[vector1.Length];
    double[] vector_xp2 = new double[vector1.Length];
    double[] vector_yp2 = new double[vector1.Length];
    for (int i = 0; i < vector1.Length; i++)
        vector_xy[i] = vector1[i] * vector2[i];
    for (int i = 0; i < vector1.Length; i++)
        vector_xp2[i] = Math.Pow(vector1[i], 2.0);
    for (int i = 0; i < vector1.Length; i++)
        vector_yp2[i] = Math.Pow(vector2[i], 2.0);
    double sum_x = 0;
    double sum_y = 0;
    foreach (double n in vector1)
        sum_x += n;
    foreach (double n in vector2)
        sum_y += n;
    double sum_xy = 0;
    foreach (double n in vector_xy)
        sum_xy += n;
    double sum_xpow2 = 0;
    foreach (double n in vector_xp2)
        sum_xpow2 += n;
    double sum_ypow2 = 0;
    foreach (double n in vector_yp2)
        sum_ypow2 += n;
    double Ex2 = Math.Pow(sum_x, 2.00);
    double Ey2 = Math.Pow(sum_y, 2.00);

    double den = Math.Sqrt((vector1.Length * sum_xpow2 - Ex2) * (vector1.Length * sum_ypow2 - Ey2));
     double num = (vector1.Length * sum_xy - sum_x * sum_y);
     if(den == 0)
     {
         return 0;
     }
     double coeficiente = num /den;

     return coeficiente;
 }
```

1.1.5. Seguridad

La aplicación enfrenta los retos de seguridad de acceso beneficiándose de la arquitectura de autenticación, seguridad y autorización que ofrece ASP.NET MVC. En esta nueva fórmula ofrecida por ASP.NET se incluye el sistema Identity[14], el cual permite:

- Acceder fácilmente a la información del perfil del usuario.
- Aporta control de persistencia, permitiendo la modificación de la base de datos fácilmente. Ésta es creada con Entity Framework Code First.
- Provee un sistema de roles que permite restringir el acceso a determinadas partes de la aplicación.
- Permite añadir fácilmente métodos de acceso a través de redes sociales como Facebook, Twitter o Google.

Los datos sensibles, como las contraseñas de los usuarios, son almacenados encriptados en la base de datos. A ello hay que añadir que esta información introducida por el usuario debe tener al menos un carácter que no sea una letra ni un dígito, al menos un dígito ('0'-'9') y al menos una letra en mayúscula ('A'-'Z'), buscando de esta manera la complejidad

de la misma y la seguridad de los datos de usuario. Estos avisos son reportados al usuario a través de controles JQuery.

1.1.6. Test

Para evaluar las funcionalidades de la aplicación, se plantea el siguiente plan de pruebas de aceptación por parte del usuario:

Identificador	CP_001		
Objetivo	Comprobar entrada Home		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación

Identificador	CP_002		
Objetivo	Comprobar registro de usuario		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 2	**Entrada**	Hacer clic en el enlace “Registrarse”
		Salida esperada	La aplicación muestra el formulario de registro
	Paso 3	**Entrada**	Completar los campos del formulario y pulsar en Guardar
		Salida esperada	El sistema informa de que el usuario se ha creado correctamente

Identificador	CP_003		
Objetivo	Comprobar búsqueda de plan		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 2	**Entrada**	Completar los filtros de búsqueda y pulsar en Buscar
		Salida esperada	La aplicación muestra listado de locales encontrados ordenados por relevancia

Identificador	CP_004		
Objetivo	Comprobar consulta de opiniones		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador

		Salida esperada	Se muestra la página Home de la aplicación con la ciudad del usuario cargada
	Paso 2	**Entrada**	Completar los filtros de búsqueda y pulsar en Buscar
		Salida esperada	La aplicación muestra listado de locales encontrados ordenados por relevancia
	Paso 3	**Entrada**	Hacer clic sobre el link de comentarios de uno de los locales
		Salida esperada	La aplicación muestra las opiniones registradas en el sistema para el local

Identificador	CP_005		
Objetivo	Comprobar consulta de reseñas de Google		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 2	**Entrada**	Completar los filtros de búsqueda y pulsar en Buscar
		Salida esperada	La aplicación muestra listado de locales encontrados ordenados por relevancia
	Paso 3	**Entrada**	Hacer clic sobre el link Reseñas Google de un local del listado
		Salida esperada	El usuario es redirigido a las reseñas de Google del local seleccionado

Identificador	CP_006		
Objetivo	Comprobar consulta de web del local		
Prerrequisitos			
Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 2	**Entrada**	Completar los filtros de búsqueda y pulsar en Buscar
		Salida esperada	La aplicación muestra listado de locales encontrados ordenados por relevancia
	Paso 3	**Entrada**	Hacer clic sobre la web del local
		Salida esperada	El usuario es redirigido a la web del local seleccionado

Identificador	CP_007
Objetivo	Comprobar inicio de sesión
Prerrequisitos	El usuario está registrado en la aplicación

Procedimientos	**Paso 1**	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 2	**Entrada**	Hacer clic en “Iniciar sesión”
		Salida esperada	La aplicación muestra el formulario de inicio de sesión
	Paso 3	**Entrada**	Introducir credenciales y hacer clic en el botón Iniciar sesión
		Salida esperada	El usuario es redirigido a la Home de la aplicación

Identificador	CP_008		
Objetivo	Comprobar edición de perfil		
Prerrequisitos	El usuario ha iniciado sesión en la aplicación		
	Paso 2	**Entrada**	Hacer clic en sobre nombre de usuario en la parte superior de la interfaz
		Salida esperada	Se muestra la página de gestión de la cuenta de usuario
	Paso 3	**Entrada**	Hacer clic sobre el link Editar
		Salida esperada	Se muestra el formulario de edición de perfil de usuario
	Paso 4	**Entrada**	Actualizar datos del formulario y hacer clic en Guardar
		Salida esperada	Se redirige al usuario a la página de gestión de la cuenta de usuario

Identificador	CP_009		
Objetivo	Comprobar cambio de contraseña		
Prerrequisitos	El usuario ha iniciado sesión en la aplicación		
	Paso 2	**Entrada**	Hacer clic en sobre nombre de usuario en la parte superior de la interfaz
		Salida esperada	Se muestra la página de gestión de la cuenta de usuario
	Paso 3	**Entrada**	Hacer clic sobre el link “Cambiar la contraseña”
		Salida esperada	Se muestra el formulario de cambio de contraseña
	Paso 4	**Entrada**	Actualizar contraseña y hacer clic en el botón “Cambiar contraseña”
		Salida esperada	Se redirige al usuario a la página de gestión de la cuenta de usuario

Identificador	CP_010
Objetivo	Comprobar registro de opinión
Prerrequisitos	El usuario está registrado en la aplicación

	Paso 2	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 3	**Entrada**	Hacer clic en “Iniciar sesión”
		Salida esperada	La aplicación muestra el formulario de inicio de sesión
	Paso 4	**Entrada**	Introducir credenciales y hacer clic en el botón Iniciar sesión
		Salida esperada	El usuario es redirigido a una página solicitando opiniones sobre los locales recomendados
	Paso 5	**Entrada**	Escribir opinión y valorar uno o varios locales y hacer clic en Guardar
		Salida esperada	Se muestra la Home de la aplicación

Identificador	CP_011		
Objetivo	Comprobar consulta de recomendación		
Prerrequisitos	El usuario está registrado en la aplicación		
	Paso 2	**Entrada**	Introducir la URL de la aplicación en el navegador
		Salida esperada	Se muestra la página Home de la aplicación
	Paso 3	**Entrada**	Hacer clic en “Iniciar sesión”
		Salida esperada	La aplicación muestra el formulario de inicio de sesión
	Paso 4	**Entrada**	Introducir credenciales y hacer clic en el botón Iniciar sesión
		Salida esperada	El usuario es redirigido a la Home de la aplicación y se muestra un botón ofreciendo al usuario locales recomendados
	Paso 5	**Entrada**	Hacer clic sobre el botón
		Salida esperada	Se muestra listado de locales recomendados al usuario

1.1.7. Bugs

El bug más destacado ha consistido en que, en determinados momentos durante el testeo de la aplicación, la lista de locales de una búsqueda obtenida de la API de Google se veía reducida drásticamente, además de no ofrecer la información correctamente, mostrando imágenes de error por defecto de Google en lugar de las fotos del sitio. Esto se ha debido a la limitación de uso diaria de la API de Google, ya que el servicio gratuito sólo ofrece un tope de solicitudes al día. Por tanto, en futuras mejoras y ante un número de usuarios mayor habría que contratar un servicio Premium de estos servicios de Google.

3. Conclusiones

En primer lugar, me planteé el trabajo como un reto, sobre todo, tecnológico y como una oportunidad para conocer técnicas de las que no hago uso en mi puesto de trabajo actual y en las que me gustaría adentrarme en un futuro próximo. Una vez realizada una planificación inicial del proyecto, los primeros pasos fueron un proceso de investigación y aprendizaje de ASP.NET MVC, esfuerzo que podría haber sido evitado usando la tecnología a la que estoy habituado, pero prefería aceptar el reto. De estos meses impregnado de este patrón diseño, si bien cuesta habituarse al principio, me quedo con la facilidad proveída para estructurar un proyecto adecuadamente, además de la reducción de código respecto al ASP clásico. Por tanto, en este punto, obtengo una conclusión positiva y me motiva para continuar aprendiendo más de esta tecnología.

Respecto a los objetivos iniciales planteados en la memoria, si los tomamos como requisitos de funcionalidad de la aplicación, considero que han sido satisfechos con el trabajo realizado. Sí mencionar que algunas tecnologías con intención de ser incluidas no han sido finalmente parte del proyecto, aunque se deja la puerta abierta para las líneas de trabajo futuras, aspecto que será abordado más en profundidad en líneas posteriores. Una de los puntos no incluidos finalmente ha sido el uso de técnicas de *web scraping*, ya que en esta primera versión funcional a entregar, se ha considerado que el servicio web de Google permitía la suficiente extracción de información para mostrar al usuario, unido a que existe un elevado número de locales que no cuentan con web propia, algo no valorado en la planificación y que, por tanto, ha hecho que finalmente se excluya el uso de estas técnicas.
Sí que se incluye un algoritmo de recomendación, conocimiento adquirido gracias a la asignatura Inteligencia Artificial Avanzada de este mismo máster.

La planificación del trabajo se plantea en un principio siguiendo un ciclo de vida básico o modelo lineal secuencial en cascada. Sin embargo, a lo largo del proceso se ha aprendido que hubiera sido más apropiado utilizar una metodología ágil. El hecho de compaginar el desarrollo del trabajo con la actividad laboral y el resto de obligaciones que surgen día a día ha complicado ajustarse a los tiempos planteados inicialmente. Por ello, aunque no ha sido lo planteado formalmente, se puede afirmar que en la práctica se han seguido algunas técnicas ágiles, al tener el proyecto ciertas características que ayudaban a ello. Por ejemplo, el proyecto no es de un volumen grande y ha sido flexible en cuanto a admitir cambios en su funcionalidad, además de contar pocos artefactos a entregar. Asimismo, se ha hecho hincapié en la simplicidad, tanto visual como de uso. Estos cambios han permitido garantizar el éxito del trabajo.

Como se ha mencionado anteriormente, no se ha decidido incluir técnicas de *web scraping* en esta versión del trabajo. No obstante, se plantea en próximas actualizaciones el uso de alguna técnica de este tipo para conseguir información adicional de los locales y hacer que los usuarios tengan disponible una búsqueda avanzada que incluya más filtros además de los ya presentes.

Otra característica valorable de ser añadida es permitir el inicio de sesión en la aplicación a través de servicios externos como pueden ser Facebook o Google, tomando como ventaja la facilidad que aporta la estructura de ASP.NET MVC para incluir esta funcionalidad.

En cuanto a las recomendaciones que proporciona la aplicación al usuario, se plantea mejorar el algoritmo de recomendación, actualmente usando la correlación de Pearson. Este método sólo mide relaciones lineales, por lo que conforme suba el volumen de usuarios y opiniones en la base de datos, se pretende utilizar algún algoritmo de recomendación ponderada, técnica que calcula la media de las valoraciones ponderada por la afinidad del usuario correspondiente. De esa manera, la valoración global de los locales está personalizada para cada usuario y, por tanto, debería resultar más útil y acertada.

Por último, la línea de trabajo en la que se pretende hacer más énfasis en el futuro es la mejora del servicio de búsqueda. Se desea que al entrar en TengoPlan4U, sea detectada la localización del usuario y automáticamente se cargue en los filtros de búsqueda la ciudad en la que se encuentra. Adicionalmente, paulatinamente se incluirán una mayor variedad de tipos de planes, de la mano también de la profundidad que se consiga con las técnicas de *web scraping* al buscar la información de los locales.

4. Glosario

1. Términos
   1.1. Framework: estructura de conceptos y módulos tecnológicos concretos que sirve de base para el desarrollo software.
   1.2. Open-source: código abierto, software distribuido y desarrollado de forma libre, es decir, es posible modificar la fuente del programa sin restricciones de licencia.
   1.3. Web scraping: técnica utilizada para extraer información de páginas web, tarea realizada por programas que simulan la navegación de un humano por el sitio web
   1.4. Diseño responsive: filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarlas. Se centra especialmente en la posibilidad de visualización en dispositivos móviles.
   1.5. Array: conjunto de elementos del mismo tipo. Por ejemplo, un vector de números enteros.

2. Acrónimos
   2.1. IDE (Integrated Development Environment): aplicación que provee facilidades a programadores para desarrollar sus trabajos software. Normalmente cuenta con un editor de texto, motor de compilación automática y un debugador.
   2.2. MVC (Model View Controller): patrón de diseño de programas software que separa la estructura de la aplicación en tres partes interconectadas: modelo de datos, vista y controlador.
   2.3. SoC (Separation of Concerns): principio de diseño consistente en separar en distintas secciones un programa.
   2.4. API (Application Programming Interface): conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca para ser utilizada por otro software como una capa de abstracción.

5. Bibliografía

[1] Estadísticas uso de motores de búsqueda en el mundo. gs.statcounter.com. Consultado el 03/10/2016.

[2] https://www.tripadvisor.es/. Consultado el 03/10/2016.

[3] https://11870.com/. Consultado el 04/10/2016.

[4] Beneficios de ASP.NET MVC. cmarix.com. Consultado el 04/10/2016

[5] Ventajas de MVC frente a Web Forms. codeproject.com. Consultado el 04/10/2016

[6] Sepataraion of Concerns. wikipedia.org. Consultado el 04/10/2016.

[7] Web scraping. wikipedia.org. Consultado el 04/10/2016.

[8] Cómo escribir una historia de usuario. agilecoaching.com.ar. Consultado el 24/10/2016.

[9] Introduction to ASP.NET MVC. mva.microsoft.com. Consultado el 01/11/2016

[10] https://msdn.microsoft.com/es-es/data/jj679962(v=vs.113).aspx. Consultado el 10/11/2016

[11] https://developers.google.com/places/supported_types. Consultado el 27/11/2016

[12] http://mantascode.com/c-how-to-get-correlation-coefficient-of-two-arrays/. Consultado el 05/01/2017

[13] https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson. Consultado el 05/01/2017

[14] https://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity. Consultado el 06/01/2017