**TRAVELLO**

<Logo del Proyecto>

Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2019-2020

José Manuel Tabares Rodríguez (jmtr2000@gmail.com)

Víctor Monteseirín Puig (victormp00@gmail.com)

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

Alexander Sánchez Hossdorf (alexhossdorf@gmail.com)

Tutor: Javier Troya Castillo

Número de grupo: 2

Enlace de la aplicación: <Enlace de la aplicación en AppEngine>

Enlace de proyecto en projETSII, GitHub o similar: <Enlace proyecto> (opcional)

**Historial de versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Detalles | Participantes |
| 14/03/2014 | 1.0 | - Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue. | Jose Manuel Tabares  Víctor Monteseirín  Ale Hossdorf  Horacio García |
|  |  | <Mencionar los cambios más significativos con respecto a la versión anterior> |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

ferhbgeor

* Introducción

Existen numerosas app de hoteles que muestran el lugar perfecto en el que deseamos estar pero no el camino que vincula al hotel y el lugar en el que nos encontremos actualmente, por tanto, existe este problema que podremos solucionar mediante nuestra idea.

El objetivo de esta app es relacionar los hoteles que estén en un determinado lugar que se busque, en una fecha determinada. Esta app nos mostrará el mejor camino y forma para llegar según nuestras condiciones. También nos mostrará nuestro hotel ideal y toda la información necesaria con ello. Pretendemos hacer un mashup que combine diferentes apis de hoteles en los que podemos hospedarnos, junto con apis de vuelos, taxis, autobuses y trenes.

* Aplicaciones integradas

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

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| Facebook | <https://developers.facebook.com/docs/graph-api/using-graph-api/> |
|  |  |
|  |  |

Tabla 1. Aplicación integradas

* Evolución del proyecto

Es habitual que la aplicación final diste mucho de la idea inicial. Puede que la idea fuese muy compleja, no haya sido posible integrar alguna de las aplicaciones o alguno de los miembros del grupo haya abandonado. Explicar en esta sección cuál ha sido la evolución del proyecto, problemas, cambios, decisiones, etc.

* 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 X

Descripción textual de la vista



Figura 1. Prototipo de interfaz de usuario de la vista X

* Vista Y

* Arquitectura

Insertar los diagramas UML de componentes y de despliegue de la aplicación. Describir textualmente

* 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

Diagrama UML de clases indicando la distribución de las clases entre las distintas capas, según el patrón MVC.

* Diagramas de secuencia

Diagramas UML de secuencia ilustrando la comunicación entre vistas, controladores y clases del modelo.

* Implementación

Describir brevemente los aspectos de la implementación que creen da más mérito al trabajo. Añadir algún fragmento de código si se considera oportuno.

* Pruebas

Documentar las pruebas realizadas a la aplicación. Justificar textualmente la estrategia de pruebas seguida y por qué (ej. pruebas incrementales ascendentes).

Indicar el número total de pruebas realizadas y cuáles de ellas han sido automatizadas mediante JUnit.

|  |  |
| --- | --- |
| Resumen |  |
| Número total de pruebas realizadas | 25 |
| Número de pruebas automatizadas | 20 (80%) |

|  |  |
| --- | --- |
| ID | **Prueba 1** |
| Descripción | Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful. |
| Entrada | Se hace uso de la librería XXX para invocar al servicio usando la URI YYY desde nuestra aplicación. |
| 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.