**<FITNASSIO >**

<Logo del Proyecto>

Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso <2º>

<Álvaro Rubia Tapia> (<alvrubtap@gmail.com>)

<Laura Portillo Soler> (<lauraportillosoler@gmail.com>)

<Miguel Ángel Gómez Gómez> (<miguelangelgomezx2@gmail.com>)

<Natalia Crespo Bravo> (<nataliacrespo98@gmail.com>)

Tutor: Javier Troya

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. | Álvaro Rubia Tapia  Laura Portillo Soler  Miguel Ángel Gómez Gómez  Natalia Crespo Bravo |
|  |  | <Mencionar los cambios más significativos con respecto a la versión anterior> |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Presentar y motivar el problema que se va a resolver con la aplicación. ¿Por qué es un problema importante?

Describir el mashup desarrollado y como éste le da solución al problema mencionando anteriormente. ¿Por qué es un buen mashup?

Debido al auge de la influencia de las redes sociales y los hábitos saludables de las personas en la actualidad, nos surgió una idea para hacerles la vida más fácil con la siguiente aplicación llamada “FITNASSIO”:

* **Características**
  + Sugerencias de rutas dónde hacer deporte, llevándote a sitios de interés urbano o rural.
  + Compartición de rutas y fotos de los recorridos.
  + Consulta meteorológica.
  + Filtro y notificación de eventos deportivos según preferencias.

Como se puede observar, no existen aplicaciones que reúnan estas carácteristicas, es por ello que podemos considerar el hecho de hacer un mashup con APIs que abarquen los ámbitos propuestos como una buena idea. Por otra parte la mayoría de APIs pertenecen a aplicaciones de uso cotidiano, lo cual ayuda a preveer la usabilidad que tendrá.

* **APIs del mashup**
  + GoogleMaps (localización de lugares, intercambio de rutas...).
  + Instagram Graphs (fotos de los recorridos).
  + Moves (seguimiento de actividad diaria).
  + Endomondo (consulta de eventos, planificación de rutinas, nutrición...).
  + WorldWeather Online (tiempo atmosférico).

## Aplicaciones integradas

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

* + GoogleMaps (localización de lugares, intercambio de rutas...).
  + Instagram Graphs (fotos de los recorridos).
  + Moves (seguimiento de actividad diaria).
  + Endomondo (consulta de eventos, planificación de rutinas, nutrición...).
  + WorldWeather Online (tiempo atmosférico).

|  |  |
| --- | --- |
| Nombre aplicación | URL documentación API |
| Moves | https://www.programmableweb.com/api/moves |
| GoogleMaps | https://cloud.google.com/maps-platform/?hl=es |
| Instagram Graphs | https://developers.facebook.com/docs/instagram-api/ |
| Endomondo | https://bit.ly/2TwS9OV |
| WorlWeather Online | https://www.worldweatheronline.com/developer/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 principal

Prototipo de vista de inicio de sesión o registro.

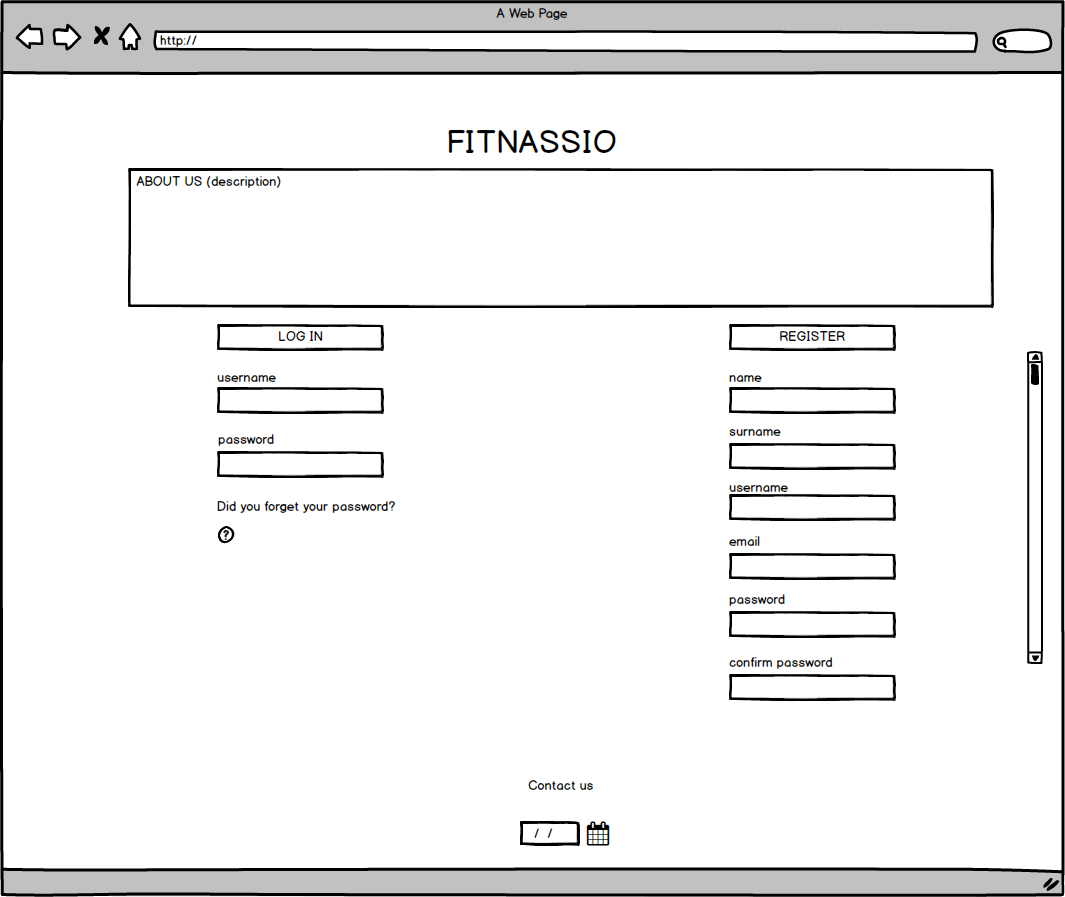


Figura 1. Prototipo de interfaz de usuario de la vista PRINCIPAL

## 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

Indique la documentación de la API REST (contrato) implementada [2]. Cómo mínimo, la API debería incluir:

1. Protocolo de aplicación empleado por el servicio.
2. URIs para invocar a las operaciones del servicio.
3. Formato empleado para las representaciones de los recursos.
4. Códigos de estado empleados por el servicio.
5. Ejemplos de uso.

Está información también debe facilitarse en formato HTML como parte de la aplicación.

# 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.