

Documentación Externa II Proyecto Programado

16 de Octubre

2012

Instituto Tecnológico de Costa Rica.

Escuela de Computación.

Profesor: Andréi Fuentes Leiva.

Alumnos:

Yader Morales López. Frank Brenes Alvarado. María Mercedes Escalante Karr.

TI – 3404

Lenguajes de
Programación

Índice

ÍNDICE	2
INTRODUCCIÓN	3
DESCRIPCIÓN DEL PROBLEMA	4
DISEÑO DEL PROGRAMA.....	5
LIBRERÍAS UTILIZADAS	7
ANÁLISIS DE RESULTADOS	8
MANUAL DEL USUARIO	9
PROCESO DE INSTALACIÓN DE PYSWIP.....	14
CONCLUSIÓN PERSONAL	15
BIBLIOGRAFÍA.....	16

Introducción

El paradigma lógico es la aplicación de los conocimientos y reglas de lógica para el diseño de programas, este paradigma puede ser tanto declarativo como funcional. El paradigma declarativo, corresponde al conjunto de afirmaciones que describen el problema (no se especifica la solución, solo lo que se desea buscar). Por otro lado el paradigma funcional es la especialización del paradigma declarativo, basado en la aplicación de funciones.

Dentro de las aplicaciones del paradigma lógico tenemos la inteligencia artificial, el desarrollo de sistemas expertos y la demostración automática de sistemas.

Dado a sus amplias aplicaciones dentro de diversos campos, en especial el informático, el desarrollo del segundo proyecto programado se llevó a cabo bajo el paradigma lógico, por medio del lenguaje de programación de Prolog (Programation et logique), el cual es un lenguaje semi – interpretado, el cual cuenta con características del paradigma lógico tales como el mecanismo de inferencia propio, razonamiento deductivo (también conocido como encadenamiento hacia atrás), además de proporcionar el rápido prototipaje de programas.

Algunas de las aplicaciones dentro del mercado de este lenguaje son el análisis de redes sociales, sistema de consejería de abogados, algunos proyectos de bases de datos como el proyecto del genoma humano, entre otras.

Descripción del Problema

El Chef Giovanni ha tenido recientemente muchos problemas con sus clientes en su restaurante Le Poulet, debido a la falta de organización con sus recetas y a la poca atención que ha prestado en la cocina ha cometido errores en las órdenes de sus comensales, por lo que desea un programa que le permita mantener todas las recetas de sus creaciones en orden, sin que se pierdan o se alteren.

Dados los requerimientos del chef, el objetivo principal de la segunda tarea programada es crear una base de conocimiento que almacene las recetas y les permita a los encargados del restaurante revisar las recetas y los pasos a seguir de cada platillo en específico. Además estas consultas se deben hacer ya sea por los ingredientes de la receta o los pasos a seguir, el programa debe permitir la búsqueda por varios ingredientes e incluso varios pasos a seguir.

De la misma manera se debe tomar en cuenta que cada vez que el Chef Giovanni o alguno de los encargados de cocina del restaurante inventen un nuevo platillo, es necesario agregarlo a la base de conocimientos, por lo que también debe existir la posibilidad dentro la aplicación de agregar nuevos platos.

Diseño del Programa

Para suplir las necesidades y requerimientos del programa deseado por el Chef Giovanni y su restaurante Le Poulet, se implementó el prototipo de una aplicación que fue desarrollada bajo las características de Python y Prolog, mediante la librería Pyswip, esta librería será objeto de un detalle más amplio más adelante.

¿Por qué usar pyswip?

Se encontraron una serie de programas que cumplían con los objetivos de la liga que se nos solicitaba, pero fueron probados y no funcionaron correctamente, ejemplo swipy. Pero al final se decidió pyswip por la facilidad de instalación y que se encontraba una serie de información valiosa en la web sobre el programa, además de ejemplos que fueron de ayuda para el desarrollo del sistema.

El programa desarrollado para el restaurante Le Poulet está conformado de la siguiente manera:

- **Ventana Principal**
- **Ventana para ingresar recetas:** En esta ventana se presentan los campos necesarios para ingresar las recetas, una vez que el usuario ingresa todos lo necesario y si cumple con todo, además de presionar el botón para ingresar la receta, se agrega la receta a la base de conocimientos, por medio de la función `ingresar_receta`, la cual valida que los espacios de los entry's no sea vacíos, además esta función retorna la función `guardar_cont` la cual es la responsable de insertar las recetas dentro del documento de texto (txt) que corresponde a la base de conocimientos del programa. Esta función cuenta con dos variables Functor (receta y assertz). De los cuales receta funciona tanto para agregar como para consultar datos en la base de conocimientos.
- **Ventana de consultas:** Recibe la información del usuario para la consulta, una vez que el usuario ingresa la información que desea consultar y oprime el botón consultar, este llama a la función `consultar_receta` la que unifica los valores del usuario con cinco variables distintas para poder realizar la búsqueda dentro del txt que conforma la Base de Conocimiento. Esta función utiliza banderas (variable booleana) para contemplar los campos a buscar, por lo que luego mediante el comando de pyswip Query realiza la consulta con las variables que el usuario

desea. Luego se buscan si existen más recetas que cumplan con los requerimientos del encargado o del chef.

Si no se encuentran resultados, la función muestra un mensaje en el que especifica que no se encontraron coincidencias, de lo contrario la función llama a la ventana que muestra los resultados al usuario.

- **Ventana que muestra los resultados sobre las consultas de la Base de Conocimientos:** Esta ventana recibe como parámetros las cinco variables por las que se compone una receta, estas variables son unificadas dentro de la función `consultar_receta`. De igual manera dentro de esta ventana se verifica mediante un ciclo `while` que se verifica que existan soluciones en la base de conocimientos y con el uso de banderas que la unificación de la consulta. De manera que si la bandera es `true` quiere decir que este argumento se uso para consultar.

Librerías Utilizadas

Tkinter

Tkinter es la librería gráfica incluida en la versión oficial del intérprete Python, la cual nos permite realizar interfaces gráficas de forma muy sencilla. Es robusta y madura, el nombre Tkinter proviene de “TK Interface”. Dentro de la tarea programada la encontramos presente en los llamados a las ventanas tales como la ventana principal, la de ingresar las respectivas recetas entre otros. Además la creación de botones, el menú, entrys y labels fueron desarrollados con esta librería.

pyswip

PySWIP es un puente entre los lenguajes de programación Python y SWI-Prolog, esta librería permite consultar en Prolog, utilizando para esto SWI-Prolog en programas de Python. Incluye tanto una interfaz SWI-Prolog y una clase de utilidad que hace que sea fácil de consultar con SWI-Python. Dado que se usa SWI-Prolog como una biblioteca compartida y ctypes para acceder a ella, que no requiere compilación para ser instalado. Dentro de las funciones que se utilizaron para la tarea programada están las siguientes:

- **Functor:** Se utiliza para formar las clausulas por las cuales deberá cargar la base de conocimientos, tanto para agregar como para realizar consultas.
- **Variable:** Es utilizada para realizar las consultas y obtener el valor que el usuario desea consultar.
- **Query:** Este es el comando con el cual se realizan las consultas en Prolog.
- **Call:** Se llama a Prolog para realizar las consultas y a su vez para obtener los valores de las variables.

tkMessageBox

Librería encargada para mostrar cuadros de mensajes en pantalla. Dentro del programa se muestra como los mensajes en la ventana de consultar receta cuando no se encuentran coincidencias. Además en algunos casos para mostrar warnings sobre campos incompletos, como por ejemplo en la ventana de ingreso de recetas.

Análisis de Resultados

Dentro de los logros que se completaron en la tarea programada tenemos:

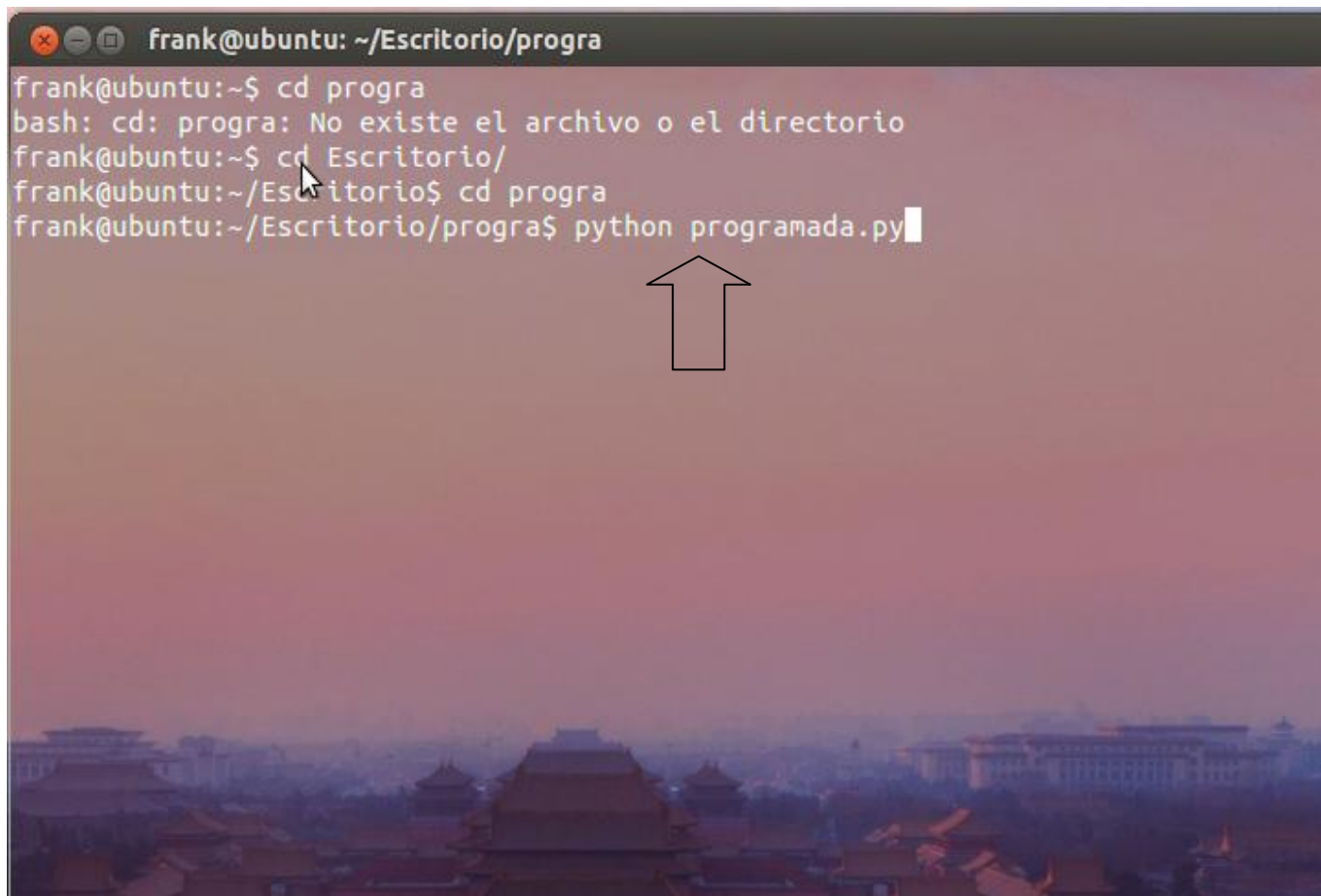
- El programa es capaz de crear una base de conocimiento con los datos ingresados por el usuario.
- La aplicación realiza las consultas de las recetas que se encuentran dentro de la base de conocimientos.
- El programa puede insertar nuevas recetas de acuerdo a los requerimientos del chef.
- La aplicación cuenta con interfaz gráfica.
- La base de conocimientos que contiene las recetas está conformada por un archivo de texto (txt). El cual no se carga con el programa, sin embargo sirve como respaldo.
- El programa acepta los datos del usuario en cualquier formato, sin importar letras mayúsculas o minúsculas.
- Al cerrar los programas no se tienen problemas con los ciclos.
- Se realizó exitosamente la conexión entre Python (front end) y Prolog (Back end).
- Cada vez que se cierra la aplicación se pierde la información de la base de conocimientos.

Dentro de los puntos que no se pudieron completar tenemos:

- No se puede realizar la búsqueda por pasos a seguir de una receta o por múltiples ingredientes.

Manual del usuario

Para ejecutar el segundo proyecto programado, lo primero que se debe hacer es ingresar a la dirección donde se ubica el archivo .py, una vez que se ingresa a la carpeta llamar al archivo primero indicando el programa con el que se desea abrir.

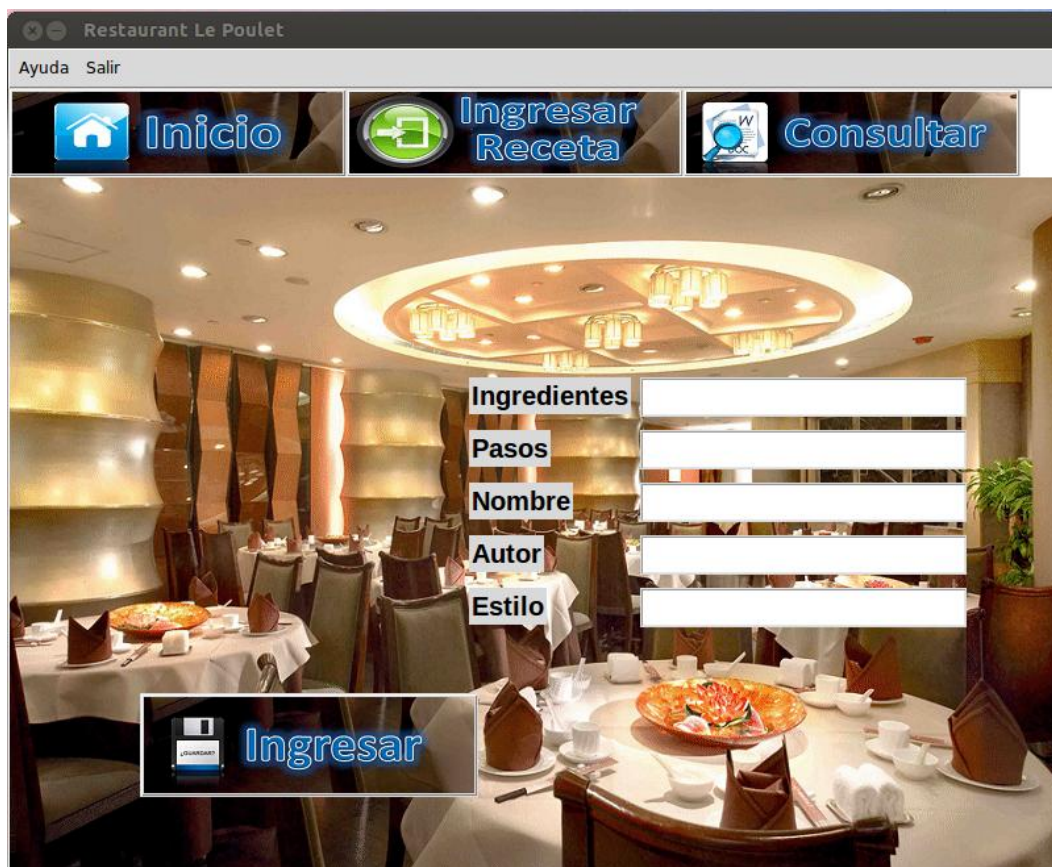
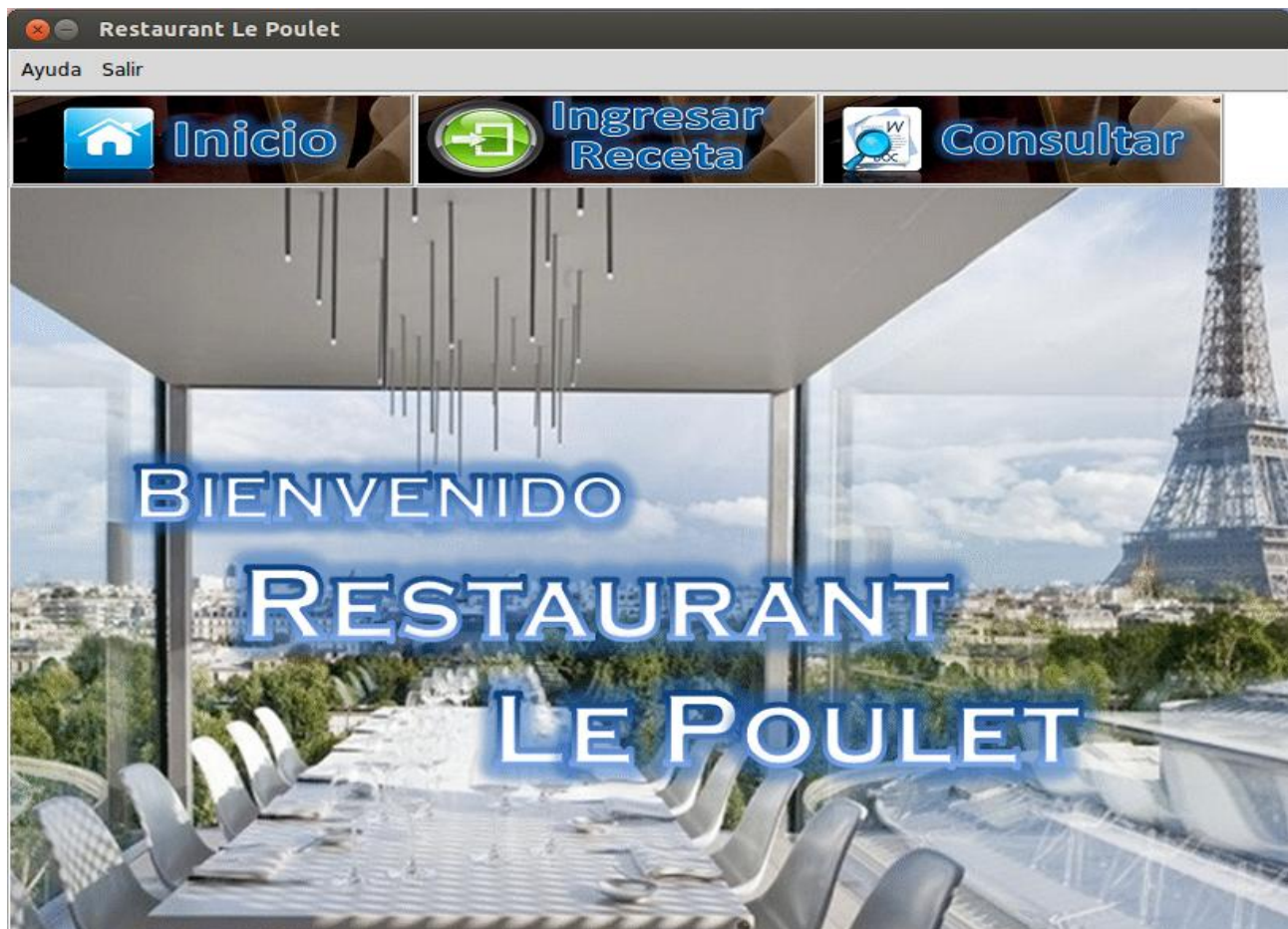
A terminal window titled 'frank@ubuntu: ~/Escritorio/progra' displays the following commands and output:

```
frank@ubuntu:~$ cd progra
bash: cd: progra: No existe el archivo o el directorio
frank@ubuntu:~$ cd Escritorio/
frank@ubuntu:~/Escritorio$ cd progra
frank@ubuntu:~/Escritorio/progra$ python programada.py
```

A mouse cursor is positioned over the 'Escritorio/' prompt. A large white arrow points upwards from the bottom center of the terminal window towards the 'python programada.py' command. The background of the terminal window features a faint, artistic image of traditional Chinese architecture.

En la imagen anterior se muestra un ejemplo de cómo llamar al archivo mediante la consola.

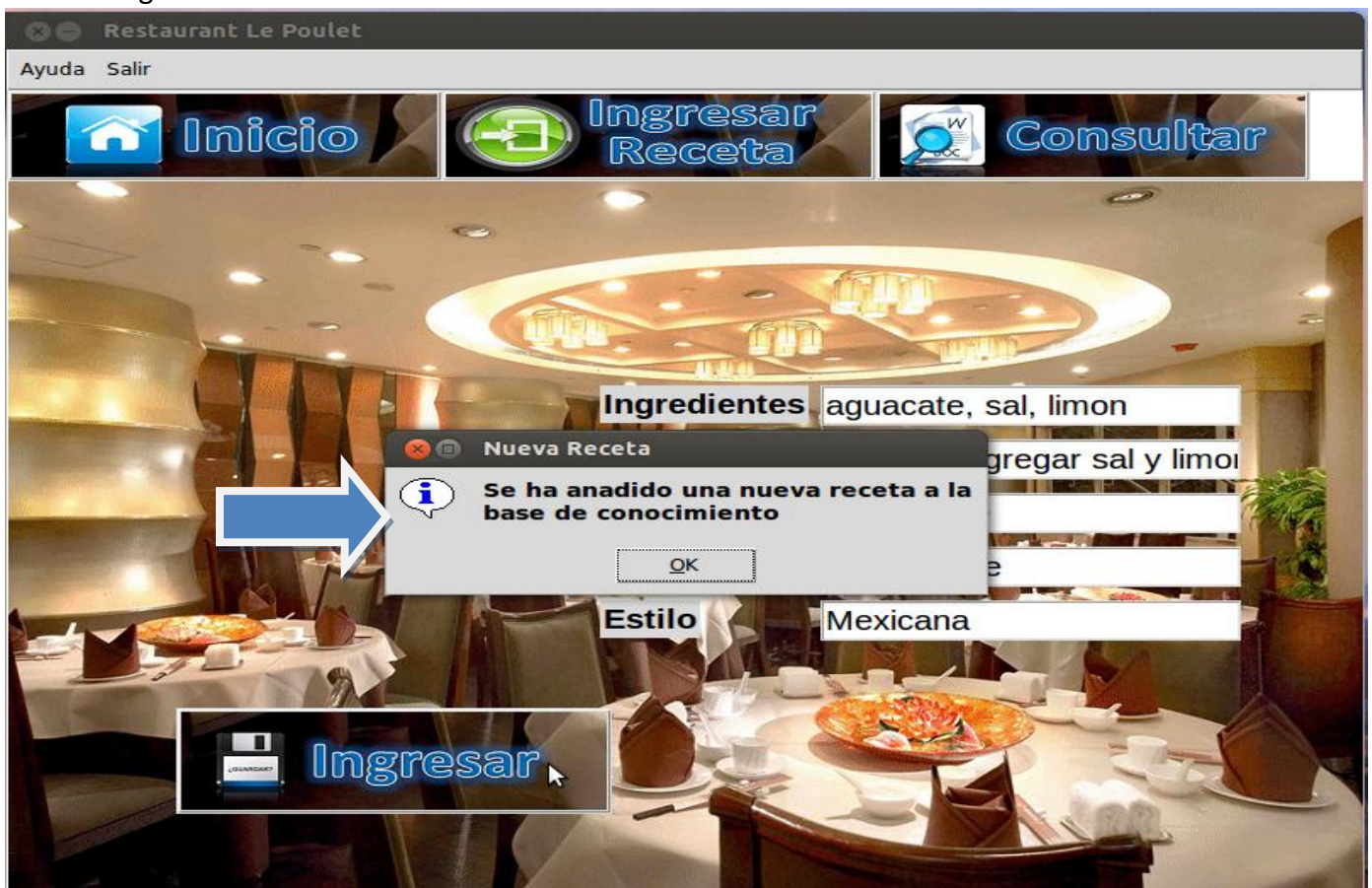
Al ejecutar el comando, la interfaz gráfica de la tarea programada se desplegará, obteniendo como resultado la pantalla principal de la tarea programada.



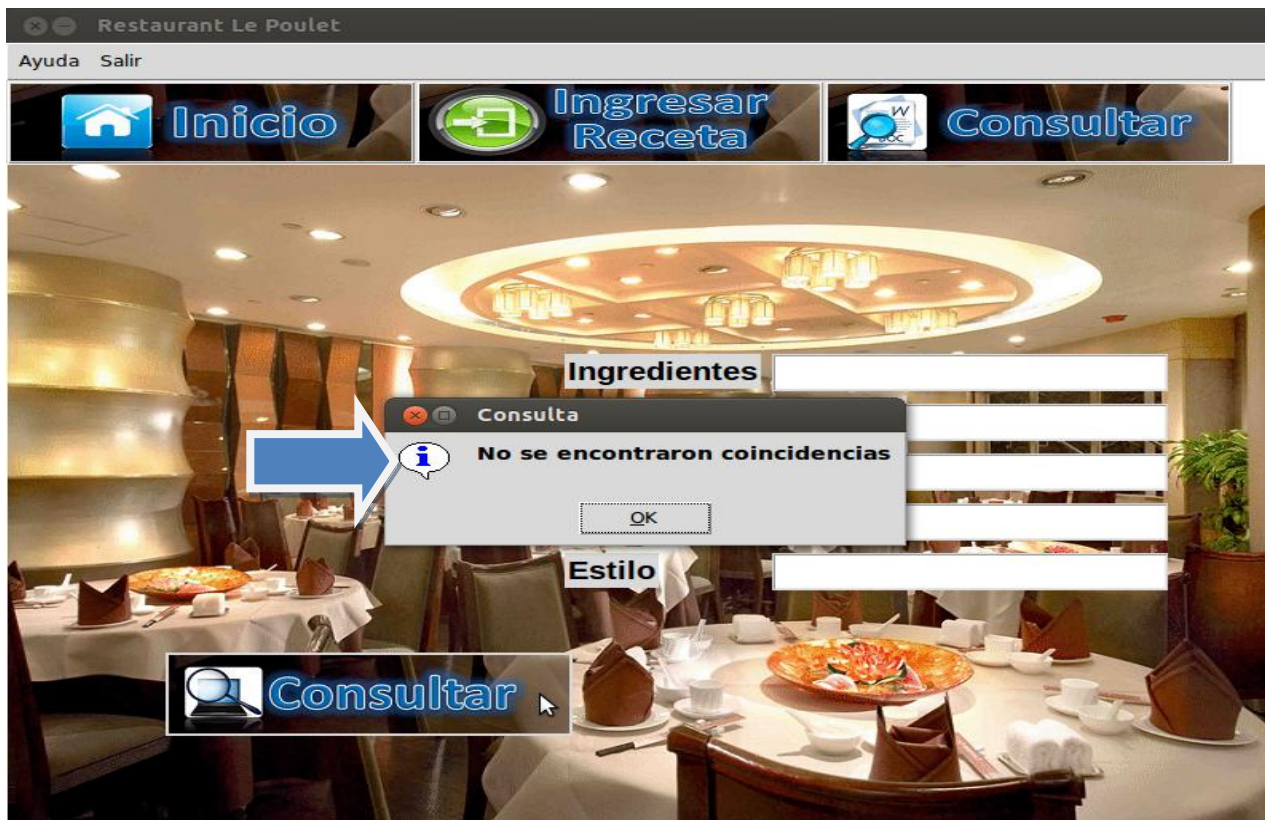
En la pestaña de ingresar receta contamos con los espacios necesarios para poder almacenar los datos de una nueva receta. Como se muestra en la imagen al lado izquierdo, estos son los pasos necesarios para ingresar una receta.



Una vez que el chef o algún encargado del restaurante, ingrese la información de la receta, debe presionar el botón de ingresar el cual proporcionará que se guarden los datos dentro de la base de conocimientos, enseñando un mensaje de confirmación al usuario de la siguiente manera.



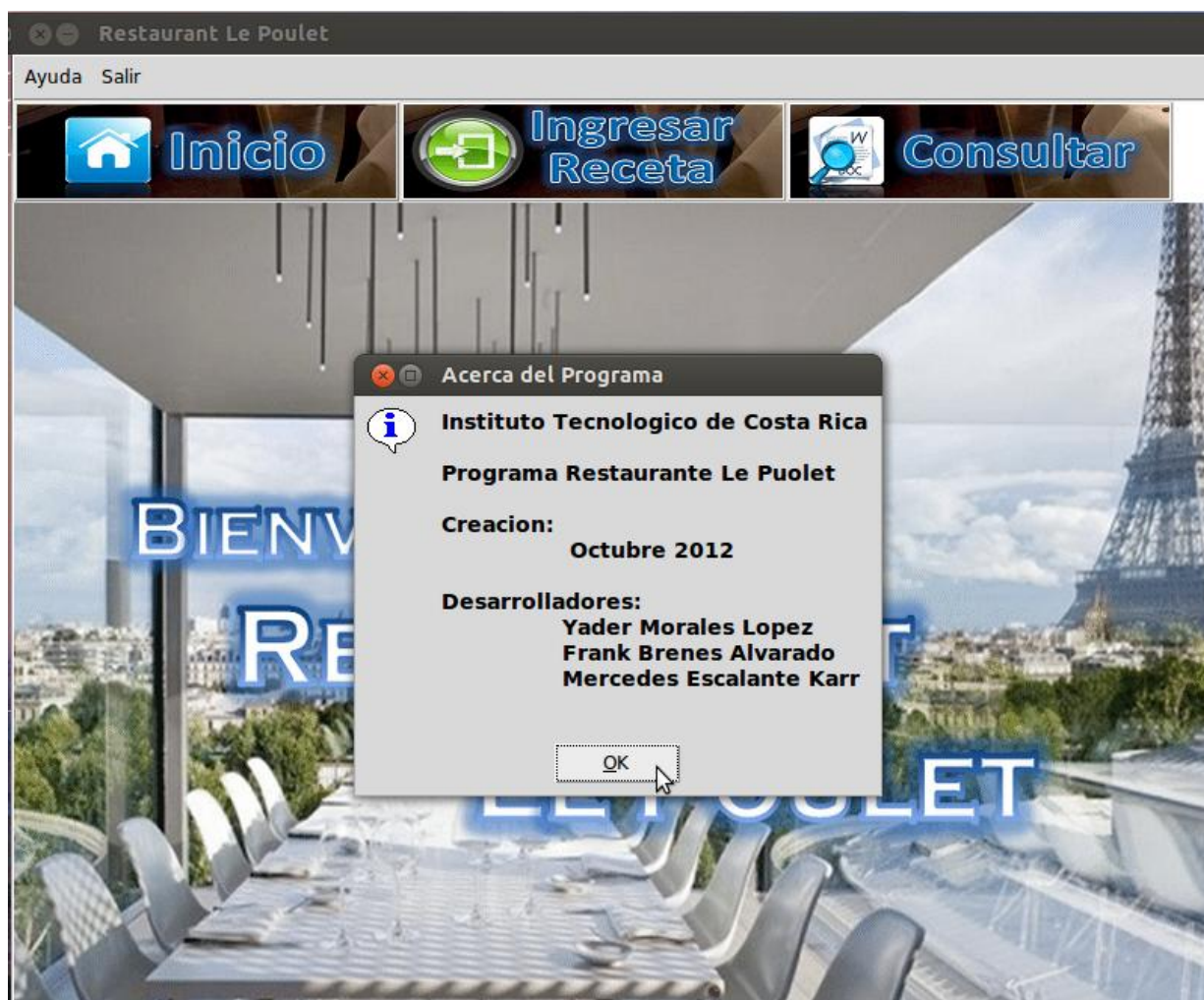
También contamos con la pantalla consultar, por lo que se encontraran los espacios para ingresar los datos que se desean consultar, en caso de que el usuario deje todo en blanco o no se encuentren recetas dentro de la base de datos con lo que está buscando se desplegará un mensaje indicándole que no se encontraron coincidencias.



En caso contrario se mostrará las consultas realizadas.



Por último tenemos la ventana que nos muestra la versión de la aplicación, el nombre de los desarrolladores y la institución donde fue desarrollada.



Proceso de Instalación de Pyswip

Para la instalación de la librería Pyswip, se siguieron los siguientes pasos

1. **Obtener la fuente desde la siguiente dirección <http://www.swi-prolog.org/download/stable>:**

```
$ wget http://www.swi-prolog.org/download/stable/src/pl-6.0.2.tar.gz
```

2. **Extraer los archives y acceder a la carpeta:**

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

```
$ cd pl-6.0.2.tar.gz
```

3. **Configurar la fuente con la librería compartida:**

```
$ ./configure --enable-shared
```

***Nota: Si se usa un sistema de 64-bit, es preferable compilarlo con la bandera -ggdb en orden para tener PySWIP y para que este trabaje correctamente con SWI-Prolog.

```
$ CFLAGS=-ggdb ./configure --enable-shared
```

4. **Compilar la fuente:**

```
$ make
```

5. **Instalar la fuente:**

```
$ sudo make install
```

6. **Instalar Python con una version superior a la 2.5.**

7. **Desempaquetar PySwIP e instalarlo con ``python setup.py install``.**

Conclusión Personal

Luego de desarrollar el segundo proyecto programado dentro de un paradigma totalmente distinto al paradigma imperativo, podemos rescatar la importancia de comprender las distintas aplicaciones dentro de los lenguajes de programación que se encuentran bajo el paradigma lógico.

De igual manera se rescatan las diferencias entre los paradigmas estudiados (imperativo y lógico) y las características dentro de las aplicaciones de la industria para ambos.

Durante el desarrollo de la tarea programada fomentamos habilidades claves para ser programadores exitosos, tales como la investigación, la cual fomentamos con el uso de la librería Pyswip, que fue una pieza clave para terminar el proyecto programado en el tiempo plazo.

Es importante recalcar, que con el desarrollo de esta tarea no solo aprendimos sobre distintos puntos de vista para resolver un mismo problema; si no también reforzamos el trabajo en equipo, el desarrollo de algoritmos en conjunto, conocimientos de cada uno de los integrantes y lo más importante nos enseñó a motivarnos entre nosotros para lograr concluir una tarea en común.

Bibliografía

Pyswip (s. f.). Recuperado el 1 de Octubre del 2012, de 5:42pm <http://nullege.com/codes/search/pyswip.Variable>

pyswip - PySWIP is a bridge between Python and SWI-Prolog. - Google Project Hosting (s. f.). Recuperado el 15 de Octubre del 2012, de <http://code.google.com/p/pyswip/>

pyswip.Atom - Nullege Python Samples (s. f.). Recuperado el 8 de Octubre del 2012, de <http://nullege.com/codes/search/pyswip.Atom>

pyswip.Functor - Nullege Python Samples (s. f.). Recuperado el 8 de Octubre del 2012, de <http://nullege.com/codes/search/pyswip.Functor>

Duda con ventanas en Tkinter en python (s. f.). Recuperado el 2 de Octubre del 2012, de http://foro.elhacker.net/scripting/duda_con_ventanas_en_tkinter_en_python-t282600.0.html

Examples - pyswip - PySWIP Examples - PySWIP is a bridge between Python and SWI-Prolog. - Google Project Hosting (s. f.). Recuperado el 8 de Octubre del 2012, de <http://code.google.com/p/pyswip/wiki/Examples>

Pyswip 0.2.0 : Python Package Index (s. f.). Recuperado el 4 de Octubre del 2012, de <http://pypi.python.org/pypi/pyswip/0.2.0>

Python and %s - Stack Overflow (s. f.). Recuperado el 9 de Octubre del 2012, de <http://stackoverflow.com/questions/997797/python-and-s>

Tkinter Scrollbar Patterns (s. f.). Recuperado el 8 de Octubre del 2012, de <http://effbot.org/zone/tkinter-scrollbar-patterns.htm>

Python wrapper for SWI Prolog(s. f.). Recuperado el 7 de Octubre del 2012, de <http://code.google.com/p/swipy/>

Pérez, B. (2012). TuPera (Versión Alnitak) [Software]. San José, Costa Rica.