**Plan Your Trip**

Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2º

Barea López, Manuel (manbarlop1@alum.us.es)

Outeiriño Barneto, Manuel (manoutbar@alum.us.es)

Tutor: Antonio Gámez Díaz

Número de grupo: 4

Enlace de la aplicación: https://planyourtrip.appspot.com/

Enlace de proyecto en GitHub: https://github.com/ManuBarea/PlanYourTrip.git

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 17/03/2019 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | Manuel Barea  Manuel Outeiriño |
| 28/04/2019 | 1.1 | -Se añade diagrama de componentes.  -Despliegue de la aplicación. | Manuel Barea  Manuel Outeiriño |
| 26/05/2019 | 2.0 | -Se incluye todos los diagramas.  -Documentación API REST.  -Pruebas.  -Aplicación web desplegada. | Manuel Barea  Manuel Outeiriño |

Índice

[1 Introducción 4](#_Toc471899224)

[1.1 Aplicaciones integradas 4](#_Toc471899225)

[1.2 Evolución del proyecto 4](#_Toc471899226)

[2 Prototipos de interfaz de usuario 5](#_Toc471899227)

[2.1 Vista X 5](#_Toc471899228)

[2.2 Vista Y 5](#_Toc471899229)

[3 Arquitectura 6](#_Toc471899230)

[3.1 Diagrama de componentes 6](#_Toc471899231)

[3.2 Diagrama de despliegue 6](#_Toc471899232)

[3.3 Diagrama de secuencia de alto nivel 7](#_Toc471899233)

[3.4 Diagrama de clases 7](#_Toc471899234)

[3.5 Diagramas de secuencia 8](#_Toc471899235)

[4 Implementación 10](#_Toc471899236)

[5 Pruebas 11](#_Toc471899237)

[6 Manual de usuario 14](#_Toc471899238)

[6.1 Mashup 14](#_Toc471899239)

[6.2 API REST 14](#_Toc471899240)

[Referencias 15](#_Toc471899241)

# Introducción

La aplicación que se va a desarrollar es un mashup llamado “PLAN YOUR TRIP”, que permitirá planificar de forma global un viaje. Inicialmente se buscará el lugar de destino con el soporte de *MAPBOX y* se proporcionará información de lugares de interés en la zona con *FOURSQUARE.* A su vez el servicio de *FLICKR* apoyará la visualización del contenido que sea relevante para el usuario.

## Aplicaciones integradas

Describir cada una de las aplicaciones integradas dando detalles sobre cada una de ellas

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| MAPBOX | https://docs.mapbox.com/api/ |
| FOURSQUARE | https://developer.foursquare.com/docs |
| FLICKR | https://www.flickr.com/services/api/ |

Tabla 1. Aplicación integradas

## Evolución del proyecto

El proyecto tuvo que reestructurarse para adaptarse al modelo MVC. Debido a que basamos el proyecto en REACT, que normalmente se utiliza para la parte de Front End del desarrollo.

# Prototipos de interfaz de usuario

Insertar las imágenes de todos los prototipos desarrollados. Añadir para cada prototipo una breve descripción textual. Se recomienda hacer prototipos simples y realistas. Para los prototipos pueden usarse aplicaciones como Balsamiq [1]

## Vista Global

A continuación, se muestra una visión global del mockup. Se parte de una página inicial que contiene un mapa (Mapbox API) donde se mostraran los lugares de interés (Foursquare API) después de haber realizado una búsqueda.

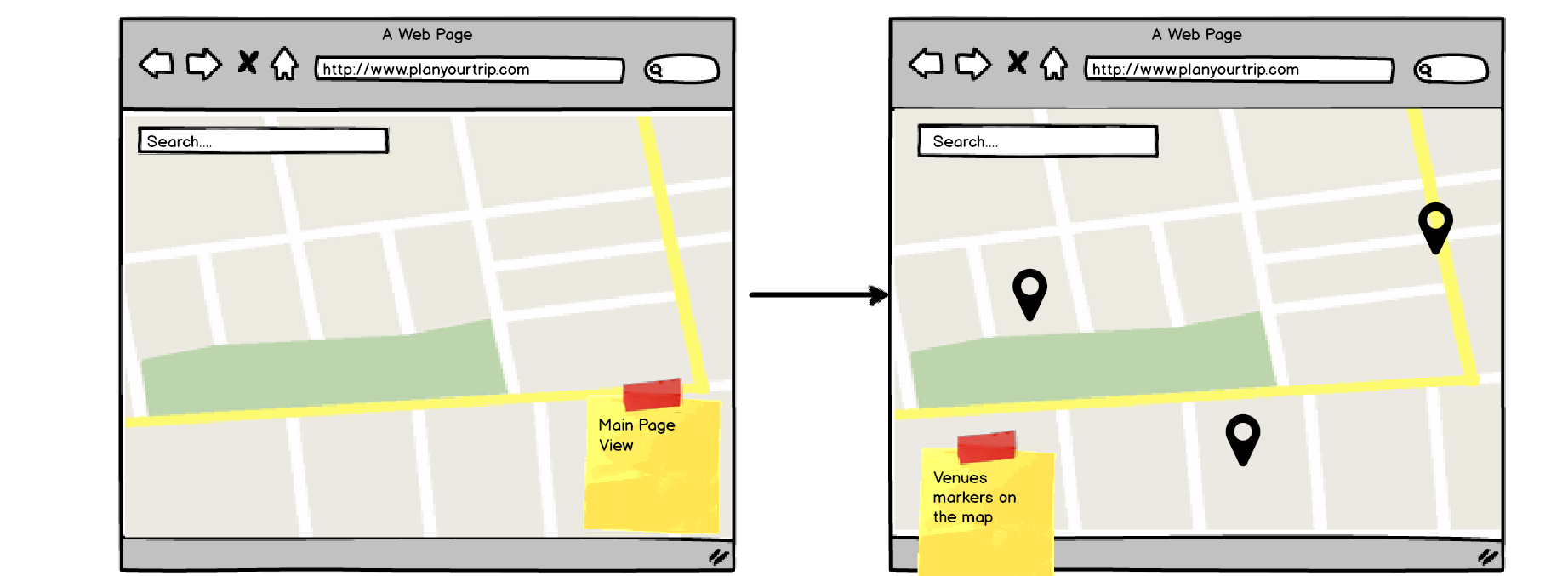
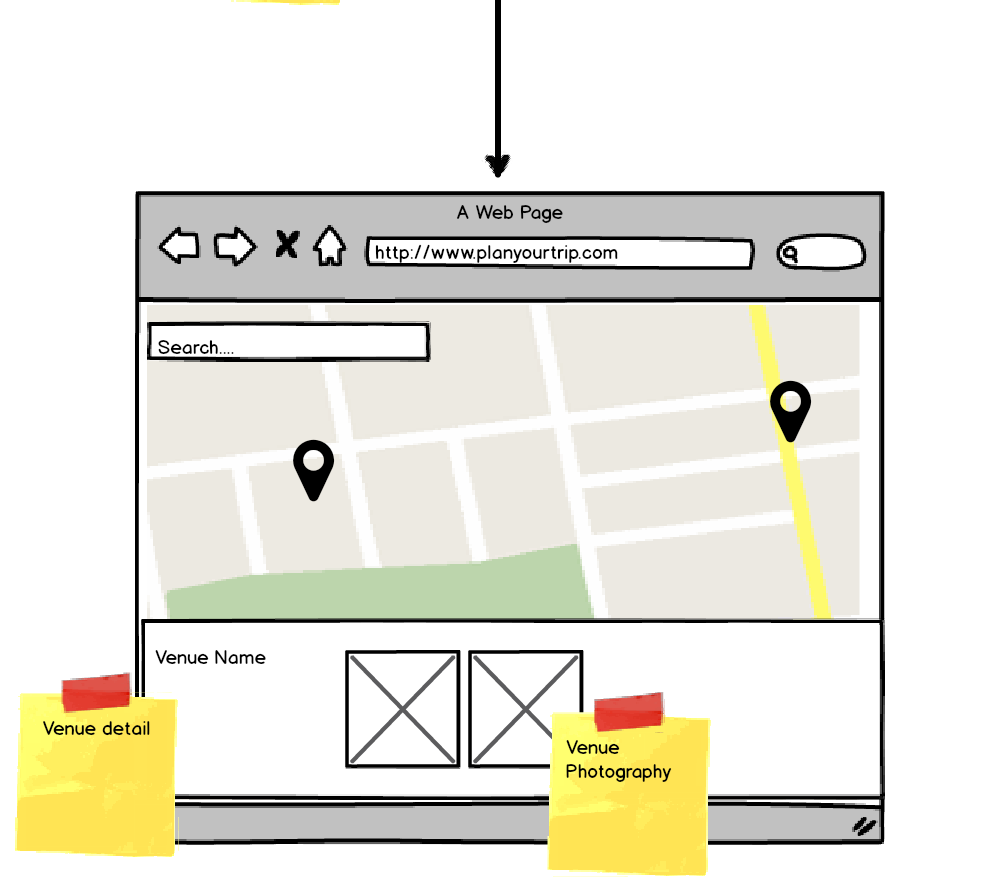


Figura 1. Prototipo de interfaz de usuario de la vista Global y de resultado de busqueda



## Vista detalle

De la vista de resultados de búsqueda se obtiene la vista detallada al hacer click sobre el el marcador en el mapa.

La información de detalle contiene fotos (Flicker API).

Figura 2. Prototipo de interfaz de usuario de la vista Global y de resultado de busqueda

# Arquitectura

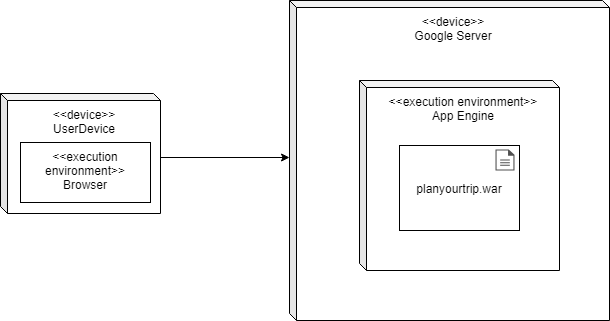
## Diagrama de componentes

Diagrama UML de componentes de alto nivel. Debe incluir las aplicaciones integradas y nuestra propia aplicación como un componente independiente.

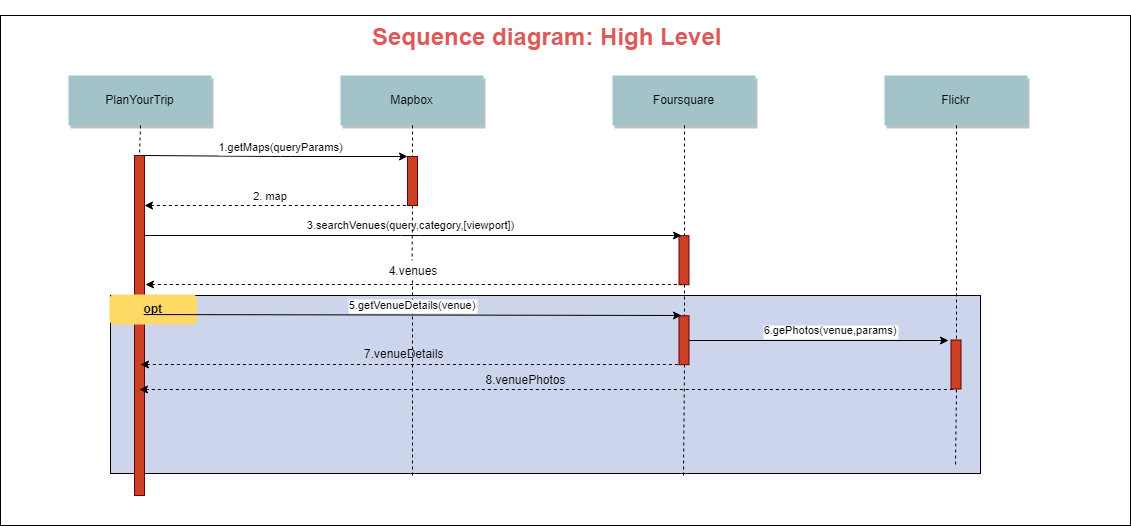


## Diagrama de despliegue

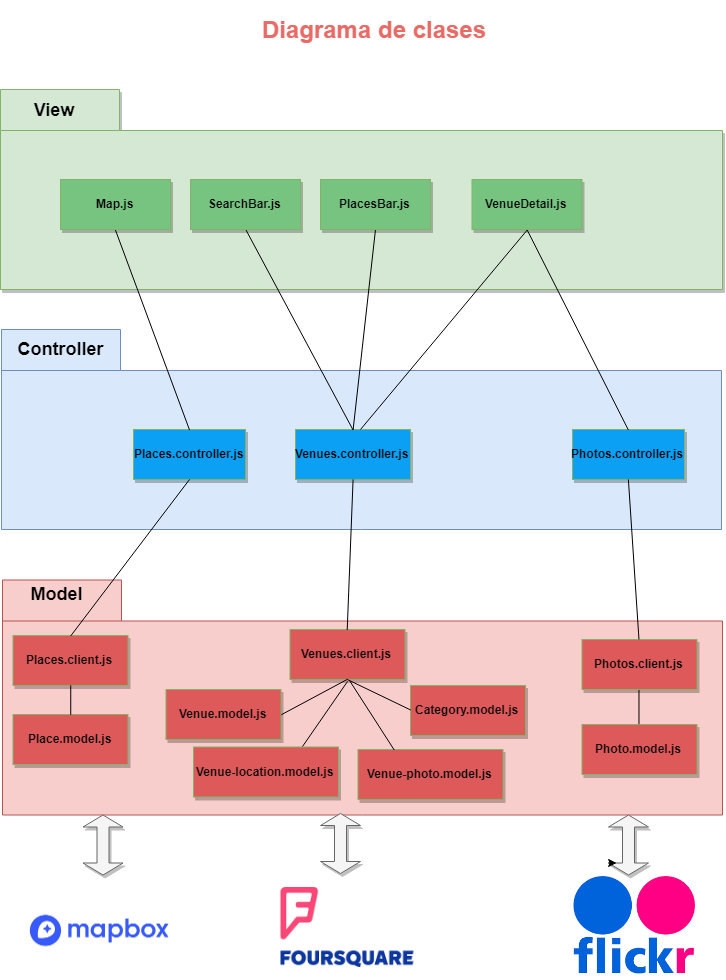
Diagrama UML de despliegue de la aplicación.



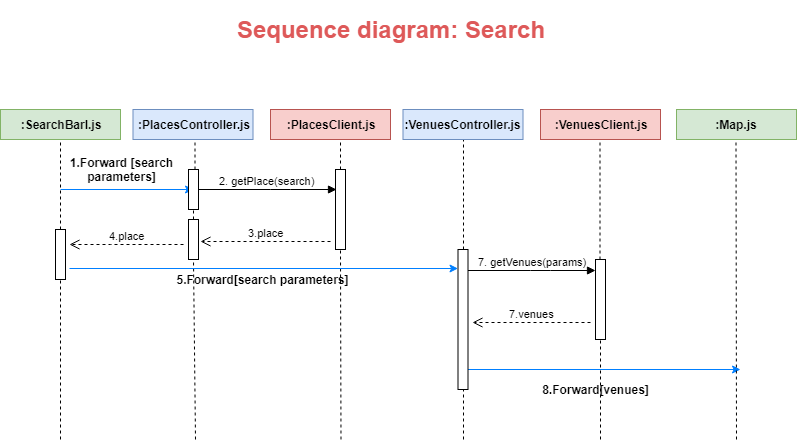
## Diagrama de secuencia de alto nivel

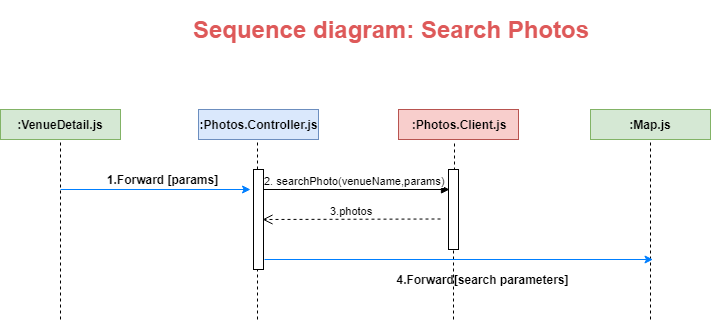
Diagrama UML de secuencia indicando el flujo de mensajes entre las distintas aplicaciones integradas.

## Diagrama de clases



## Diagramas de secuencia





# Implementación

-Como aspecto a destacar en la implementación de la aplicación web, es el uso de la librería de JavaScript REACT.

-Se han implementado pruebas automatizadas con JUNIT.

-La documentación de la API se ha generado con SWAGGER.

# Pruebas

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 7 |
| Número de pruebas automatizadas | 3 (42%) |

1. **Pruebas sobre la aplicación web.**

Estas pruebas son de tipo sándwich, debido a que se prueba la funcionalidad de varios componentes al mismo tiempo.

|  |  |
| --- | --- |
| ID | **Prueba 1.1** |
| Descripción | Prueba para la detección de errores en la búsqueda de lugares. |
| Entrada | Se introduce un criterio de búsqueda. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase JavaScript y a continuación se muestran en el mapa indicados con un marcador para cada lugar encontrado, |
| Resultado | **EXITO** |
| Automatizada | No |

|  |  |
| --- | --- |
| ID | **Prueba 1.2** |
| Descripción | Prueba para la detección de errores en la búsqueda de lugares por categorías. |
| Entrada | Se introduce un criterio de búsqueda y se seleccionan una o varias categorías. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase JavaScript y a continuación se muestran en el mapa indicados con un marcador para cada lugar encontrado. |
| Resultado | **EXITO** |
| Automatizada | No |

|  |  |
| --- | --- |
| ID | **Prueba 1.3** |
| Descripción | Prueba para la detección de errores en la búsqueda vacia por categorías. |
| Entrada | Se deja en blanco el parámetro de búsqueda, y se seleccionan alguna categoría. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase JavaScript y a continuación se muestran en el mapa indicados con un marcador para cada lugar encontrado. |
| Resultado | **EXITO** |
| Automatizada | No |

|  |  |
| --- | --- |
| ID | **Prueba 1.4** |
| Descripción | Prueba para la detección de errores al abrir los detalles de un lugar marcado en el mapa. |
| Entrada | Se selecciona un marcador después de haber realizado una búsqueda. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase JavaScript y a continuación aparece una división en la pantalla con el nombre del lugar y fotos relacionadas. |
| Resultado | **EXITO** |
| Automatizada | No |

|  |  |
| --- | --- |
| ID | **Prueba 1.5** |
| Descripción | Prueba para la detección de errores en la al cambiar la vista de detalles |
| Entrada | Se selecciona otro marcador después de haber seleccionado uno. |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase JavaScript y a continuación aparece una división en la pantalla con el nombre del lugar y fotos relacionadas del nuevo marcador seleccionado. |
| Resultado | **EXITO** |
| Automatizada | No |

1. Pruebas sobre la API desarrollada.

|  |  |
| --- | --- |
| ID | **Prueba 2.1** |
| Descripción | Prueba para la detección de errores al obtener los itinerarios de un usuario. |
| Entrada | Se hace uso de la librería JUNIT para invocar el método testGetItinerariesPerUser(). |
| Salida esperada | Lista de itinerarios para un usuario determinado. |
| Resultado | **EXITO** |
| Automatizada | Si |

|  |  |
| --- | --- |
| ID | **Prueba 2.2** |
| Descripción | Prueba para la detección de errores al obtener los itinerarios de un del repositorio. |
| Entrada | Se hace uso de la librería JUNIT para invocar el método testGetItineraries (). |
| Salida esperada | Lista de itinerarios disponibles en el repositorio. |
| Resultado | **EXITO** |
| Automatizada | Si |

|  |  |
| --- | --- |
| ID | **Prueba 2.3** |
| Descripción | Prueba para la detección de errores al intentar añadir un itinerario al repositorio. |
| Entrada | Se hace uso de la librería JUNIT para invocar el método testAddItinerary () |
| Salida esperada | El itinerario se añade con éxito y se muestra. |
| Resultado | **EXITO** |
| Automatizada | Si |

# Manual de usuario

## Mashup

Al iniciar la aplicación web se carga un mapa con una barra de búsqueda cerrada.

Al hacer click sobre la barra, esta se despliega y nos da la posibilidad de realizar búsquedas de lugares por categorías y en un lugar concreto del mundo.

Una vez que se ha realizado una búsqueda, aparecen sobre el mapa los lugares cercanos al sitio donde hemos buscado. Si hacemos click sobre ellos aparecerá una vista detallada de lugar, así como fotos relacionadas

## API REST

La documentación de la API REST se ha generado usando la herramienta SWAGGER por lo que se encuentra disponible de forma dinámica en la siguiente dirección: https://planyourtrip.appspot.com/api

# Referencias

[1] *Balsamiq*. <http://balsamiq.com/>. Accedido en Enero 2014.

[2] J. Webber, S. Parastatidis y I. Robinson. *REST in Practice: Hypermedia and Systems Architecture.* O'Reilly Media. 2010.