TECNOLÓGICO DE COSTA RICA
LICENCIATURA EN ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN

Sistema de consultas para recetas

Restaurante Le Puolet

Tarea Programada II

José Daniel Chacón Bogarín
John Largaespada
Evelyn Madriz Mata

16 de Octubre del 2012

Tabla de contenido

Descripción del problema

El siguiente trabajo, basado en el desarrollo de un sistema de consultas para un restaurante, consiste en la implementación de un mecanismo de comunicación entre una base de conocimientos dada en Prolog y un sistema de interacción con el usuario dado en Python.
Esto para el curso de Lenguajes de Programación del Tecnológico de Costa Rica.

Dicha aplicación consiste en dos módulos principales: Mantenimiento de datos y Consulta de datos. En el módulo de mantenimiento de datos, los usuarios podrán ingresar datos para todas las recetas del restaurante, estos datos se almacenarán mediante los siguientes campos: Lista de ingredientes, Lista de pasos, Nombre de receta, Autor de la receta y Estilo de la comida, por tanto la modalidad de consulta podrá hacer referencia a estos datos a través de ciertos atributos de las recetas.

La estructuración del programa consiste en Front-end y Back-end. El Front-end consiste básicamente en el lenguaje de programación elegido y la interacción del usuario, incluyendo la entrada y salida de datos y la intercomunicación con el Back-end, el Back-end por otro lado, contiene toda la base de conocimientos de recetas y que responde a todas las consultas pero de manera indirecta.

Los puntos a considerar en el desarrollo fueron los siguientes:

- Estructura de la base de conocimientos
- Funcionamiento de las consultas
- Actualización y mantenimiento de datos
- Librería externa responsable de la conexión entre Prolog y el lenguaje de programación
- Interfaz gráfica

Diseño del programa

Para el desarrollo de la tarea se tomaron decisiones respecto al diseño e implementación de la aplicación, entre estas decisiones se logró elegir el lenguaje de programación más adecuado y sencillo para programar. Según recomendaciones y consultas a internet se decidió el uso de Python como lenguaje complementario, esto debido a que es un lenguaje el cual se conoce y permite la conexión al lenguaje dado Prolog.

El desarrollo del código en Python se logró en el paradigma orientado a objetos ya que resultó ser más conveniente el uso de clases y manejo de objetos en el sistema de consultas, manipulación de datos y control del código de Prolog.

El diseño consta de dos grupos de clases principales: ManejoProlog, clase encargada de controlar el archivo de texto en donde se encuentra todo el código de Prolog y las clases de Interfaz gráfica que corresponden a VentanaPrincipal, VentanaMantenimiento, VentanaConsulta y VentanaResultado.

Toda la conexión entre Prolog y Python se maneja a través ManejoProlog que controla un archivo de Prolog llamado "Recetas.pl". En este archivo se encuentran todas las reglas y es donde se almacenan todos los hechos creados por el usuario. Cada vez que el usuario crea una regla, se generan las líneas de código pertinentes y se escriben el archivo de Prolog. Una vez terminado de escribir el código se vuelve a cargar la base de conocimientos para que esté actualizada con todos los nuevos datos. En el caso de borrado y actualización, ocurre el mismo resultado.

Las reglas de las recetas se definen con el siguiente formato:

- Receta(NombreReceta, AutorReceta, EstiloReceta).
- recetaIngrediente(NombreReceta, Ingrediente). <= Se pueden encontrar múltiples hechos de este tipo indicando que esa receta tiene varios ingredientes. Esto se intentó hacer en una lista en el hecho de Receta propiamente, pero tal implementación falló debido a que la regla de búsqueda presentó StackError y se tuvo que cambiar al siguiente formato, el cual complicó mucho las cosas, especialmente en la búsqueda.
- recetaPaso(NombreReceta, Paso). <= Al igual que los ingredientes pueden haber varios hechos de este tipo y también se intentó implementar como una lista, pero presentó el mismo problema.

Además de esto, se encuentran los siguientes predicados:

- buscarIngrediente(Nombre,Ingrediente):-receta(Nombre,Y,Z),recetaIngrediente(Nombre,Ingrediente). <= Este predicado se encarga de buscar todos los ingredientes de una receta determinada cuando se cumpla que el nombre de la receta dado se cumpla para alguna receta y el ingrediente se encuentre registrado con el mismo nombre de la receta.
- buscarPaso(Nombre,Paso):-receta(Nombre,Y,Z),recetaPaso(Nombre,Paso). <= Este predicado se encarga de buscar los pasos de una receta siguiendo los mismos pasos que el predicado anterior.
- buscarReceta(V,W,X,Y,Z) :- receta(V,W,X), buscarIngrediente(V,Y), buscarPaso(V,Z). <= Este predicado se encarga las recetas y todos los pasos e ingredientes que se le asocien llamando a los predicados anteriormente explicados.

Aquí el diseño presentó un problema ya que los resultados obtenidos de las consultas dan el producto cartesiano de todos sus ingredientes con sus pasos. Es decir, si se tiene:

receta(pizza, giovanni, italiana).

recetaIngrediente(pizza, queso).

recetaIngrediente(pizza, jamon).

recetaPaso(pizza, poner_ingredientes).

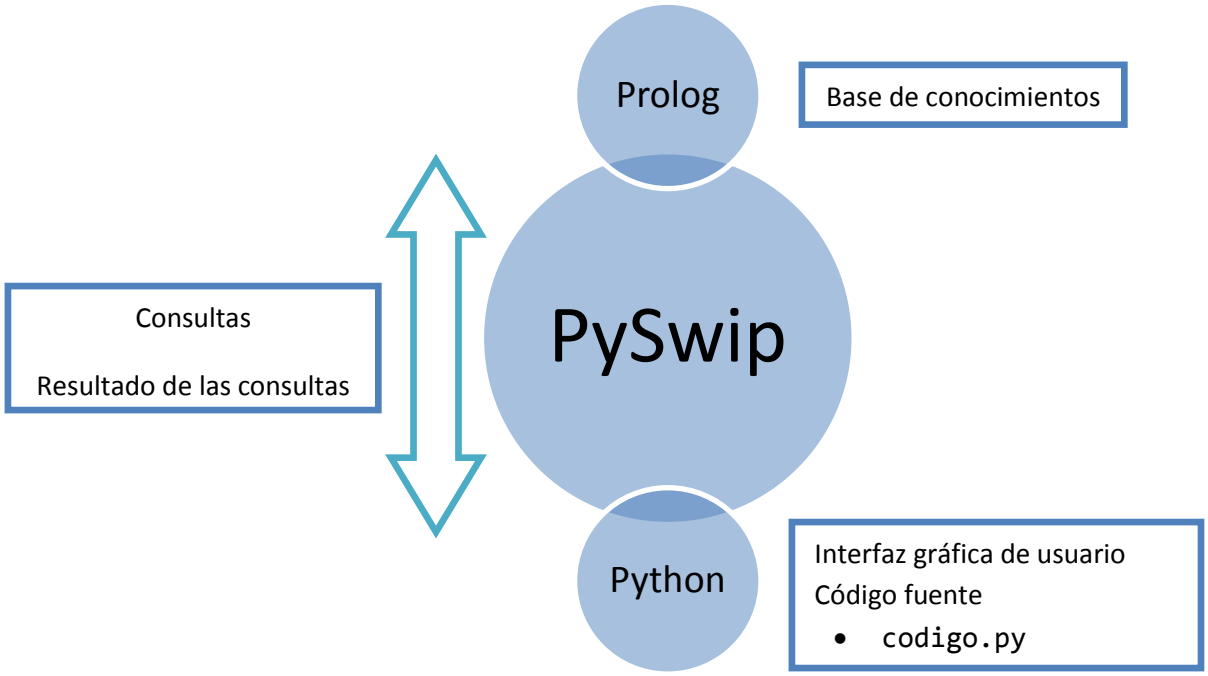
recetaPaso(pizza, cocinar).

Se obtendría de resultado:

- pizza, giovanni, italiana, queso, poner_ingredientes
- pizza, giovanni, italiana, queso, cocinar
- pizza, giovanni, italaiana, jamon, poner_ingredientes
- pizza, giovanni, italiana, jamon, cocinar

Por lo que hubo que concatenar los resultados y en consultas grandes se vuelve un poco complicado e ineficiente.

A continuación un enlace con referencia a los mismos: http://nullege.com/codes/search/pyswip



Librerías usadas

Para el desarrollo de la tarea se utilizaron las siguientes librerías:

1. `from Tkinter import *`
   Librería para la interfaz gráfica de usuario

2. `import ttk`
   Librería para el aspecto visual de widgets en la interfaz gráfica de usuario
3. `import tkMessageBox as box`
   Librería para el uso de mensajes de información al usuario
4. `from PIL import Image`
   Librería para el uso de imágenes en la interfaz gráfica de usuario
5. `from pyswip import Prolog`
   Puente que logra la conexión entre Python y Prolog

La elección de Pyswip como medio de interconexión entre Prolog y el lenguaje elegido Python, se da por múltiples búsquedas en la web, en las que se recomienda el uso del mismo.

A primera vista, pareciera que ambos lenguajes no comparten características en común, Python mantiene un lenguaje más natural en comparación a Prolog, el cual contiene más enunciados matemáticos; sin embargo comparten varias características lo que permite una conexión dada entre estos.

Análisis de resultados

Se cumplieron con las expectativas presentadas en la descripción del problema, en las que destacan dos módulos principales:

- Mantenimiento de datos:
  - Inserción completamente funcional.
  - Borrado completamente funcional.
  - Actualización presenta leves inconsistencias cuando se tratan de actualizar las listas. Si en la base de conocimientos se encuentran 3 ingredientes, y el usuario desea cambiar los ingredientes a dos, se actualiza ambos, pero el tercero anterior aún queda registrado. Situación similar con la actualización de los pasos. Por el otro lado, si se intentan actualizar 4 ingredientes, se actualizan tres y el cuarto se desecharía.
- Consulta de datos:
  - La consulta de datos funciona adecuadamente para cualquier campo y bajo cualquier combinación. Si no se encontraron resultados, se muestra un mensaje de error.

Sin embargo, por los problemas presentados con la consulta por listas y los cambios efectuados para solucionar tal situación, se presenta un fallo en el resultado de hacer un programa eficiente.

Manual de usuario

Guía de instalación

1. Iniciar sesión en Ubuntu.
2. Abrir el buscador web en la siguiente ruta:
   http://code.google.com/p/pyswip/downloads/detail?name=pyswip-0.2.2.zip

3. Descargar el archivo: **pyswip-0.2.2.zip**.
4. Posicionarse en el buscador web para la siguiente ruta: http://www.swi-prolog.org/download/stable?show=all
5. Dirigirse al apartado de **Sources** y descargar el archivo con el nombre **SWI-Prolog source for 6.0.2.**
6. Abrir la terminal de Ubuntu.
7. Colocarse en la ruta donde se han descargado los archivos.
8. Ejecutar los comandos siguientes para descomprimir, instalar y configurar Prolog:

    a. `tar xzvf pl-6.0.2.tar.gz`

    b. `cd pl-6.0.2/`

    c. `./configure --prefix=/usr --enable-shared`

    d. `make && sudo make install`

    e. `cd packages/clpqr/`

    f. `./configure --prefix=/usr`

    g. `make && sudo make install`

    h. `sudo ln -s /usr/lib/swipl-6.0.2/lib/i686-linux/libswipl.so /usr/lib/libpl.so`

    i. `sudo ln -s /usr/lib/swipl-6.0.2/lib/i686 linux/libswipl.so.6.0.2 /usr/lib/.`

9. Volver a colocarse en la ruta donde se descargaron los archivos.
10. Ejecutar los siguientes comandos para instalar Pyswip:

    a. `unzip pyswip-0.2.2.zip`

    b. `cd pyswip-0.2.2/`

    c. `sudo python setup.py install`

11. Ir al Centro de Software de Ubuntu e instalar Python 2.7
12. Ejecutar Python y abrir el archivo: **codigo.py**

A continuación la ejecución del programa con interfaz gráfica:

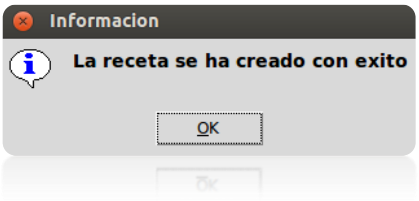
Guía de uso

Pantalla de inicio de la aplicación:

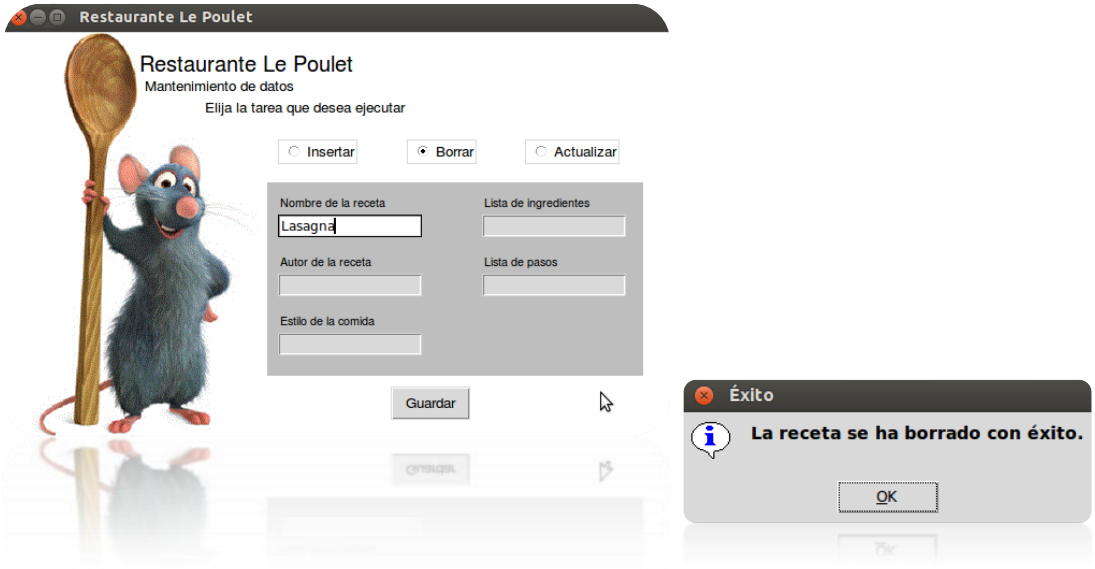
Vista panorámica del inicio de la aplicación

Al ingresar al Modo Mantenimiento:

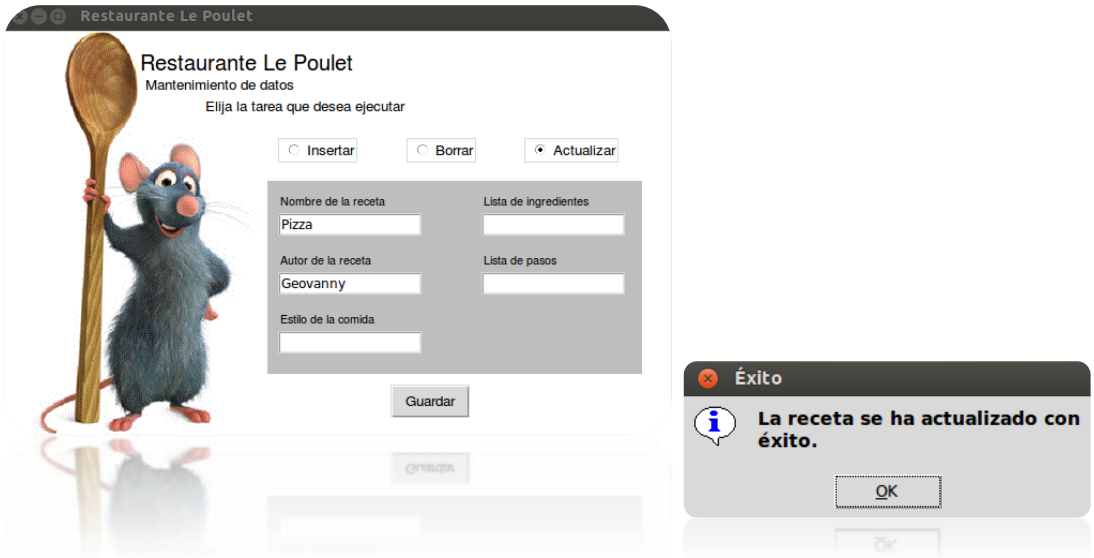
Podrá ingresar una receta a la base de conocimientos:



Podrá borrar una receta a la base de conocimientos:



Podrá actualizar una receta a la base de conocimientos:

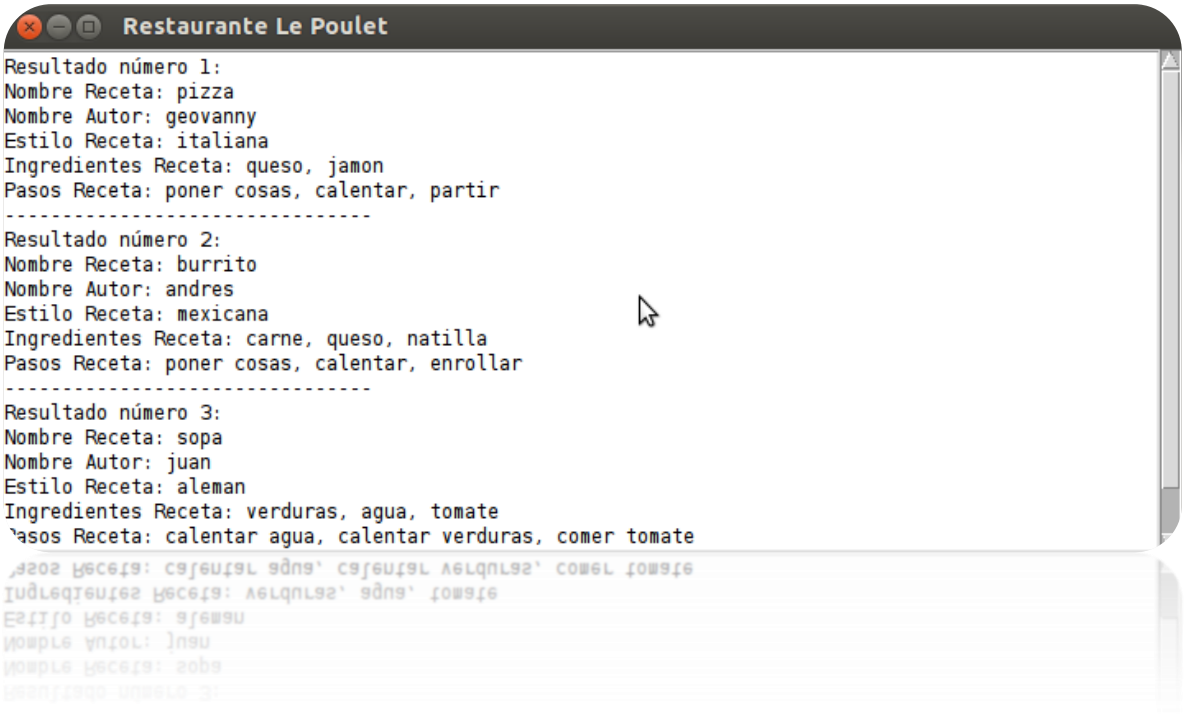


Al ingresar al Modo Consulta:

Podrá consultar por campos determinados:

O por todos los campos:

Para obtener los siguientes resultados:



Conclusión personal

Al desarrollar esta tarea se lograron los objetivos propuestos y se aprendió acerca de la conexión entre distintos lenguajes de programación, inclusive de distinto paradigma computacional.
El uso de hechos y reglas para la creación de la base de conocimientos resultó ser conveniente para el almacenamiento y consulta de datos según distintos criterios dados por el usuario en el lenguaje elegido como el más conveniente: Python.

Dentro de los aspectos más difíciles de la tarea, se incluye la instalación del PySwip en Ubuntu y el manejo de las listas en Prolog para la consulta de datos como: Lista de ingredientes y Lista de Pasos, el cual presentó bastantes problemas y por falta de un conocimiento amplio del lenguaje, se tuvo que tomar una solución ineficiente a la problemática presentado, lo cual a su vez trajo problemas de control de datos.

Bibliografía

Treasure-Jones , . (1996). *Introduction to Prolog - Rules* Recuperado de http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/rules.html

A simple learning program using pyswip « Pythonism (2012). Recuperado de http://pythonism.wordpress.com/2011/02/05/a-simple-learning-program-using-pyswip/

Lundh, F. (2009). *The Tkinter Listbox Widget* Recuperado de http://effbot.org/tkinterbook/listbox.htm

Lundh, F. (2009). *The Tkinter Listbox Widget* Recuperado de http://effbot.org/tkinterbook/listbox.htm

Lundh, F. (2009). *The Tkinter Text Widget* Recuperado de http://effbot.org/tkinterbook/text.htm

Lundh, F. (2009). *The Variable Classes (BooleanVar, DoubleVar, IntVar, StringVar)* Recuperado de http://effbot.org/tkinterbook/variable.htm

PySWIP: Facts and Rules (2009). Recuperado de http://yuce.tekol.net/2009/pyswip-facts-and-rules/

Python Software Foundation (2012). *5. Built-in Types ? Python v2.7.3 documentation* Recuperado de http://docs.python.org/library/stdtypes.html

Python Software Foundation (2012). *5. Data Structures ? Python v2.7.3 documentation* Recuperado de http://docs.python.org/tutorial/datastructures.html

Python Software Foundation (2012). *7. Input and Output ? Python v2.7.3 documentation* Recuperado de http://docs.python.org/tutorial/inputoutput.html