**TRAVELLO**



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2

José Manuel Tabares Rodríguez (jmtr2000@gmail.com)

Víctor Monteseirín Puig (victormp00@gmail.com)

Horacio García Lergo (geko.hor@gmail.com)

Alexander Sánchez Hossdorf (alexhossdorf@gmail.com)

Tutor: José Antonio Parejo Maestre

Número de grupo: 2

Enlace de la aplicación: <https://travello-271918.appspot.com/>

Enlace de proyecto en projETSII, GitHub o similar: <https://github.com/NebulousAlex/AISS>

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 14/03/2020 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | José Manuel Tabares Rodríguez  Víctor Monteseirín Puig  Horacio García Lergo  Alexander Sánchez Hossdorf |
| 10/04/2020 | 2.0 | -Cambio en la idea del proyecto, APIS, diagramas e interfaces de usuario. | José Manuel Tabares Rodríguez  Víctor Monteseirín Puig  Horacio García Lergo  Alexander Sánchez Hossdorf |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

[1 Introducción 5](#_Toc471899224)

[1.1 Aplicaciones integradas 5](#_Toc471899225)

[1.2 Evolución del proyecto 5](#_Toc471899226)

[2 Prototipos de interfaz de usuario 6](#_Toc471899227)

[2.1 Vista X 6](#_Toc471899228)

[2.2 Vista Y 6](#_Toc471899229)

[3 Arquitectura 7](#_Toc471899230)

[3.1 Diagrama de componentes 7](#_Toc471899231)

[3.2 Diagrama de despliegue 7](#_Toc471899232)

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

[3.4 Diagrama de clases 7](#_Toc471899234)

[3.5 Diagramas de secuencia 7](#_Toc471899235)

[4 Implementación 8](#_Toc471899236)

[5 Pruebas 9](#_Toc471899237)

[6 Manual de usuario 10](#_Toc471899238)

[6.1 Mashup 10](#_Toc471899239)

[6.2 API REST 10](#_Toc471899240)

[Referencias 11](#_Toc471899241)

# Introducción

Muchas personas suelen viajar sin planear nada previamente, y cuando llegan al lugar de destino no saben que hacer ni a donde ir.

Existen muchos restaurantes, pero ¿dónde están? ¿Tendrán buenas críticas? Vas a alguno y ves el menú que tiene, pero tienen una moneda diferente ¿cuánto cuesta? ¿Será muy caro?

Tal vez quieras ir a ver un espectáculo, pero ¿habrá alguno? ¿Cuál será? ¿Hará buen tiempo atmosférico? ¿Y si ya tenemos planes para ese día, pero no nos acordamos?

Travello plantea una solución para todos estos problemas con un mashup en el que podremos buscar que restaurantes hay por la zona, cuanto costarán las cosas, si hay algún evento que nos puede interesar e incluso saber que tiempo habrá ese mismo día. Y sin olvidarnos de que podremos guardar todos nuestros planes en Google Calendar para poder aprovechar al máximo nuestro viaje. Además de estar completamente en inglés para un uso internacional, y no exclusivamente para personas con conocimiento de lengua española.

## Aplicaciones integradas

|  |  |
| --- | --- |
| **Nombre aplicación** | **URL documentación API** |
| **Google Calendar** | <https://developers.google.com/calendar/overview?hl=es> |
| **Cambio.Today** | <http://cambio.today/api> |
| **Triposo** | <https://www.triposo.com/api/documentation/20200405/> |
| **TicketMaster** | <https://developer.ticketmaster.com/products-and-docs/apis/getting-started/> |
| **OpenWeatherMap** | <https://openweathermap.org/api> |

Tabla 1. Aplicación integradas

**Google Calendar:**

Podremos ingresar en nuestra cuenta de Google calendar para poder ver, añadir o eliminar eventos para facilitar la organización de nuestro viaje.

**Cambio.Today:**

Implementaremos la API de Cambio.Today, lo que nos permitirá poder realizar un cambio de moneda sencillo y rápido. Tendremos que ingresar la abreviatura de la moneda que tenemos y a la que queremos pasarla y la cantidad. Por ejemplo, tenemos 5 EUR y queremos saber cuantos USD son.

**Triposo:**

Podremos buscar restaurantes en la ciudad que queramos y obtener su puntuación, ver sus comentarios y fotos para saber si nos interesa ir o no.

**TicketMaster:**

TicketMaster nos permitirá acceder a todos los eventos de la ciudad dada que se producirán en las fechas, junto con una foto y una breve descripción. Además, proporcionará un link al evento en concreto en caso de que el cliente le interese.

**OpenWeatherMap:**

Con OpenWeatherMap podremos conseguir el tiempo atmosférico (lluvioso, despejado, nublado) que habrá ese mismo día en la ciudad que usted busque.

## Evolución del proyecto

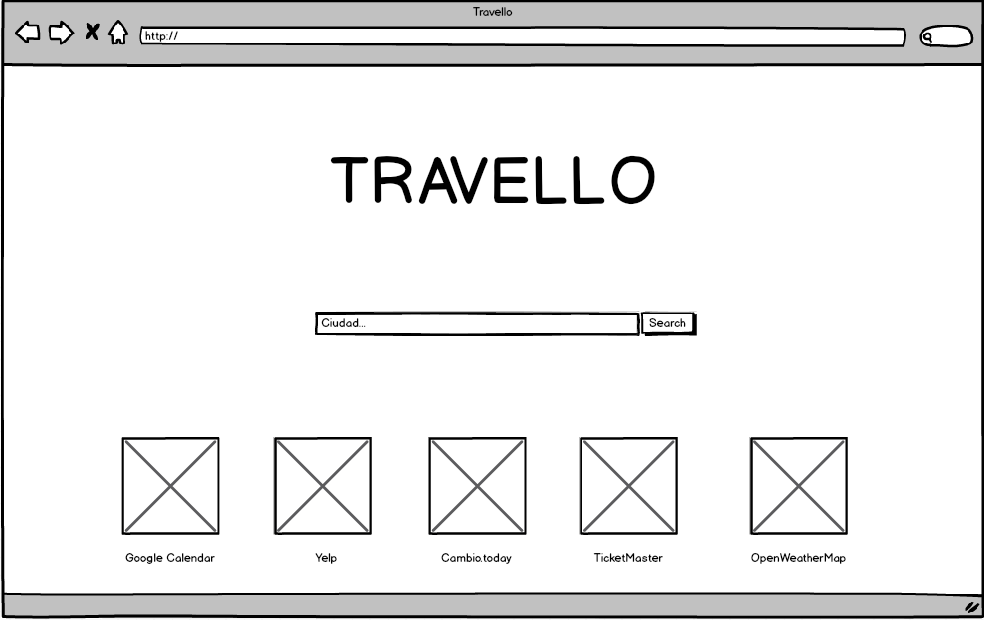
En un principio nuestra idea principal era hacer un mashup que organizase viajes pudiendo buscar vuelos, hoteles, transporte para poder moverse por la Ciudad, etc. Pero debido a que no existen APIs gratuitas o disponibles que pudiesen abarcar esos ideales y que algunas nos dieron algunos problemas, tuvimos que cambiar la idea del trabajo.

Para poder rescatar algunas APIs que ya teníamos (OpenWeatherMap y Cambio.Today) decidimos hacer un mashup con el mismo concepto, organizar un viaje, pero en vez de hacerlo con tiempo, hacerlo ya una vez en la ciudad. Buscamos APIs disponibles y encontramos las anteriormente mencionadas, las cuales si funcionan perfectamente.

# Prototipos de interfaz de usuario

## Inicio

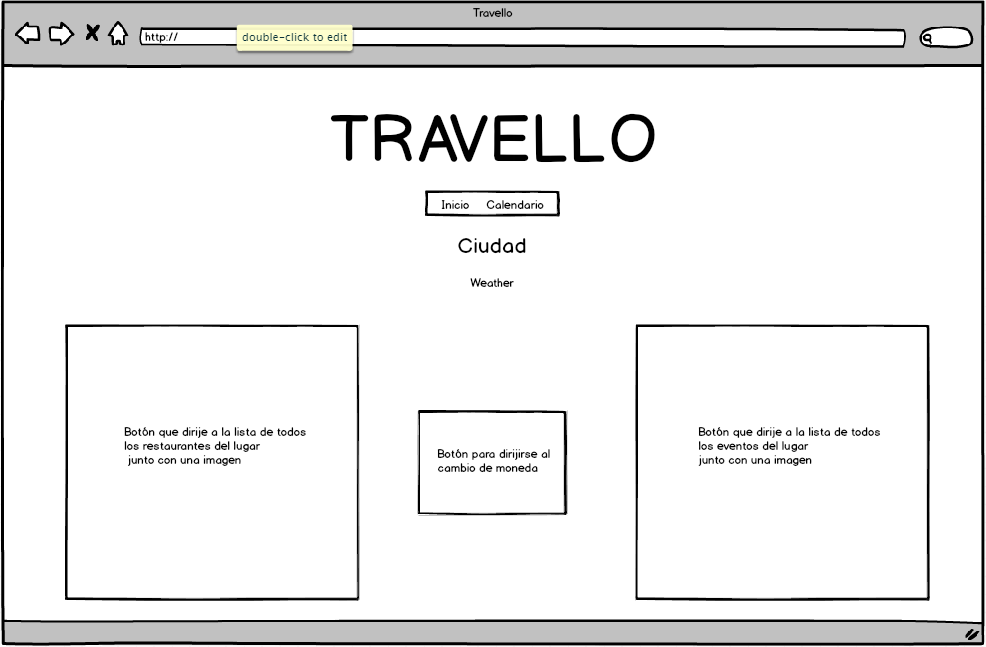
La pagina inicial es lo primero que veremos nada mas entrar, en ella tenemos todos los iconos de cada API usada junto al link que lleva a su página y una barra de búsqueda para introducir el lugar que quieras.



## Página principal

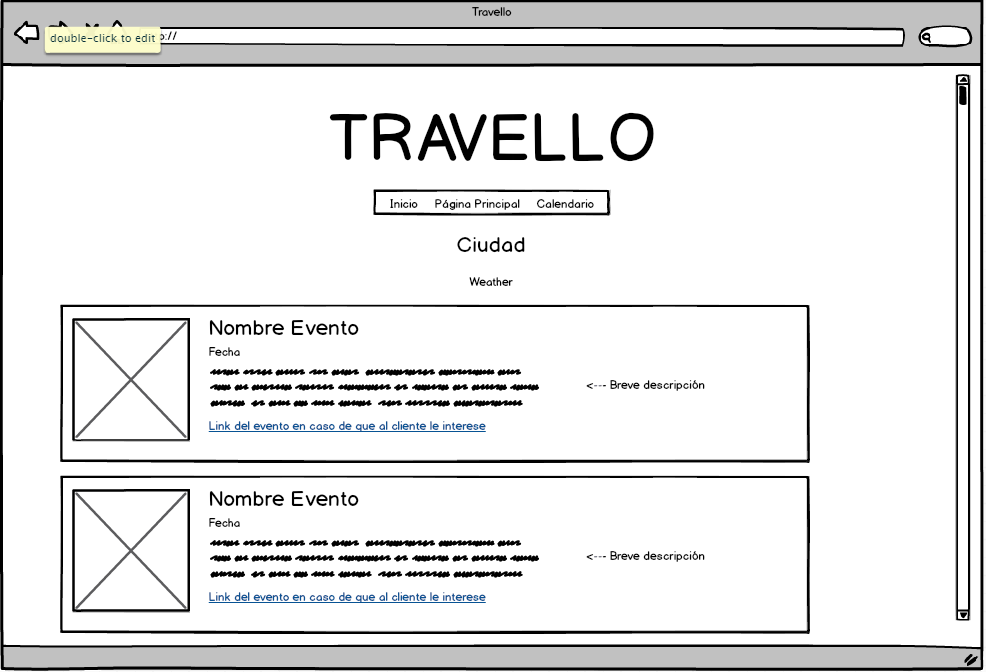
En la pagina principal tendremos un menú en el que podremos volver al inicio o ir al calendario. También tendremos que tiempo atmosférico habrá ese día en la ciudad.

Debajo podemos ver tres botones grandes que te llevan a diferentes páginas, uno a la lista de eventos, otro a la lista de restaurantes y otro al conversor de divisas.



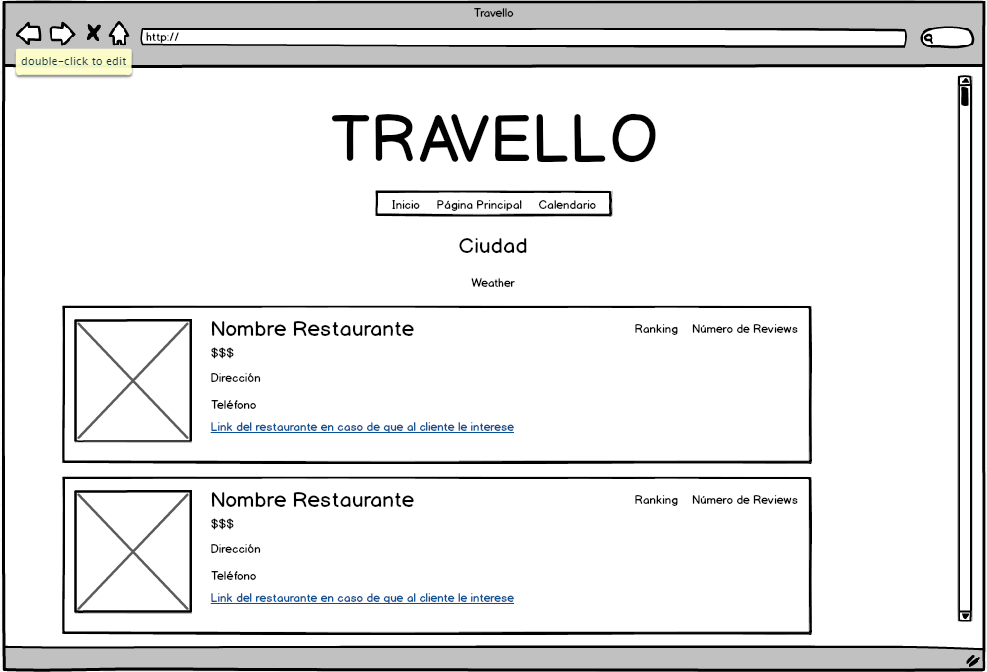
## Página de eventos

Aquí podemos ver la lista con eventos en la ciudad, junto a su fecha, una breve descripción, una foto y un link hacia el evento por si se quiere más información.



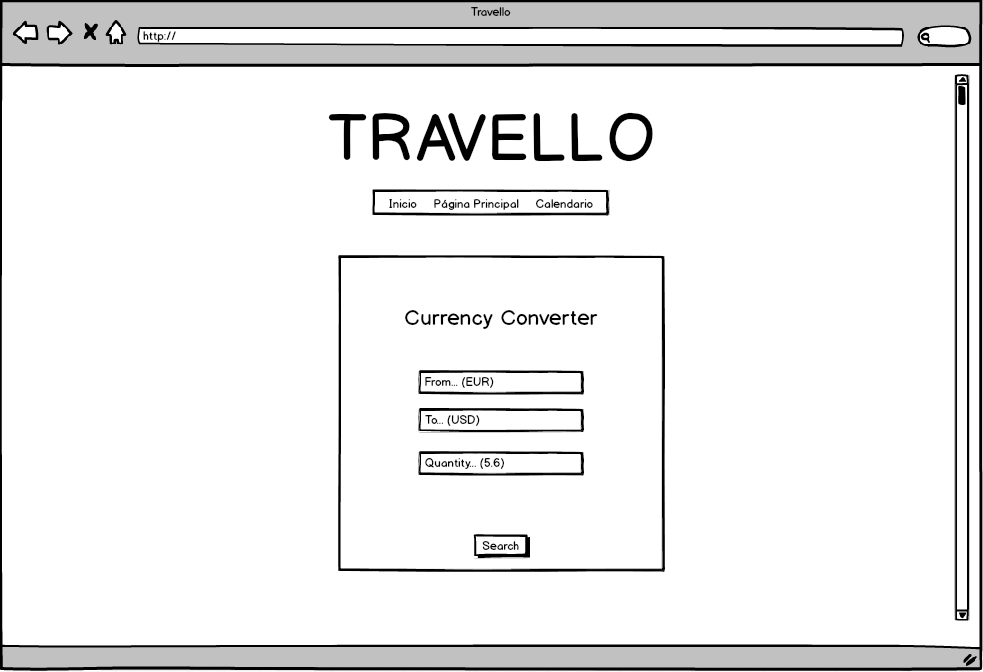
## Página de restaurantes

Es exactamente igual que la pagina de eventos solo que aquí se muestra una lista de restaurantes, junto a una foto, el ranking (entre 1 y 5), el numero de reviews, el rango de precios ($ a $$$$), su dirección y teléfono, y un link para ir al restaurante en Triposo.



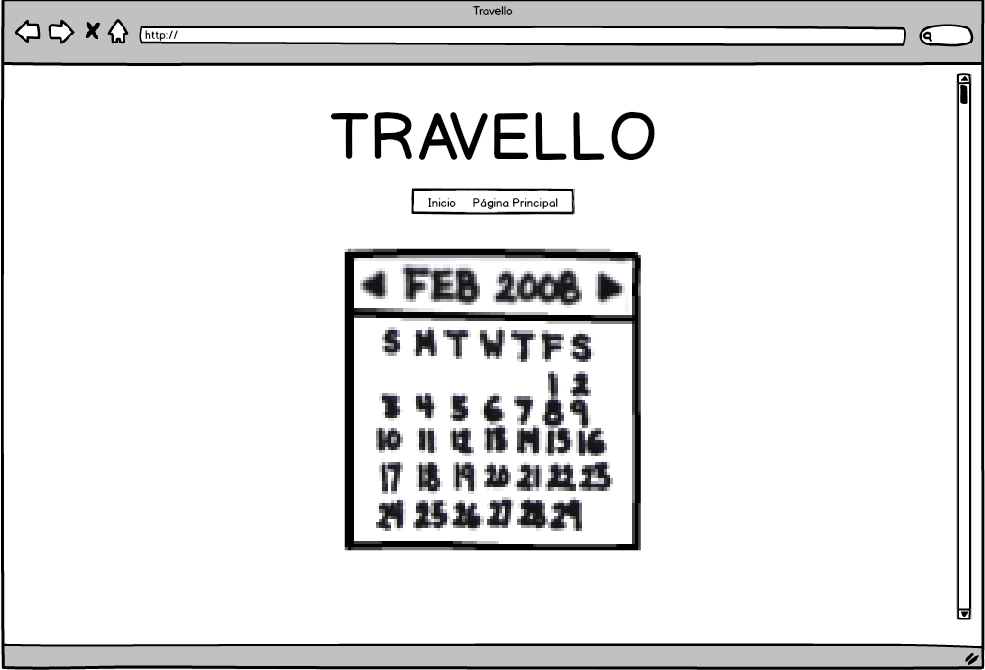
## Conversor de moneda

Introducimos la moneda que tenemos, la que queremos conseguir, la cantidad y al pulsar search obtenemos el cambio.



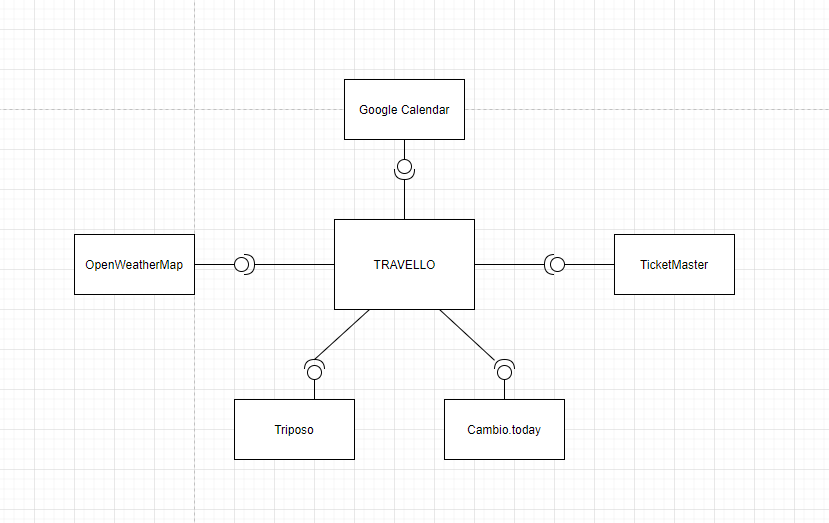
## Calendario

Al pulsar calendario, iniciaremos sesión en Google Calendar y podremos ver el calendario y modificarlo.



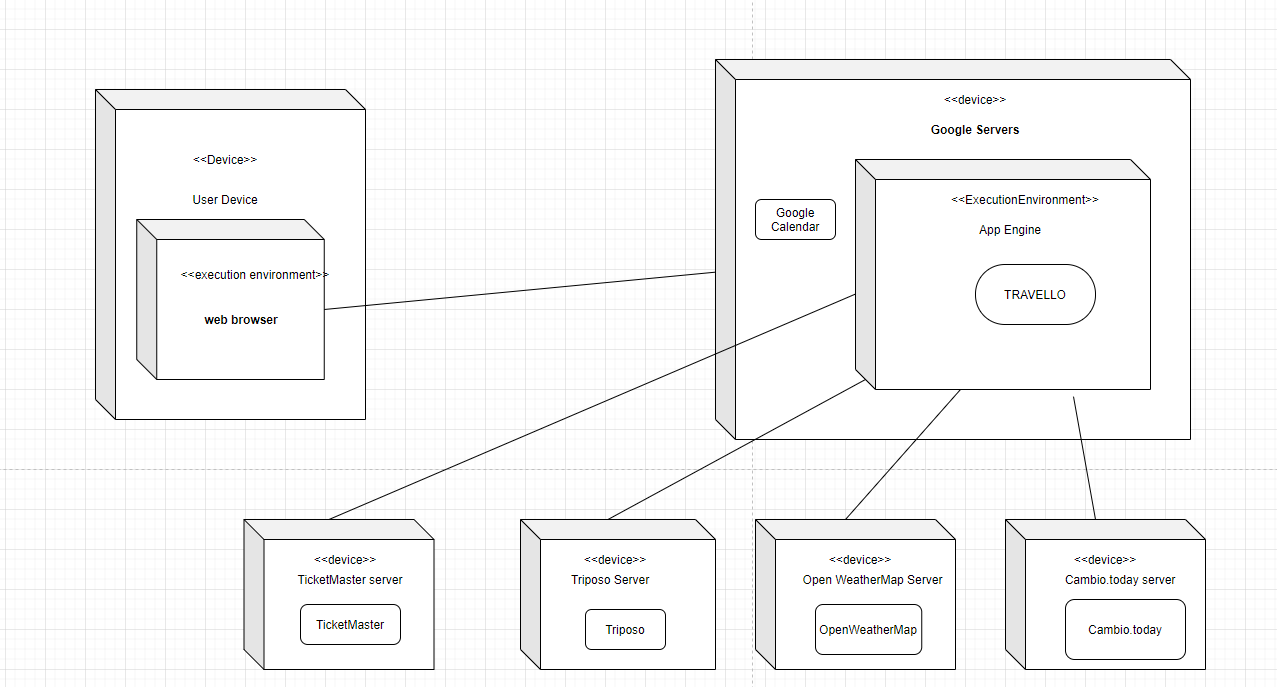
# Arquitectura

## Diagrama de componentes



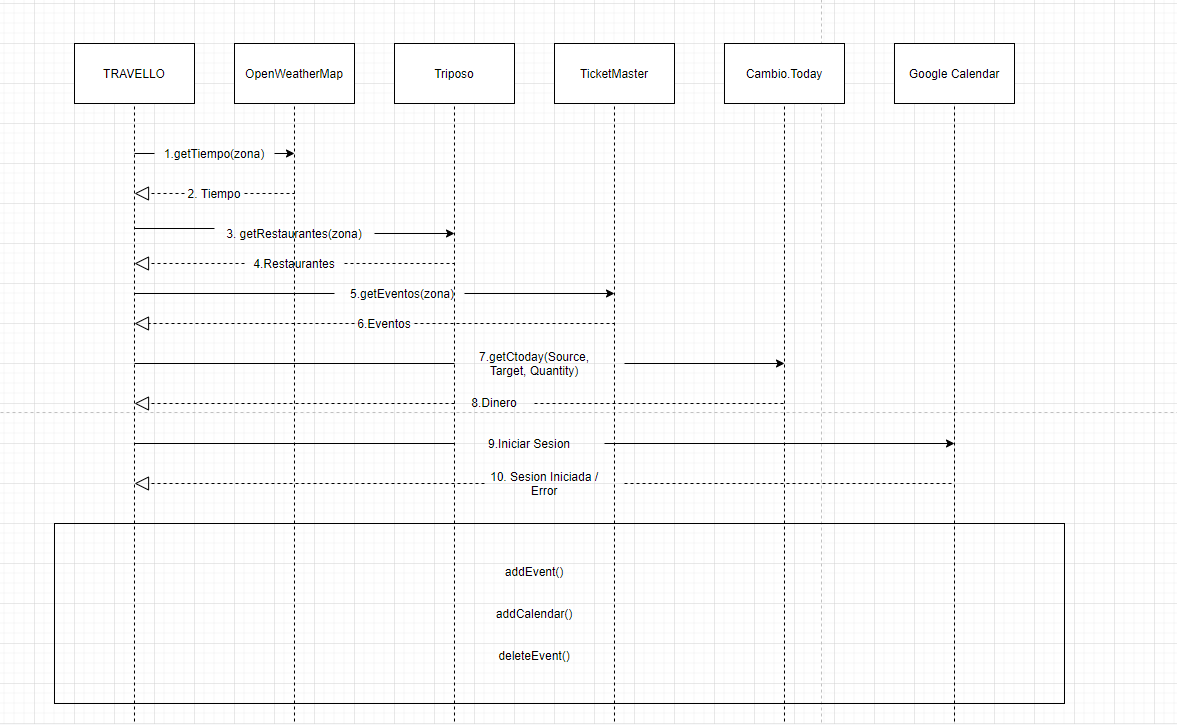
## 

## Diagrama de despliegue

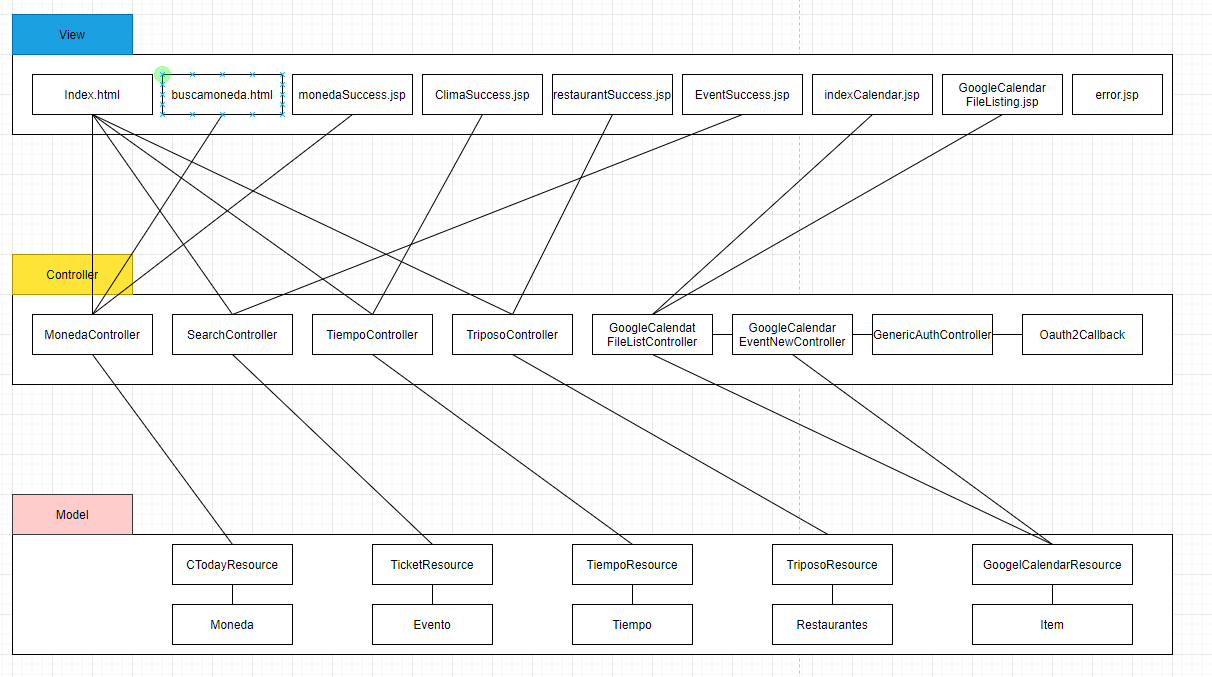


## 

## Diagrama de secuencia de alto nivel



## Diagrama de clases



## Diagramas de secuencia

**Cambio.Today**

**TicketMaster**

**OpenWeatherMap**

**Triposo**

# Implementación

Hemos conseguido implementar correctamente las 5 APIs siendo una por Oauth.

Con poner el nombre de una ciudad podremos obtener que tiempo atmosférico hay ese día en ese lugar, además de poder ver varios restaurantes de la ciudad junto a su ranking y su posición con un link a Google Maps. También podemos ver que eventos habrá en ese lugar junto a su fecha y varios links para poder informarte si deseas comprar entradas, y un botón para poder agregar dicho evento a tu calendario.

Además de todas estas funciones, también disponemos de un conversor de divisas para que te sea más fácil poder comprar en la ciudad. Y sin olvidarnos de una lista de puntos de interés para poder visitar, en el que la gente podrá ir añadiendo sitios si lo desea.

Con Google Calendar, además de poder añadir un evento desde la lista de eventos, también podremos añadir uno nosotros mismos, actualizarlo o eliminarlo.

Finalmente, estamos muy contentos y orgullosos con la estética de la página y el como está organizada.

# Pruebas

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 8 |
| Número de pruebas automatizadas | 8 (100%) |

|  |  |
| --- | --- |
| ID | **Prueba 1** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en CToday usando servicios RESTful. |
| Entrada | Invocamos al servicio para obtener el cambio de moneda usando la URI  "https://api.cambio.today/v1/quotes/"+ URLEncoder.*encode*(source, "UTF-8")  +"/"+ URLEncoder.*encode*(target, "UTF-8")  +"/json?quantity="+ URLEncoder.*encode*(quantity, "UTF-8")  +"&key="+ ***CTODAY\_API\_KEY***; |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 2** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Ticket Master usando servicios RESTful. |
| Entrada | Invocamos al servicio para obtener los eventos de la ciudad usando la URI  "http://app.ticketmaster.com/discovery/v2/events.json?city="+queryFormatted+"&apikey="+***TICKET\_API\_KEY***+"&callback=myFunction"; |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 3** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Triposo usando servicios RESTful. |
| Entrada | Invocamos al servicio para obtener los restaurantes de la ciudad usando la URI  "http://www.triposo.com/api/20200405/poi.json?tag\_labels=poitype-Restaurant&poitype-Restaurant\_score=%3E=3&location\_id=" + queryFormatted + "&order\_by=-score&fields=name,images,coordinates,score&account=" + ***secretAccount*** + "&token=" + ***APIkey***; |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 4** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Open Weather Map usando servicios RESTful. |
| Entrada | Invocamos al servicio para obtener el tiempo atmosferico de la ciudad usando la URI  "http://api.openweathermap.org/data/2.5/weather?q="+q+"&appid="+***Tiempo\_API\_KEY***; |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 5** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Google Calendar usando servicios RESTful. |
| Entrada | Invocamos al servicio para insertar un evento en el calendario usando la URI  "https://www.googleapis.com/calendar/v3/calendars/"+CalendarId+"/events?alt=json" |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 6** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Google Calendar usando servicios RESTful. |
| Entrada | Invocamos al servicio para obtener la lista de eventos de tu calendario usando la URI  "https://www.googleapis.com/calendar/v3/calendars/"+CalendarID+"/events?alt=json" |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 7** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Google Calendar usando servicios RESTful. |
| Entrada | Invocamos al servicio para eliminar un evento de tu calendario usando la URI  "https://www.googleapis.com/calendar/v3/calendars/"+CalendarID+"/events/"+eventID |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

|  |  |
| --- | --- |
| ID | **Prueba 8** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Google Calendar usando servicios RESTful. |
| Entrada | Invocamos al servicio para actualizar un evento de tu calendario usando la URI  "https://www.googleapis.com/calendar/v3/calendars/"+CalendarID+"/events/"+eventID |
| Salida esperada | Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla. |
| Resultado | **EXITO** |
| Automatizada | Sí |

# Manual de usuario

## Mashup

Indique textualmente e **incluyendo capturas de pantalla** el manual de uso del mashup.

## API REST

Incluir enlace a 1) fichero YAML de la API en OAS (Swagger) y 2) documentación interactiva generada a partir de la especificación OAS.

# 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.