



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y
COMPUTACIÓN

Parsing XML with Haskell

Autor:

Adriana RODRÍGUEZ

Profesor:

Ing. Javier TIBAU

19 de enero de 2014

Índice general

Capítulo 1

Introducción

XML es un lenguaje de marcado que permite almacenar datos, para ello esta conformado de una estructura abstracta para que así se puedan trabajar con datos grandes.

En este caso, el propósito de este documento es dar a conocer como en el lenguaje Haskell se puede trabajar para la administración de archivos XML y así mostrar datos que sean específicos para el usuario.

Para llevar un mejor control en los datos, decidimos trabajar por medio del *parsing* que permite que los datos sean trabajados de una manera mas textual en la observación de todo el documento en si.

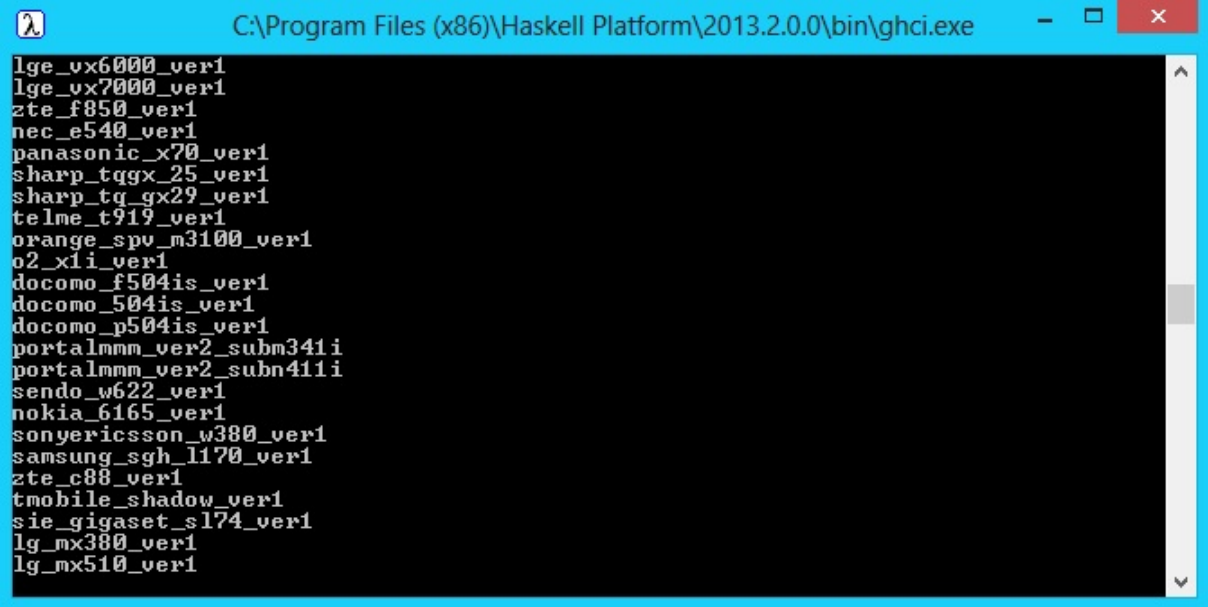


Figura 1.1: Logo de Haskell.

Capítulo 2

Alcance

El alcance del proyecto es poder realizar un procesamiento de la información que está contenida en el archivo *wurfl23.xml* y mostrar datos correspondientes a los dispositivos que cumplan con ciertas características dichas por el usuario.



```
C:\Program Files (x86)\Haskell Platform\2013.2.0.0\bin\ghci.exe
lge_vx6000_ver1
lge_vx7000_ver1
zte_f850_ver1
nec_e540_ver1
panasonic_x70_ver1
sharp_tqgx_25_ver1
sharp_tq_gx29_ver1
telme_t919_ver1
orange_spv_m3100_ver1
o2_xli_ver1
docomo_f504is_ver1
docomo_504is_ver1
docomo_p504is_ver1
portalmmm_ver2_subm341i
portalmmm_ver2_subn411i
sendo_w622_ver1
nokia_6165_ver1
sonyericsson_w380_ver1
samsung_sgh_l170_ver1
zte_c88_ver1
tmobile_shadow_ver1
sie_gigaset_sl74_ver1
lg_mx380_ver1
lg_mx510_ver1
```

Figura 2.1: Ejemplo del capability.

Capítulo 3

Descripción

El objetivo principal del proyecto es poder comprender como trabaja Haskell, como interpreta los datos y las funcionalidades que este posee. Iniciando por la recursividad con la que este lenguaje trabaja, para ello crearemos ciertas funciones que nos ayuden a manejar los datos correctamente; y así manipularlos manualmente.

Los archivos XML trabajan por medio del marcado, llevan un control de los grupos de información para así mostrarlos como *elementos* que proporcionan una lista de estos elementos. El archivo antes mencionado, está conformado por una serie de elementos principales, estos son: Device, Group y los capability que son las listas principales que nos ayudarán a manipular los datos.

En el caso de los Device están conformados por:

- id
- user agent
- fall back

A su vez, los Group están conformados por:

- id

Y por último los capability están conformados por:

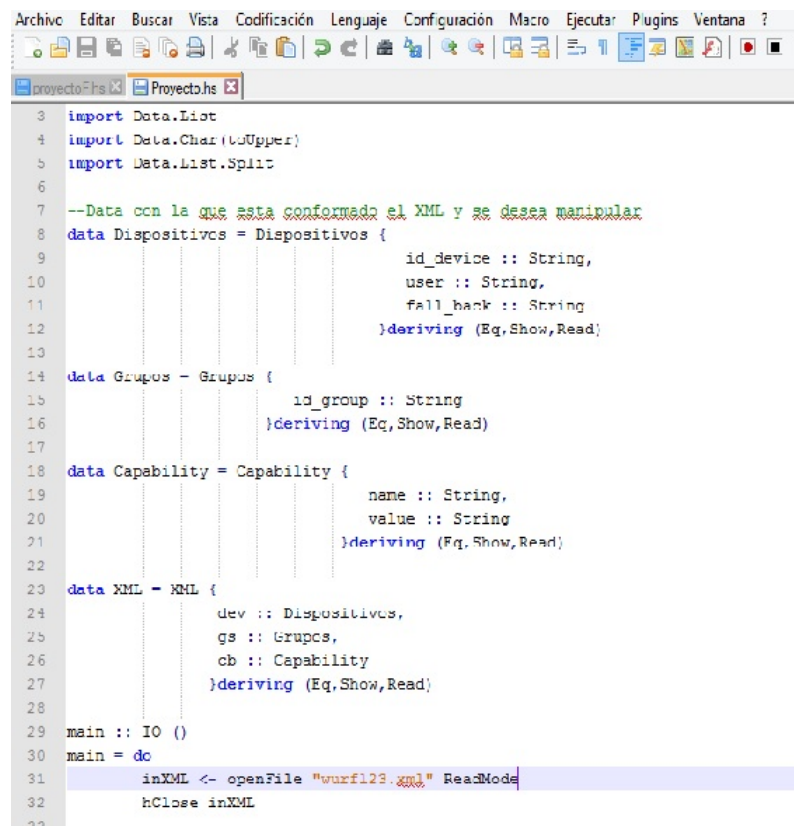
- name
- value

Capítulo 4

Implementación

El proyecto está compuesto inicialmente por la estructura principal del xml, Device, Group y Capability con sus correspondientes datos, ayudandonos por el mismo medio se encuentre Grupo que es un dato que esta conformado por los tres elementos principales.

Cada elemento, tiene su respectivo create, y a su vez tiene una función que permite listarlos de manera independiente. Sin embargo, lo dificultoso de esto fue, la interpretación de los datos como manejarlos y entender de manera mas abstracta la recursividad.



```
1  Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Macro  Ejecutar  Plugins  Ventana  ?
2
3  import Data.List
4  import Data.Char (toUpper)
5  import Data.List.Split
6
7  --Data con la que esta conformado el XML y se desea manipular
8  data Dispositivos = Dispositivos {
9      id_device :: String,
10     user :: String,
11     fall_back :: String
12 } deriving (Eq, Show, Read)
13
14 data Grupos = Grupos {
15     id_group :: String
16 } deriving (Eq, Show, Read)
17
18 data Capability = Capability {
19     name :: String,
20     value :: String
21 } deriving (Eq, Show, Read)
22
23 data XML = XML {
24     dev :: Dispositivos,
25     gs :: Grupos,
26     cb :: Capability
27 } deriving (Eq, Show, Read)
28
29 main :: IO ()
30 main = do
31     inXML <- openFile "wuxfl23.xml" ReadMode
32     hClose inXML
33
```

Figura 4.1: Ejemplo de la codificación.

Capítulo 5

Observaciones

La manipulación del nuevo lenguaje no fue para nada fácil de utilizarlo, se tuvo que investigar bastante y proporcionarse de varios ejemplos para poder entenderlo. Sin embargo, Haskell es un lenguaje que nos ayuda a manejar los datos de una manera mas sencilla sin la necesidad de las condiciones de recursividad, sino que uno mismo debe realizarlo independientemente. La herramienta ayuda al programador a tener una idea mas amplia de como estas funciones son implementadas y que existen diferentes maneras de realizar un proyecto en diferentes lenguajes de programación.

Capítulo 6

Conclusiones

Haskell de cierta forma nos ayudó en:

- Comprender como trabajan los lenguajes funcionales.
- Trabajar y entender la recursividad.
- Establecer metas claras de cuales son las funciones principales.
- Mejor entendimiento del manejo del parsing.