Extreme Android

Programa

Viernes, 18 de julio
20:00 Anuncio retos programación
20:30 Inicio programación

Sábado, 19 de julio
Programación
Uso libre de las instalaciones
18:00 Birra+Tapa en la Taberna Ixarso (usad el ticket!)
21:00 Cena en el restaurante

Domingo, 20 de julio
10:00 Cierre Admisión de Aplicaciones
Test de Aplicaciones por parte del jurado
12:00 Votaciones
13:00 Entrega de Premios

Contacto

Organización y apoyo técnico
José Ignacio Latorre (j.i.latorre@gmail.com)
David Fuentes Ruiz (support@benasque.org) (Tlf. urgencias: 615094163)
Ismael Ruiz (ismaxxl@gmail.com)
Ayuntamiento
Marcos Liminiana Perruc (mliminiana@benasque.es)

Informaciones

Conectividad y equipamiento
- Conexión Wi-Fi y puntos ethernet en todo el edificio conectados a 30Mbps/1Mbps.
- 10 sistemas para uso público (usuario: participant passwd: benasque13)
- 4 Linux con eclipse + ADT + SDK android

Entrega de Aplicaciones
- Todos los retos deberán ser entregados antes de las 10:00h del domingo.
- Tanto los proyectos como las aplicaciones deberán nombrarse como sobrenombre.apk donde:

sobrenombre = alias anónimo identificativo del programador o equipo de programadores.

- La URL para subir los retos desarrollados es:
http://benasque.org/2014android/cgi-bin/upload.pl

Test de aplicaciones
Todos los retos deberán ser programados para versiones android 4.0 o superior.

Valoración
Un jurado valorará el grado de consecución del reto planteado, desde los mínimos a los máximos. Los aspectos de programación fuera de los requisitos de los retos serán bienvenidos pero secundarios para la valoración final de la aplicación.

Jurado
D. Fuentes (CCBPP), J. I. Latorre (CCBPP), M. Liminiana Perruc (Ayuntamiento), I. Ruiz.

R1: Mi Bus

Descripción breve

Plataforma de seguimiento de buses escolares y/o línea.

Características de la apk

Esta apk deberá permitir el seguimiento de autobuses escolares/línea (usando el teléfono del conductor/profesor como localizador) por otros usuarios y mostrarlo en un mapa.

Mínimos

La aplicación tendrá dos grandes secciones principales: la del teléfono que será rastreado (BUS en servicio) y la de rastreo de buses en funcionamiento (FAMILIARES).

En la sección BUS se introducen los datos relativos a un recorrido: nombre del recorrido, clase, hora de inicio y hora de fin de trayecto, trayecto periódico y frecuencia de paso, paradas... (ej: “Excursión Escuelas Pías a Benasque” o “Recorrido de las mañanas”) y un código libre de 8 dígitos previamente acordado.

En la sección FAMILIARES, se selecciona el código asociado a la excursión a rastrear y automáticamente se obtienen los datos de la excursión y la posición dinámica en un mapa del bus. En cualquier momento BUS y FAMILIARES pueden cancelar el seguimiento. Debe haber un backend para gestionar el cruce de datos.

La apk debe además ofrecer un servicio de notificaciones privado. BUS tiene un campo en el que añade un comentario (ej: “¡Hemos llegado a Benasque!”). Ese mensaje aparece en FAMILIARES junto a la hora de emisión y se suma al histórico de mensajes para ese seguimiento.

Deseable

Añadir track recorrido.
Filtrado de autobuses por escuela/empresa.
El servidor backend puede ofrecer un mapa general de todos los seguimientos y un histórico.
Estimación del tiempo de llegada a una parada.
Geoposicionamiento de paradas en el mapa.

R2: API Semantic Analisys + cliente

Descripción breve

Generar una API + cliente de análisis semántico

Características de la apk

Deberemos generar una API con los métodos que se detallan a continuación y un cliente que permita testear todas las funcionalidades desarrollas en nuestra API.

Se valorará muy positivamente la velocidad algorítmica del cliente.

Material de soporte: http://benasque.org/2014android/semanticData/
(ficheros de ejemplo useless, plurals en ES y EN)

Dictionary
({{palabra/s-frecuencia}, id, idioma, n})

d*ictionary* **create (fichero / URL / conjunto de palabras , n)**
{desc: Crea nuevo dictionary a partir de la fuente donde cada elemento está compuesto por n palabras consecutivas}

void **clean (dictionary, fichero useless / conjunto de palabras)**
{desc: Elimina las palabras de un dictionary)

void **cleanse (dictionary , conjunto de signos)**
{desc: elimina los signos del dictionary}

void **no_numbers (dictionary, num_de_dígitos)**
{desc: dictionary eliminando caracteres numéricos de <= que n num_de_dígitos}

void **to_lower(dictionary)**
{desc: pasa un dictionary a minúsculas}

void **to_capital(dictionary)**
{desc: pasa dictionary a mayúsculas}

void **plurals (dictionary , fichero 'plurals')**
{desc: Unifica plurales / singulares de un dictionary}

dictionary **relevant (dictionary, fichero/conjunto de palabras)**
{desc: Devuelve un dictionary nuevo con sólo las palabras relevantes filtradas}

void **sort (dictionary, pattern)**
{desc: Ordena un dictionary según pattern = alfabéticamente, inverso alfabéticamente }

void **set_language(dictionary, lenguage “en, “es”,”fr”)**
{desc: Setea el idioma de un dictionary}

'*idioma*' **get_language(dictionary)**
{desc: Devuelve el idioma de un dictionary. Si es nulo debe extraer el idioma}

void **set_id(dictionary, id)**
{desc: Setea el identificador de un dictionary}

'*id*' **get_id(dictionary)**
{desc: Devuelve el id de un dictionary}

dictionary **common (dictionary)**
{desc: Devuelve un dictionary formado únicamente por las palabras que aparezcan en el dictionary del sistema}

dictionary **excepcional (dictionary)**
{desc: Devuelve un dictionary sin las palabras del dictionary del sistema}

frecuencia **find_word (dictionary, palabra)**
{desc: Devuelve la frecuencia de la palabra en un dictionary}

void **add_word (dictionary, palabra, frecuencia)**
{desc: Añade una palabra con frecuencia "frecuencia" a un dictionary }

void **kill_word (dictionary, palabra / conjunto de palabras)**
{desc: elimina una palabra o conjunto de palabras y su frecuencia}

dictionary **probabilities (dictionary)**
{desc: Devuelve un nuevo dictionary con las frecuencias f_i transformadas en probabilidades p_i}

$$p_i = \frac{f_i}{\sum_j f_j}$$

'entropía' **entropy (calcula la entropía)**
{desc: devuelve la entropía de Shannon de un dictionary a partir de las probabilidades de cada palabra}

$$S = -\sum_i p_i \log p_i$$

dictionary **sum (dictionary1, dictionary2)**
{desc: devuelve un dictionary suma de dos palabras y frecuencias}

dictionary **subtract (dictionary1, dictionary2)**
{desc: substrae las palabras de un dictionary a las de otro}

dictionary **total (library)**
{desc: crea un dictionary suma de todos los dictionaries de una librería, sumando también las frecuencias para cada palabra}

float distances (dictionary1, dictionary2)
{desc: calcula la distancia H_1 entre pares de dictionaries a partir de las probabilidades de sus palabras p_i y q_i (si una palabra de un diccionario no aparece en el otro, se le asigna probabilidad 0)

$$H_1 = -\sum_i |p_i - q_i|$$

Library
({dict_1,...,dict_n})

library **create (dict_1 … dict_n)**
{desc: crea una nueva librería a partir de un conjunto de dictionarys }

void **add (dictionary, library)**
{desc: Añade un nuevo dictionary a la librería}

void **purge (dictionary, library)**
{desc: Elimina un nuevo dictionary a la librería}

dictionary **total (library)**
{desc: crea un dictionary suma de todos los dictionaries de una librería, sumando también las frecuencias para cada palabra}

float **total_sigma (library)**
{desc: calcula la desviación estándar en la frecuencia de cada palabra de la librería en función de su distribución en los diccionarios parciales)

void **near (library , dictionary_id)**
{desc: ordena una librería por mínima distancia a un dictionary}

void **far (library, dictionary_id)**
{desc: ordena una librería por máximo distancia a un dictionary)

dictionary **mutual_information (library)**
{desc: Devuelve un dictionary con la información mutua de todos los diccionarios que forman una librería (palabras / frecuencias) }

boolean **find_word (library , word)**
{desc: busca una palabra en la librería}

void **add_word (library , word, frecuencia)**
{desc: suma una palabra a todos los diccionarios de una librería}

void **kill_word (library)**
{desc: elimina una palabra/frecuencia de todos los diccionarios de una librería)

dictionary **shared_words (libarary)**
{desc: Devuelve un diccionario con las palabras comunes en una librería}

void **expecional words (library)**
{desc: elimina las palabras comunes en una librería}

dictionary **over_the_mean (library)**
{desc: detecta palabras con frecuencias > 2 sigmas del total en una librería}

dictionary **under_the_mean (library)**
{desc: detecta palabras con frecuencias < 2 sigmas del total en una librería}

probability **strategies (library , conjunto de palabaras)**
{desc: dada una lista de palabras, calcula su suma de probabilidades en en una librería}

****

APP Cliente

Crear una app - cliente que utilice las herramientas programadas para el análisis semántico. A continuación os detallamos un posible ejemplo:

Comparador semántico de webs

App que permite analizar, comparar urls y presentar resultados del análisis semántico mediante gráficos. Se valorará positivamente la correcta visualización en Tablets.

1. Inserción de urls y su gestión (añadir, borrar, to_lower, set_language, ...)
2. “Word Cloud” de cada dictionary, con opción de filtro por useless, relevant, excepcional
3. Visualización de distancias entre dictionarys (urls) por parejas (una es el Total)
4. Comparación de palabras excepcionales
5. Tensiones: dada una palabra se comparan todos los dictionarys
6. Visión de dictionarys individuales , con frecuencia y probabilidad