Bombur: Aplicación Web

Lenguajes de Programación

Tarea Programada 2

Profesor:

Ing. Andrei Fuentes L.

Alumnos:

Josué Espinoza C.

Mauricio Gamboa C.

Andrés Pacheco Q.

Mayo, 2014

El siguiente escrito pretende documentar la segunda tarea programada realizada en Prolog y Python.

Contenidos

Resumen Ejecutivo

Nombre de la aplicación: Bombur

Creadores de la aplicación: Josué Espinoza Castro, Mauricio Gamboa Cubero, Andrés Pacheco Quesada.

Objetivo principal de la aplicación: Brindar al usuario una opción cómoda de toma de decisiones respecto al lugar de donde quiere obtener su comida, dependiendo de sus gustos respecto a tipo, ubicación, platillos favoritos en el momento de escoger; eliminando molestias al poder decidir desde su teléfono inteligente.

Herramientas utilizadas:

- Prolog: Lenguaje de programación lógica, de paradigma declarativo. Es utilizado en esta aplicación para guardar en su base de conocimientos los restaurantes y platillos que el usuario así lo desee desde la aplicación.
- Python: Consiste en el lenguaje multiparadigma que se utilizó para manejar las consultas e inserciones de Prolog. Además, puede decirse que actúa a manera de “puente” entre la aplicación como tal y la base de conocimientos de Prolog, ya que la aplicación se comunica con Python para acceder a la información de Prolog.
- PySWIP: Medio utilizado para conectar los dos lenguajes mencionados anteriormente. Esta es la herramienta que permite a Python realizar consultas a Prolog, así como inserciones de nuevos hechos y reglas a la base de conocimientos. Se requiere de Python 2.3 o superior.
- Flask: Consiste en un “framework” escrito en Python con una semejante amigabilidad con el usuario que este lenguaje, cuya función es desarrollar aplicaciones web, razón por la cual fue incluido en este proyecto.
- Web responsive: Consiste en un diseño web que provee una experiencia de visualización óptima, ajustando la aplicación web que se esté realizando a casi cualquier pantalla, logrando que se pueda ver perfectamente ajustado, con una visión de los íconos óptima y proporcional al tamaño de la misma.

Funcionalidades: La aplicación web permite realizar búsqueda de restaurantes por nombre y tipo; las comidas ingresadas pueden ser consultadas por restaurante, país e ingrediente incluido en platillos de restaurante. Tambíen se pueden ingresar nuevos platillos y restaurantes, con la información que se requiere para las búsquedas mencionadas anteriormente.

Razón de ser de la aplicación

Dada la necesidad que existe actualmente de mejorar o simplificar la calidad de vida de las personas, se ha desarrollado más y mejor tecnología que contribuye a que esto sea una realidad. En el caso de “Bombur”, se ha canalizado esa necesidad específicamente en el crecimiento del uso de dispositivos móviles, para los cuales se ha desarrollado una innumerable cantidad de aplicaciones con distintas funcionalidades.

Esta aplicación de tipo web, se ha especializado en la necesidad básica de las personas de buscar alimentos, más específicamente, la búsqueda de lugares que ofrezcan este alimento de la manera más cómoda que al usuario le parezca. Muchas veces, al estar fuera del hogar, puede parecer tediosa la búsqueda de un restaurante que satisfaga las necesidades obtenidas en el momento. Incluso, el traslado de un restaurante a otro en el proceso de decisión puede provocar una indisposición en el cliente, ya que puede no poseer la información de los restaurantes para decidir si ir, o no.

Está diseñada para acabar con ciertas molestias, ya que permite guardar características de cada restaurante, que puede hacer tomar una decisión más acertada y con mayor información al usuario. Lo anterior hace referencia a las distintas funcionalidades que tiene el programa, ya que permite al usuario guardar sus restaurantes y platillos favoritos. A su vez, la aplicación permite realizar búsquedas de los restaurantes que se ha ingresado por: nombre, tipo de comida, país, entre otros; además de que cada restaurante tiene dentro de sus características: ubicación, platillos favoritos del usuario, horario…, por lo que resuelve de una manera eficiente la problemática que gira en torno a escoger un restaurante ajeno al hogar.

Funcionalidades

Esta aplicación web permitiría al usuario acceder a la información utilizando un smartphone o cualquier dispositivo con acceso a internet, en este caso utilizando la misma red. El sistema tiene dos funcionalidades: mantenimiento de datos y consultas. La funcionalidad de mantenimiento de datos permite:

- Ingresar nuevos restaurantes
- Agregar nuevos platillos

Por otra parte, la funcionalidad de consulta, permite obtener la siguiente información:

- Lista de restaurantes
- Lista de restaurantes filtrados por tipo de comida
- Búsqueda de restaurantes por nombre
- Lista de restaurantes que tienen platillos de algún país específico
- Lista de platillos de un restaurante específico
- Lista de platillos de un restaurante específico que tengan un ingrediente en particular

El sistema a su vez está estructurado en dos componentes: Front-end y Back-end. El Front-end será el componente encargado de manejar toda la interacción con el usuario, incluyendo entrada y salida de datos. Adicionalmente, se comunicará con el Back-end para las tareas relacionadas con mantenimiento de datos y consulta. El front-end será una aplicación web escrita en Python. Ya que la aplicación puede ser accesada desde un dispositivo móvil, una funcionalidad adicional es el diseño responsive.

El Back-end tendrá dos funcionalidades principales: manejar la base de conocimientos y responder las consultas. La base de conocimientos se conforma por una serie de declaraciones en Prolog que define todos los atributos de los restaurantes y los platillos. El Back-end deberá actualizar la base de

conocimientos usando los datos que el usuario ingrese al sistema por medio del Front-end.

Modos de funcionamiento:

Consultas

1. Bombur abre la aplicación web desde su teléfono móvil, y hace una consulta al sistema (a través del front-end, que es la parte del sistema a la que tiene acceso)

2. El front-end pasa la consulta al backend.

3. El Backend hace una búsqueda en la base de conocimientos, y devuelve la información al front-end.

4. El front-end muestra la información al usuario.

Mantenimiento de Datos

1. Bombur abre la aplicación web desde su teléfono móvil, e ingresa los datos de un restaurante, junto con sus platillos.

2. El front-end pasa los datos al backend

3. El Backend guarda los datos en la base de conocimientos, y le devuelve el resultado de la operación al front-end

4. El front-end muestra al usuario un mensaje que le indique que ya se terminó de realizar el almacenamiento de datos en la base de conocimientos.

Finalmente, el Back-end está escrito en el lenguaje de programación Prolog y el Front-end está escrito en Python.

Requerimientos del sistema

El proyecto fue hecho y probado únicamente en la distribución de Linux Ubuntu. El programa NO funciona para accederlo mediante un teléfono móvil al ejecutar el servidor del programa en una máquina virtual.

Se recomiendan las siguientes características del sistema:

- Procesador con arquitectura x86 (32 bits)
- Memoria RAM de mínimo 1GB
- Sistema Operativo Ubuntu (12.04 o una versión superior) o alguna otra distribución de Linux
- Google Chrome como explorador de internet por defecto

Descripción detallada del proyecto

A continuación se presenta una descripción detallada de las decisiones de diseño, los lenguajes utilizados, las tecnologías usadas y las librerías importadas. Además se muestran las diferentes funciones creadas con el fin de mostrar la funcionalidad, y los argumentos que recibe cada una, así como las decisiones tomadas para su implementación de tal forma que nos pudieran beneficiar en la creación de la aplicación.

Prolog

Consiste en un lenguaje de programación del paradigma lógico (especificación del declarativo), generalmente asociado con la inteligencia artificial. Es un lenguaje de tipo declarativo, que significa que recibe el planteamiento de problemas y, basado en eso, da una resolución por medio de reglas y su base conocimientos, donde almacena lo necesario para poder resolver el conflicto planteado.

En el caso de este proyecto, se utilizó Prolog para poder ingresar a la base de conocimientos restaurantes y comidas con sus respectivas características, para ser utilizadas y manejadas posteriormente por otro lenguaje, Python.

Python

Es un lenguaje de programación multipropósito, de alto nivel. Posee gran amigabilidad con el usuario, ya que con pocas líneas de código se pueden expresar conceptos que en otros lenguajes resulta más complejo. Es un lenguaje multiparadigma, ya que contiene el orientado a objetos, imperativo y funcional.

Para esta aplicación, se utilizó Python para acceder a datos de Prolog. Se manipuló Prolog desde Python, realizando consultas e inserciones desde éste. Las funciones utilizadas fueron: .query (consiste en consultar si existe en la base de conocimientos de Prolog, un predicado con el o los elementos ingresados), .assertz (consiste en agregar a la base de conocimientos de Prolog el predicado indicado en Python). Cabe destacar que las consultas y las inserciones desde Python se manejaron como “strings”; es decir, se concatenó el elemento que se quería buscar en un “string”, y Prolog lo recibía de esa manera.

Decisiones de Diseño

Se siguió las decisiones de diseño mencionadas a continuación:

- Se utilizó la integración de Prolog con Python mediante el módulo PySWIP.
- Se utilizó el framework de Python Flask para desarrollar la aplicación web.
- La programación del Back-End de la aplicación (todo lo que tiene que ver con la administración de la Base de Conocimientos y el trabajo de consultas e inserciones en Prolog) se encuentra en el archivo app.py. En él se encuentran todas las funciones necesarias para poner a funcionar el Back-End del programa correctamente.
- Se maneja la Base de Conocimientos en un archivo .txt que se carga cada vez que se corre el programa. Esta base de conocimientos se carga en Prolog mediante la función assertz de PySWIP. Cada vez que se inserta un restaurante o un platillo al programa, se insertan en la base de conocimientos estática del programa y se escriben en el .txt respectivo para que se carguen los hechos creados en otra ocasión.
- La apariencia de la aplicación web se obtuvo mediante un template descargado del internet.
- La pantalla de inicio de la aplicación web muestra dos botones que redirigen a las secciones de mantenimiento (ingresar datos) y de consulta. Además, en cada encabezado de la página se encuentra un menú de navegación a las páginas principales.
- Hay dos páginas donde hay formularios a llenar: al ingresar un restaurante (mantenimiento) y al agregar un platillo (platillo).
- La página de consulta contiene diversos botones que redirigen a la página de mostrar resultados (mostrarConsulta) o a la página que le pide al usuario el o los términos claves para realizar la consulta respectiva.

- Todos los resultados de las consultas se muestran en la página de mostrarConsulta en formato de lista y se indica si no se encontró resultados de ser así.
- Los botones en todas las páginas no cuentan con diseño responsive pero en cualquier dispositivo se pueden ver (al menos parte de ellos) sin tener que mover la pantalla.
- "Las funciones del Front-End se dividen en las funciones de redirección a otras páginas y las funciones de obtención de datos a través de formularios, que invocan a la función del Back-End de la consulta o inserción respectiva y redirigen a la página de resultados."

Interconexión Prolog-Python: PySWIP

PySWIP es un medio para conectar Python y SWI-Prolog el cual permite hacer consultas desde Python a Prolog, así como ingresar hechos y reglas a la base de conocimientos. Para su utilización se requiere de una versión de Python superior a la 2.3 . En el manual de usuario se explica su instalación.

Ejemplos de consultas e inserción de datos:

```
from pyswip import Prolog

prolog=Prolog()

prolog.assertz(“restaurante("+nombre.lower()+","+tipo.lower()”))

prolog.query("restaurante("+nombre.lower()
+",Tipo,Ubicacion,Telefono,Horario,PlatFav)")
```

Las primeras dos líneas permiten importar Prolog y hacer la conexión. La segunda línea específicamente permite establecer la base de conocimientos sobre la cual se hacen las consultas (prolog.query) y las inserciones (prolog.assertz) tal y como se explica en los otros dos ejemplos.

Funciones Creadas

A continuación se muestran el detalle de las funciones creadas para el Back-end. Las funciones del Front-End se dividen en las funciones de redirección a otras páginas y las funciones de obtención de datos a través de formularios, que invocan a la función del Back-End de la consulta o inserción respectiva y redirigen a la página de resultados.

Nombre de la función: grabar

Aridad: 1

Argumentos: hechos o reglas (str)

Funcionalidad: Esta función permite escribir el un archivo de texto (.txt). El objetivo principal es escribir los hechos y reglas en este archivo para después actualizar la base de conocimiento con las reglas y hechos almacenados en el archivo .txt .

Nombre de la función: leertxt

Aridad: 0

Argumentos: ninguno

Funcionalidad: Permite leer el archivo txt donde se almacenan los diferentes hechos y reglas. De esta forma, una vez que se cierre la aplicación la información de la base de conocimientos pueda ser agregada nuevamente a partir del archivo de texto.

Nombre de la función: alterar

Aridad: 1

Argumentos: lista simple

Funcionalidad: Esta función recorre listas simples con el objetivo de cambiar los guiones bajos y xyz (_ , xyz) colocados anteriormente con el fin de evitar problemas con Prolog al colocar comas y espacios. De esta forma, al reemplazar (_,xyz) por su forma normal (espacio y una coma) a la hora de imprimir las consultas, no se mostraría de forma incorrecta. Esta función utiliza otra llamada CambiarEspacios2, y esta a su vez utiliza la función replace.

Nombre de la función: nuevoRestaurante

Aridad: 6

Argumentos: nombre, tipo, ubicación, teléfono, horario y platillosFavoritos (str)

Funcionalidad: El objetivo de esta función es agregar a la base de conocimientos y al archivo de texto la información del nuevo restaurante. Todos los argumentos son strings. En esta función se llama a otra función llamada CambiarEspacios, la funcionalidad de la misma se explica posteriormente. Adicionalmente, esta función utiliza assertz (funcionalidad de Pyswip) con el objetivo de agregar hechos y reglas a la base de conocimientos. Finalmente, se llama a la función grabar, cuya funcionalidad se explico anteriormente.

Ejemplo:

>>> nuevoRestaurante("Burger King","Comida Rápida","Cartago Centro, el parque","25961452","L-V de 6 a 12 mn ","Whopper Junior, Galleta de chocolate, Hamburguesa Tica")

Se almacena en la base de conocimientos:

>>>restaurante(burger_king,comida_rapida,cartago_centroxyz_el_parque,25961452,l-v_de_6_a_12_mn_,whopper_juniorxyz_galleta_de_chocolatexyz_hamburguesa_tica)

Nombre de la función: CambiarEspacios y CambiarEspacios2

Aridad:1

Argumentos: palabra

Funcionalidad: Estas dos funciones permiten reemplazar dentro de un string el espacio por un guión bajo y la coma por xyz, CambiaEspacios2 en sentido contrario. Esto con el objetivo de no tener problemas al dejar espacios en Prolog y al colocar una coma donde Prolog lo puede interpretar como otro argumento, el otro objetivo es quitar el guion bajo y el xyz para mostrar el texto como debe de ser en el Front-end.

Nombre de la función: nuevaComida.

Aridad: 5.

Argumentos: restaurante, nombre, tipo, país, receta.

Funcionalidad: Consiste en almacenar una nueva comida en la base conocimientos de Prolog. Evidentemente, la nueva comida contendrá cada uno de los argumentos recibidos. Se utiliza la función CambiarEspacios( ) para cada uno de los argumentos. Esta función fue previamente descrita en este documento, se utiliza en la presente función con fines de validación de datos. Una vez realizado esto, se concantenan los datos en una variable llamada “pred”, que almacenará un string de la forma de un predicado de Prolog, que a su vez será insertado en la base de conocimientos del mismo. Lo anterior se realiza con la función llamada “assertz” que, desde Python, ingresa el predicado a la base de conocimientos de Prolog, almacenando así, una nueva comida.

Nombre de la función: consultaNombre.

Aridad: 1.

Argumento: nombre.

Funcionalidad: Mediante la función “query” de Prolog, se utiliza el argumento ingresado “nombre” para realizar la consulta sobre si existe uno o varios restaurantes de ese nombre. Si existe algún restaurante en la base de conocimientos, se retornará, en una lista, cada restaurante con sus datos.

Nombre de la función: consultaRestaurantes.

Aridad: 0.

Argumentos: -

Funcionalidad: Se realiza una consulta con la función de Prolog “query”, ingresando un predicado con todas las variables que posee de aridad la función restaurantes, retornando así todos los restaurantes existentes en la base de conocimientos.

Ejemplo de consulta:

```
prolog.query("restaurante(Nombre,Tipo,Ubicacion,Telefono,Horario,PlatFav)")
```

Nombre: consultaTipo.

Aridad: 1.

Argumento: tipo.

Funcionalidad: En este caso, la consulta se realiza de igual forma que en las demás, con la particularidad de ingresar el argumento “tipo” como predicado de la consulta. Con esto, la función retornará todos los restaurantes que brindan platillos del tipo consultado. Una vez obtenidas las consultas, se ingresa los nombres de los restaurantes en una lista (restaurantes.append("Restaurante: "+e["Nombre"])). Posteriormente se limpia la lista de los datos agregados para el manejo con Prolog con la función alterar(restaurantes).

Nombre: consultaPlatilloPais.

Aridad: 1.

Argumento: país.

Funcionalidad: La consulta se desarolla de la misma forma que se ha manejado el resto de funciones, en vez de la variable “Pais” de Prolog, se ingresa el argumento “pais” que se recibió en esta función. Se concatena con el resto del predicado y se ejecuta la consulta. Después de obtener la consulta, se ingresan los datos a una lista para posteriormente retornarla. Al igual, se pasa por la función alterar para eliminar (“_”,”xyz”).

Nombre: consultaPlatilloRest.

Aridad: 1.

Argumento: restaurante.

Funcionalidad: Consiste en retornar todos los platillos que se tienen registrados de un restaurante. Se ingresa el restaurante y, en una lista, se retornarán cada uno de los platillos con sus respectivas características. Al igual que en las

consultas anteriores, se realiza la consulta a Prolog con . query y los datos retornados se almacenan en una lista.

Nombre: RestDeIng.

Aridad: 2.

Argumentos: restaurante,ingrediente.

Funcionalidad: Retornará cada platillo del restaurante que se desea conocer que contenga en su receta el ingrediente descrito en el segundo argumento de la función. La consulta se realizará de la misma manera que las demás, y se retornará en una lista. La consulta de los ingredientes se realiza en Python comparando los ingredientes retornados de la consulta con el ingrediente ingresado por el usuario.

Problemas Encontrados

El mayor problema encontrado fue la incompatibilidad de pyswip con django (el framework con el que se comenzó a programar). Este problema retrasó el desarrollo de la aplicación en gran manera. Se solucionó corriendo el servidor de la aplicación web deshabilitando el DEBUG de django. De ahí, surgió el problema que al estar el debugger deshabilitado, no se cargaban los archivos estáticos (css, javascript e imágenes), lo que restringía el diseño responsive y el resultado final deseado.

Manual de Usuario

La siguiente sección compone un conjunto de pasos para que el programa funcione correctamente. Cabe destacar que las funciones del programa se reducirán a sólo poderse ejecutar en el computador servidor si se ejecuta en una máquina virtual. Considere los puntos dados en la sección de requerimientos.

Instalar pyswip

Ejecutar las siguientes líneas de código en la terminal:

1- wget http://www.swi-prolog.org/download/stable/src/pl-6.0.2.tar.gz
2- tar xzvf pl-6.0.2.tar.gz
3- cd pl-6.0.2/
4- ./configure --prefix=/usr --enable-shared
5- make && sudo make install
6- cd packages/clpqr/
7- ./configure --prefix=/usr
8- make && sudo make install
9- sudo ln -s /usr/lib/swipl-6.0.2/lib/i686-linux/libswipl.so /usr/lib/libpl.so
10- sudo ln -s /usr/lib/swipl-6.0.2/lib/i686-linux/libswipl.so.6.0.2 /usr/lib/.
11- wget http://pyswip.googlecode.com/files/pyswip-0.2.2.zip
12- unzip pyswip-0.2.2.zip
13- cd pyswip-0.2.2/
14- sudo python setup.py install

Instalar Flask

Ejecutar las siguientes líneas de código en la terminal:

1- sudo apt-get install python-virtualenv
2- ubicarse en la terminal en la carpeta del programa (vía comandos cd) por ejemplo:
cd TP2_Prolog
cd VersionFinal
3- ejecutar la siguiente línea:
. venv/bin/activate
4- ejecutar la siguiente línea:
sudo pip install Flask

Ejecutar el programa

1- Cambiar la última línea (318) del archivo app.py:
app.run(host='IP actual')
Ejemplo:

app.run(host='192.168.98.130')

2- Abrir la terminal, ubicarse en la carpeta donde está el programa:

Por ejemplo:

cd TP2_Prolog

cd VersionFinal

3- . venv/bin/activate

4- python app.py

5- En su smartphone o computador dijite la siguiente dirección en un navegador de internet:

La dirección IP escrita en el punto 1, dos puntos y el número (puerto de conexión) 5000

Por ejemplo:

192.168.98.130:5000

6- Disfrute de la aplicación