Tecnológico de Costa Rica

ATI

“Hungry?”

Tarea Programada II

Luigui Madrigal

Andrés Fernández

José María Rojas

06/05/2014

Contenido

[1.Resumen Ejecutivo 3](#__RefHeading__361_1594199210)

[2.Propósito (descripción del app, requerimientos) 3](#__RefHeading__363_1594199210)

[4.Descripción detallada (decisiones de diseño, lenguajes de programación usados, tecnologías y librerías usadas) 4](#__RefHeading__365_1594199210)

[5.Problemas encontrados 5](#__RefHeading__367_1594199210)

[6.Manual de usuario 6](#__RefHeading__369_1594199210)

# Resumen Ejecutivo

El proyecto consiste sobre una aplicación móvil similar al conocido Foursquare, pero con un enfoque más a comidas y restaurantes. Se plantea al inicio el problema sobre un usuario que le importa y le presta atención sobre lo que come y busca ciertas exigencias sobre la elección de su próximo lugar de comida. Entre sus exigencias por ejemplo el amigo glotón podría encontrar un restaurante cercano que ofrezca un platillo con un ingrediente de su preferencia. Para realizar dichas consultas se debe contar con una aplicación móvil. Ya que al ser este usuario una persona con muchas exigencias y además en constante trasladación, le sería más sencillo poder desde su Smartphone, realizar las consultas que desee para poder elegir en su lugar de comida.

Parte de los retos a solucionar en este caso es el poder darle mantenimiento o cambios a los restaurantes sea a platillos o restaurantes. Se debe poder agregar nueva información como nombre, sabor, país de origen y la lista de ingredientes.

Con todos estos retos planteados se trabaja en la solución. Para poder manejar los datos de ingreso y los de consulta se requiere de una base de conocimientos. Con esto nos abrimos paso en Prolog, con dicho lenguaje se puede buscar lo que se desea en base los parámetros que se requieran o las condiciones que deban presentarse. Pero esta debe trabajar en conjunto con otro lenguaje en este caso Python, al ser un lenguaje multiplataforma y con amplías librerías se puede realizar la unión con la funcionalidad. En python se puede trabajar ya con la cara de la aplicación, con la que el usuario tendrá interacción. Es necesario que esta sea amigable y sobre todo lo más alto nivel de abstracción posible. La aplicación debe estar diseñada para procesar todos los datos y realizar búsquedas según sea compatible a realizar.

# Propósito(descripción del app, requerimientos)

El programa tiene dos funcionalidades, agregar datos a la base de conocimientos y consultarlos. Ya están establecidas las reglas para la base de conocimientos por las que simplemente se cargan al inicio. El programa primeramente interactúa con un una aplicación móvil que permita entre las funcionalidades consultar y agregar. Esta aplicación estará conectada por servidor local y estará diseñada para que se pueda ver correctamente desde un celular. Se necesita acceso a internet y contar con una computadora o dispositivo inteligente (celular, tablet, etc).

# 

|  |  |  |
| --- | --- | --- |
|  | *Funcionalidades* |  |
| Función | Parámetros | Uso |
| cargarBaseConocimiento | No recibe. | Se utiliza para cargar hechos y reglas en la base de conocimientos. |
| platillo | Nombre del platillo, nombre del restaurante, sabor del platillo, país de origen e ingredientes. | Se utiliza para agregar platillos a la base de conocimientos. |
| auxIngredientes | Cadena de ingredientes. | Se utiliza para acomodar los ingredientes separados por “\_” en una nuevo string llamado cadena. |
| restaurantesPorPais | Nombre del país. | Se utiliza para buscar restaurantes que poseen platillos del país dado como parámetro. |
| platillosPorRestaurante | Nombre del restaurante. | Se utiliza para dar todos los platillos que ofrece el restaurante dado como parámetro. |
| platillosPorIngredientes | Nombre del restaurante y el ingrediente. | Se utiliza para conocer los platillos del restaurante del primer parámetro que contienen el ingrediente del segundo parámetro. |
| restaurante | Nombre del restaurante, tipo de comida, ubicación, teléfono y horario. | Se utiliza para agregar restaurantes a la base de conocimientos. |
| buscarRestaurantePorComida | Tipo de comida. | Se utiliza para ver los restaurantes que cuentan con el tipo de comida dado como parámetro. |
| mostrarRestaurantes | No recibe. | Se utiliza para ver todos los restaurantes en la base de conocimientos. |
| imprimir | Una lista. | Se utiliza para imprimir la lista que se da por parámetro. |

# Descripción detallada (decisiones de diseño, lenguajes de programación usados, tecnologías y librerías usadas)

La plataforma en la que trabajaremos es en Linux por lo que todos los pasos aquí descritos de instalación y manipulación serán explicados para este sistema operativo.

Para el desarrollo del proyecto se programará en dos lenguajes de programación: Python 2.7 y SWI-Prolog 6.0.2

La creación de la aplicación móvil se realizó con Flask como framework. Se implementó la versión 0.10.1. Se decidió utilizar dicha librería por recomendación del profesor y otros consultados. Además por qué fue más fácil el aprendizaje y manejo a diferencia de Jango, que estaba más estructurada para una base de datos y tenía un enfoque distinto al manejo para el que requeríamos con prolog.

Para la creación de templates utilizamos la plantilla de HTML, porque es el formato predeterminado que se utiliza en Flask para la página de la aplicación.

En la conexión o interacción entre Python y SWI-Prolog hemos decidido utilizar Pyswip 0.2.3 que según nuestro jucio es más estable que el 0.2.2 y que además es compatible con ordenadores de 64bits.

Dentro de la librería de Pyswip existe una funcionalidad con “assertz” para agregar a la base de conocimiento y “query” para consultas.

Para la mayoría de funciones decidimos manejar los datos en listas por la facilidad de manejo. Por la facilidad de buscar elementos en una lista con la función predeterminada “in” y para los ciclos con while con “len” para saber el largo de dicha lista. Además de la no restricción de manejo del largo de lista permitiendo hacer “.append” a la lista cada vez de ser necesario.

El manejo de varias cadenas de texto, por ejemplo para el caso de ingredientes en platillos, se decidió utilizar una misma cadena de texto separada por “\_” como identificador de elemento a elemento. Inicialmente se había planeado almacenarlo como una lista en la base de conocimientos, pero obteníamos un error similar a Atom[985678] por lo que se decidió no utilizar listas en la base de conocimientos.

# Problemas encontrados

Poca documentación para Pyswip y flask reciente por lo que aumentó el tiempo de aprendizaje para el manejo y programación sobre esas librerías.

La instalación de Pyswip fue la más complicada, debido a que su versión 0.2.2 sólo sirve para ordenadores con arquitectura de 32bits y la extensa cantidad de pasos para lograr funcionar su instalación. El problema principal radicó en que la librería swipl.pl debía ser una librería compartida(Enable shared) y la cantidad de intentos por una solución fue numerosa.

Para la realización de la página web se requirió mucha investigación sobre HTML, debido a ser la primera vez manipulando dicha herramienta por todos los miembros del grupo y con esto hacer funcionar la plantilla. En HTML la creación de templates y manejo de datos fueron los principales retos y dificultades.

# Manual de usuario

**Guía de instalación:**

1. Abrir una terminal.
2. Ejecutar los siguientes comandos para la descarga y configuración de pyswip versión 0.2.3 :
   1. wget <http://pyswip.googlecode.com/files/pyswip-0.2.3.zip>
   2. unzip pyswip-0.2.3.zip
   3. cd pyswip-0.2.3/
   4. sudo python setup.py install
3. Ejecutar los siguientes comandos para la descarga y configuración de swi-prolog:
   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.tar.gz
   4. ./configure --enable-shared
   5. \*\* CFLAGS=-ggdb ./configure --enable-shared
   6. make
   7. sudo make install
   8. cd packages/clpqr
   9. ./configure --enable-shared
   10. make && make install
4. Ubicarse en el directorio donde contiene los archivos .py y ejecutar en la terminal:
   1. python tp.py

**Guía de uso:**

1. **Página principal del sistema**: Cuenta con las opciones de consultar todos los restaurantes registrados, restaurantes por tipo de comida, buscar restaurantes por nombre y restaurantes que cuenten con platillos de algún país especificado.



1. **Agregar restaurante:** En esta sección se pueden agregar nuevos restaurantes al sistema. Se debe especificar el nombre, tipo de comida, ubicación, teléfono y horario.



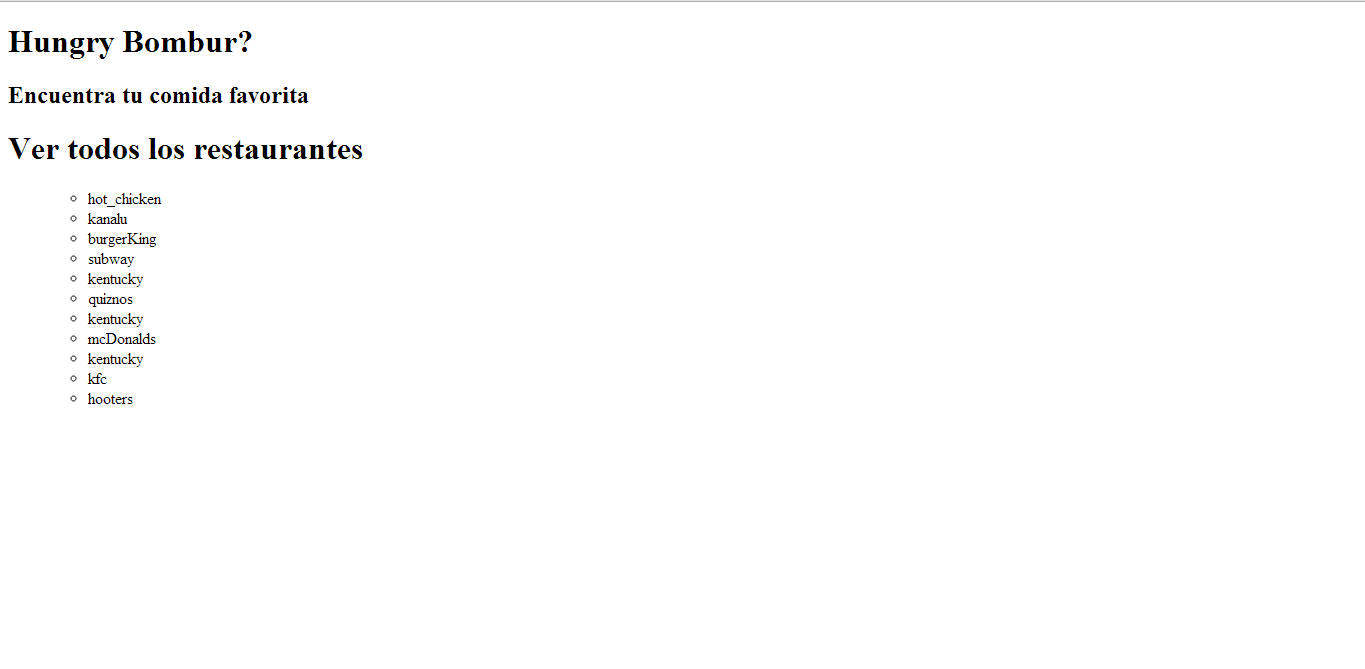


1. **Agregar platillo:** En esta página el sistema permite agregar nuevos platillos a los restaurantes. La información que se solicita es: nombre, restaurante al que pertenece, sabor, país de origen del platillo y los ingredientes.

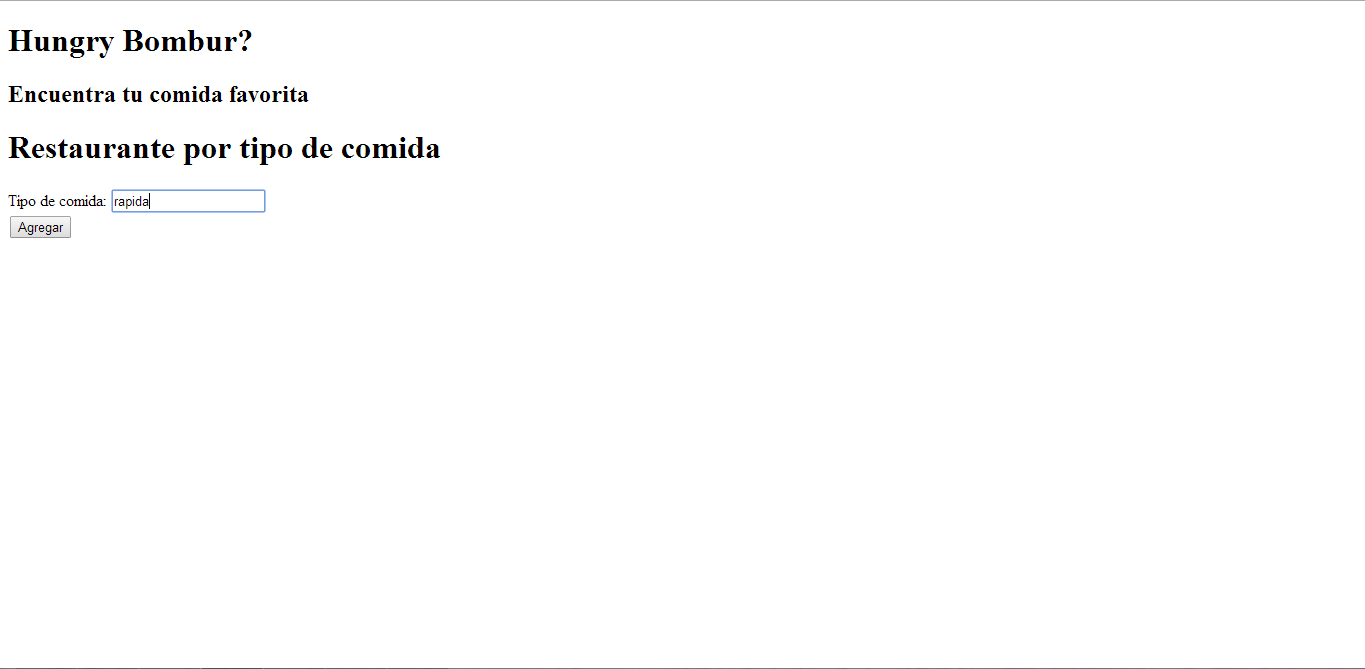


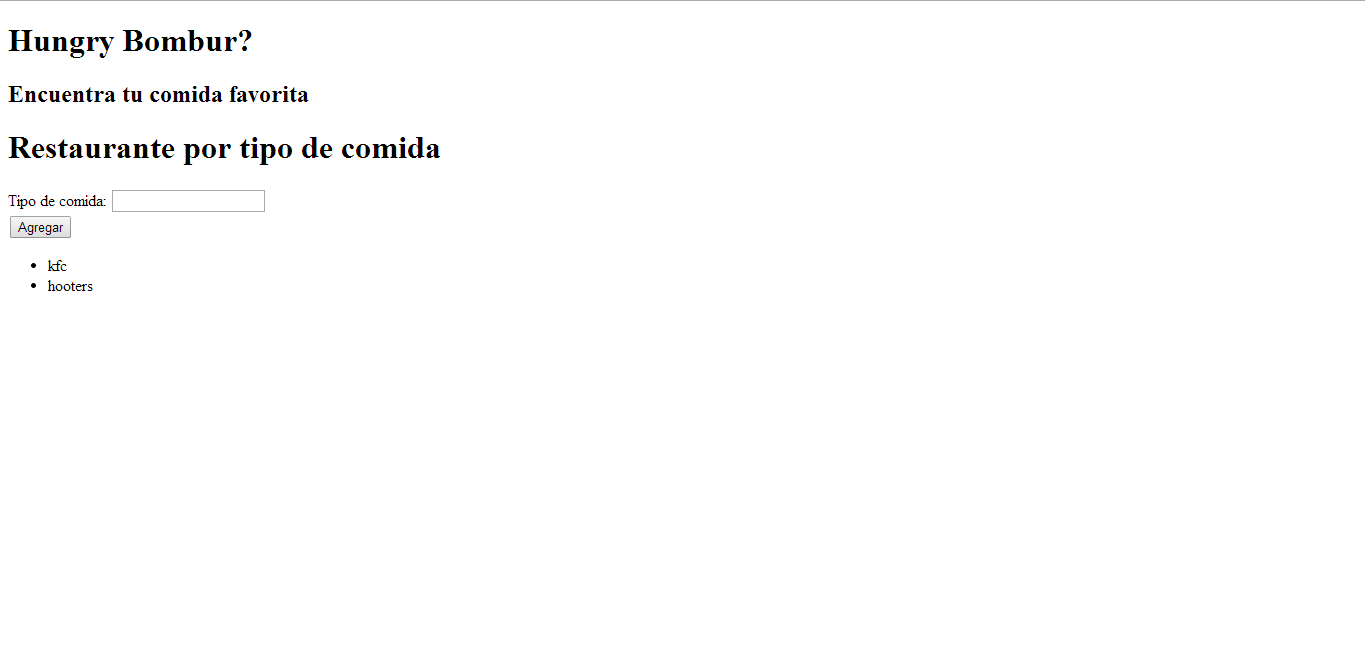


1. **Ver todos los restaurantes:** En esta parte del sistema se pueden observar todos los restaurantes registrados.



1. **Restaurantes por tipo de comida**: Esta sección del sistema solicita al usuario algún tipo de comida y retorna todos los restaurantes que tengan el tipo especificado.





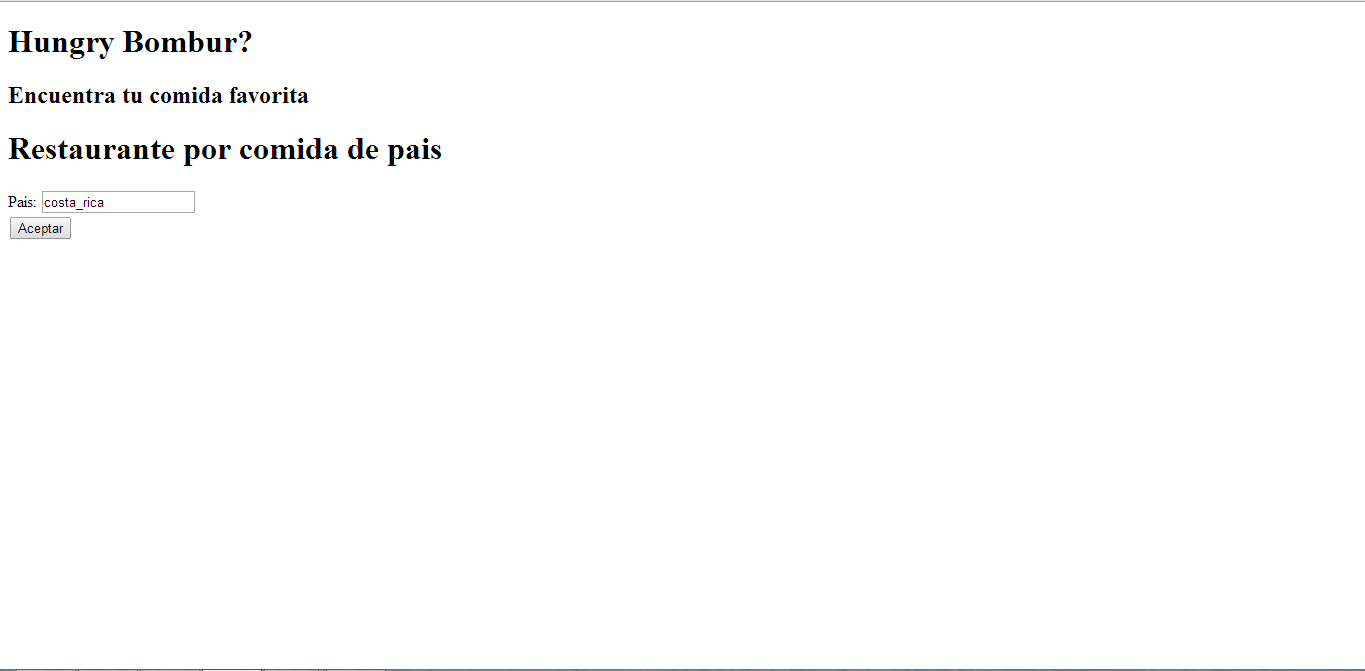
1. **Búsqueda de restaurante por nombre:** Se especifica el nombre del restaurante y el

sistema devuelve toda su información.

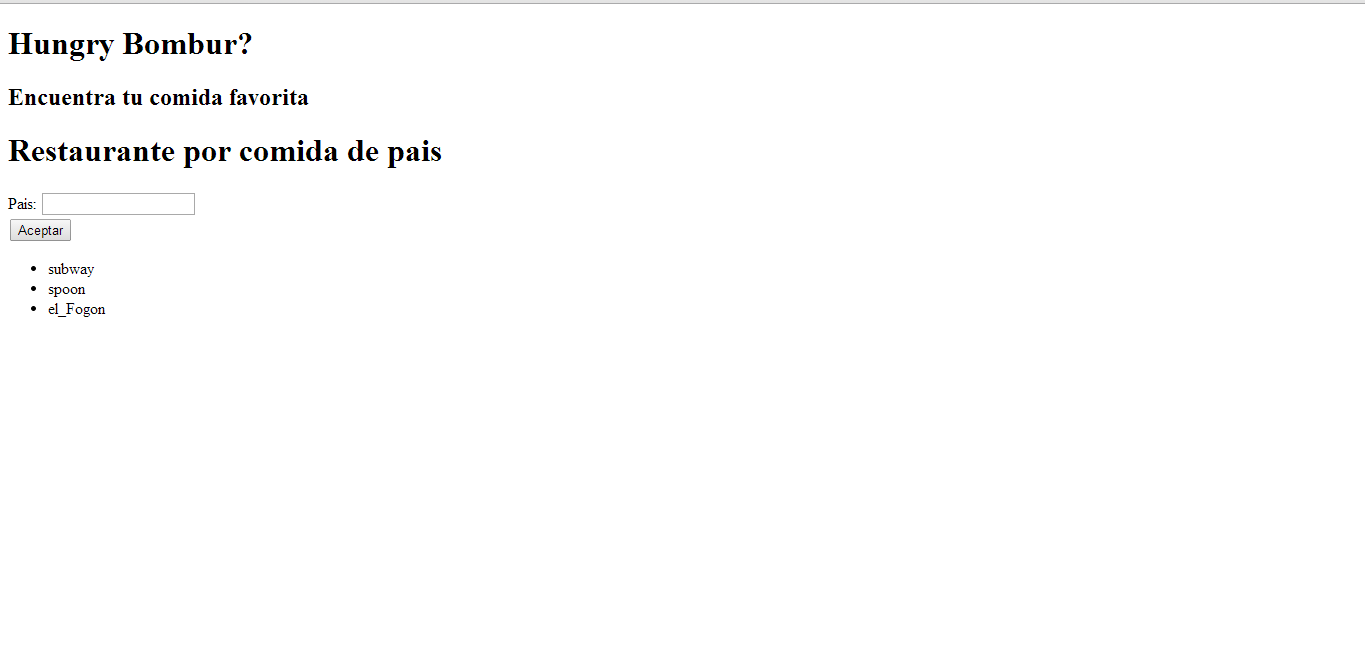




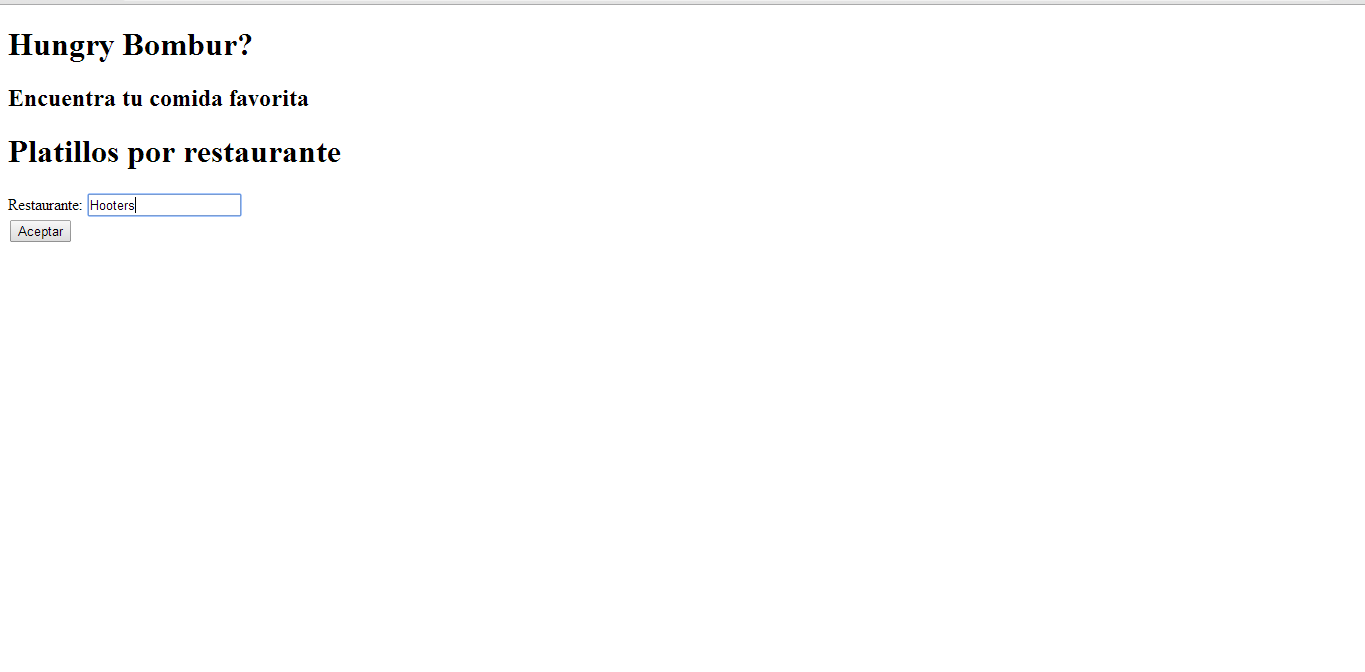
1. **Restaurantes que cuentan con platillos de un país específico:** En esta parte se ofrece al usuario la opción de conocer todos los restaurantes que cuentan con platillos de algún país.



.

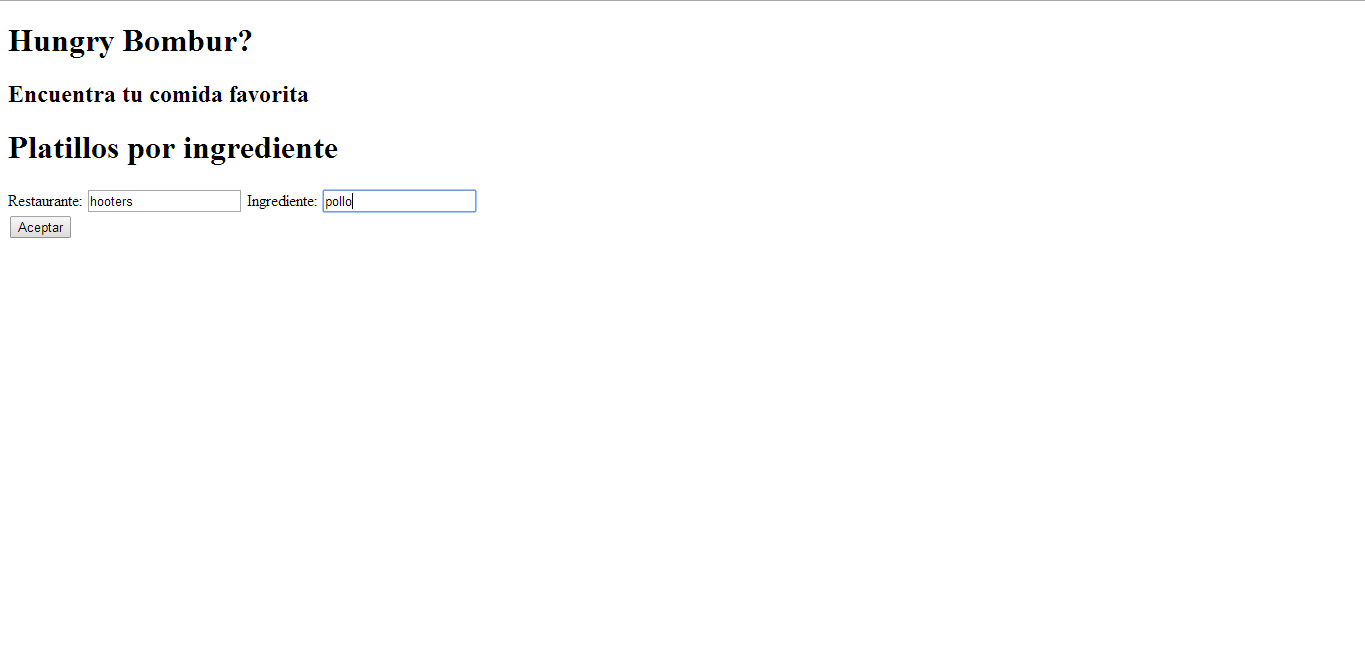


1. **Lista de platillos:** En esta opción del sistema se pueden consultar todos los platillos que posee algún restaurante.





1. **Platillos de algún restaurante que tengan algún ingrediente:** En esta funcionalidad, el sistema, tiene la opción de buscar los platillos del restaurante seleccionado que cuenten con el ingrediente también especificado.



**Bibliografía**

<https://pypi.python.org/pypi/pyswip/0.2.2>

http://code.google.com/p/pyswip/downloads/list

http://code.google.com/p/pyswip/wiki/Examples