

Universidad Peruana de Ciencias Aplicadas

**Trabajo Final**

Curso:

Complejidad Algorítmica

Docente:

Patricia Daniela Reyes Silva

Sección:

WV62

Tema Abordado:

Tarifa de las aplicaciones de taxis ofertadas en Lima Metropolitana

Integrantes:

|  |  |  |
| --- | --- | --- |
| Nombres | Código | Carrera |
| Giakomo Causso Rodolfo Mariano | U202118374 | Ingeniería de Software |
| Tony Alexander Ramos Najar | U20211A153 | Ingeniería de Software |
|  |  |  |

Abril 2022

1. Descripción del Problema

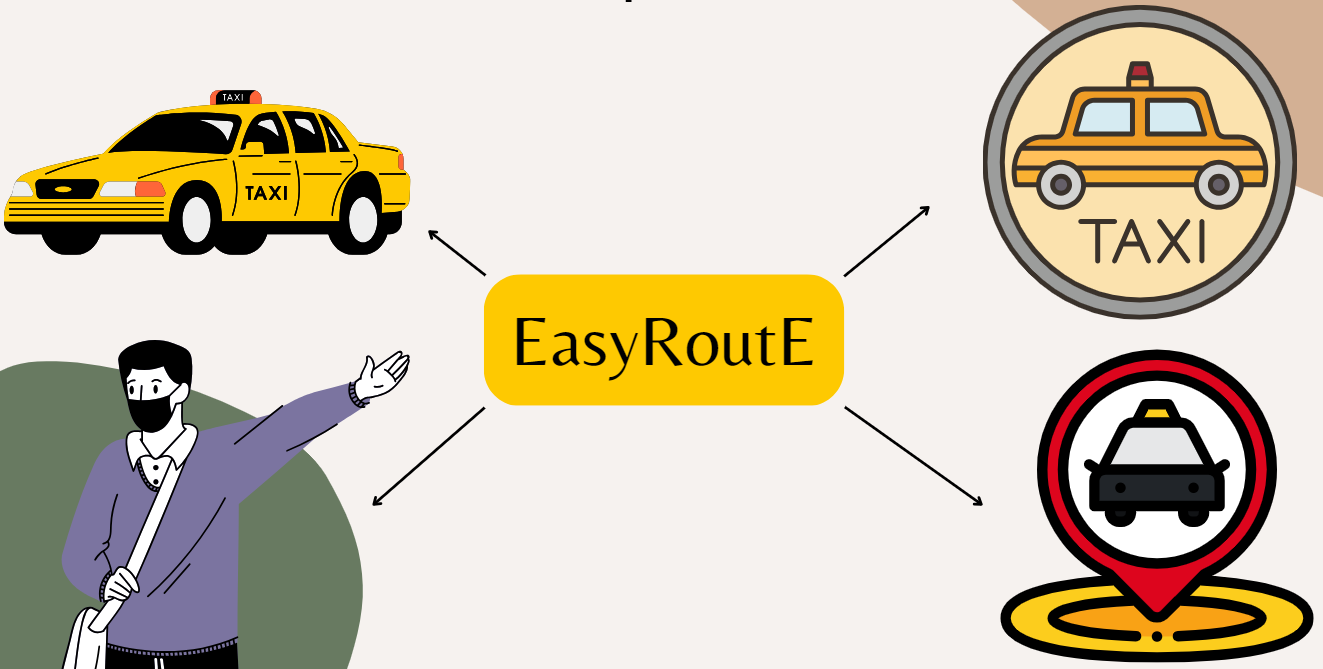
La elección de un servicio de taxi en Lima Metropolitana puede resultar complicada debido a la gran cantidad de opciones disponibles y a la variación de precios entre ellas. El objetivo principal de nuestro proyecto es facilitar esta elección al proporcionar a los usuarios información precisa y actualizada sobre las tarifas ofrecidas por las diferentes aplicaciones de taxis en la ciudad.

Para abordar este problema, hemos recolectado un dataset basado en datos reales de las aplicaciones de taxis más utilizadas en Lima. Nuestro principal parámetro de evaluación es el precio ofrecido por cada viaje o servicio, ya que consideramos que es un factor clave para determinar la opción más conveniente desde un punto de partida en cualquier distrito de la ciudad hacia un punto final en un distrito específico.

Al considerar el precio como factor principal, nuestro objetivo es identificar y comparar las tarifas más bajas disponibles para cada tipo de servicio que ofrecen las aplicaciones de taxis en Lima. Esto permitirá a los usuarios seleccionar la opción más económica y acorde a su presupuesto, optimizando así sus costos de transporte.

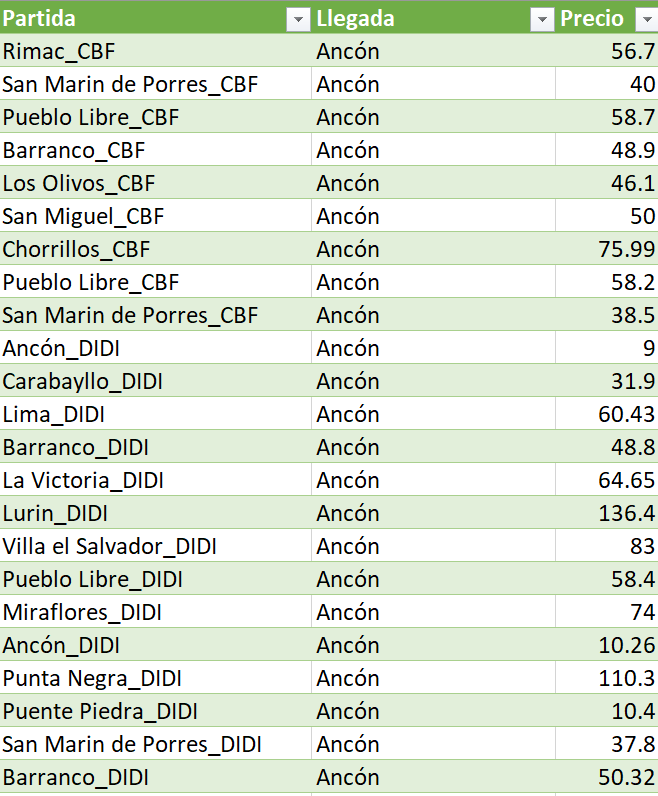
Es importante destacar que nuestro dataset se basa en datos reales, lo que garantiza la validez y confiabilidad de los resultados obtenidos. Actualizamos periódicamente nuestra base de datos para reflejar los cambios en las tarifas y las nuevas aplicaciones de taxis que se lanzan en la ciudad.

En resumen, el problema que abordamos es la elección de un servicio de taxi en Lima Metropolitana, considerando el precio como factor principal. Utilizando datos reales de las aplicaciones de taxis disponibles en la ciudad, nuestro objetivo es proporcionar a los usuarios información actualizada y confiable sobre las tarifas más bajas, permitiéndoles tomar decisiones informadas y ahorrar costos en sus desplazamientos.



1. Descripción y visualización del conjunto de datos (Generación del dataset)

Como previamente enfatizamos en la descripción de problema, recolectamos datos de nuestro dataset basándonos en datos reales de las aplicaciones que están disponibles actualmente de taxis en Lima, seleccionamos como principal parámetro al precio que ofrece cada uno, para poder determinar el menor de ellos dependiendo del servicio que desee el usuario. Tomamos como referencia un punto de partida o inicial de cualquier distrito de Lima Metropolitana, y un punto final o llegada de un distrito específico, al cual el usuario se deseará dirigir.

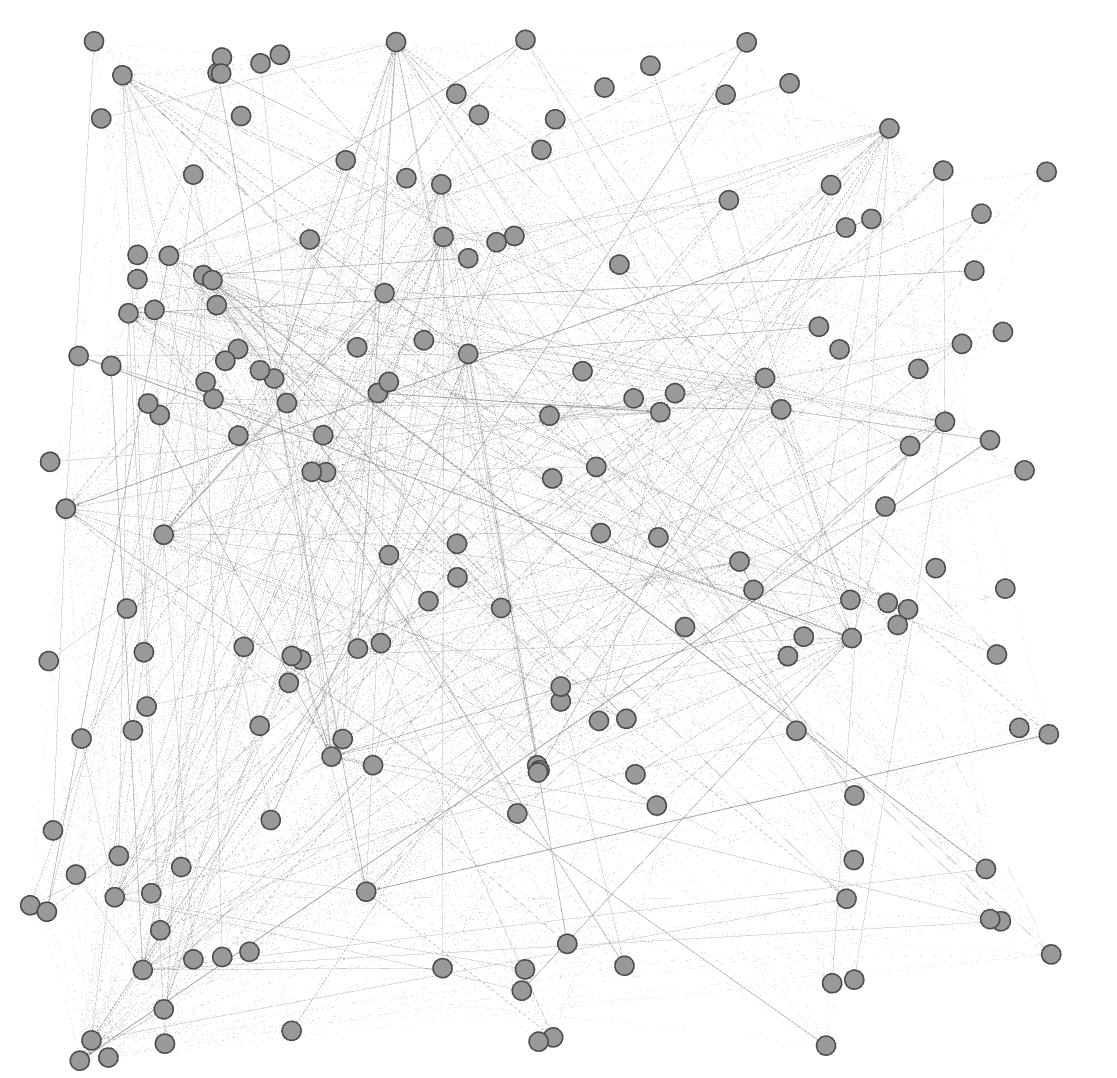


Significado de los acrónimos:

|  |  |
| --- | --- |
| Código | Significado |
| \_CBF | Aplicación de taxis Cabify |
| \_DIDI | Aplicación de taxis Didi |
| \_ETX | Aplicación de taxis EasyTaxi |
| \_INDR | Aplicación de taxis Indrive |
| \_TXB | Aplicación de taxis TaxiBeat |
| \_UBER | Aplicación de taxis Uber |

Asimismo, para la creación del grafo general, existirá un nodo final destino de cada distrito, por ejemplo, Ancón será un destino final. Por otra parte, los nodos iniciales o puntos de partida llegarán hacia el nodo final y la arista de cada uno de ellos representará el precio real que cada aplicación ofrece al usuario, un nodo inicial será un distrito con el código de aplicación que nuestra aplicación EasyRoute ofrece.

A continuación, presentamos nuestro gráfico de nuestro grafo completo:



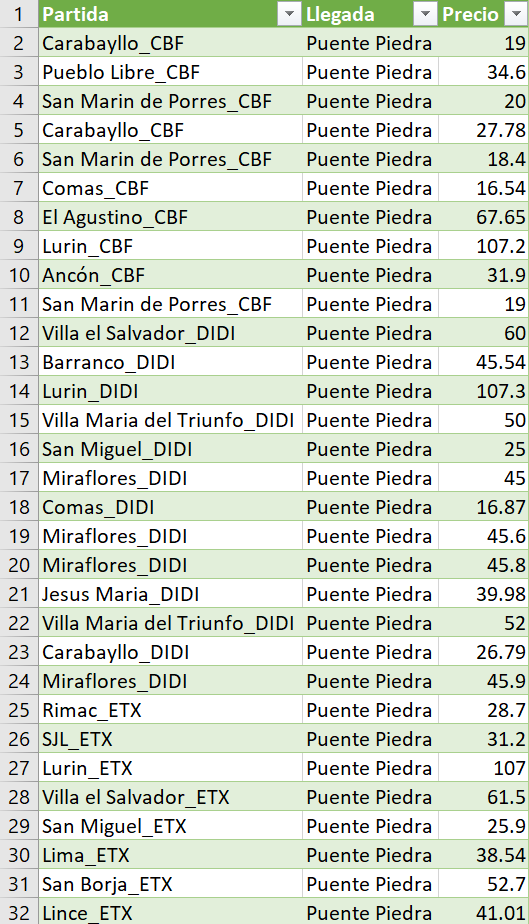
Dataset Reducido:

Con respecto a nuestro dataset reducido, solo tomamos un único nodo final y diversos nodos iniciales que llegarán a él, como puntos de partida.

Por ejemplo:

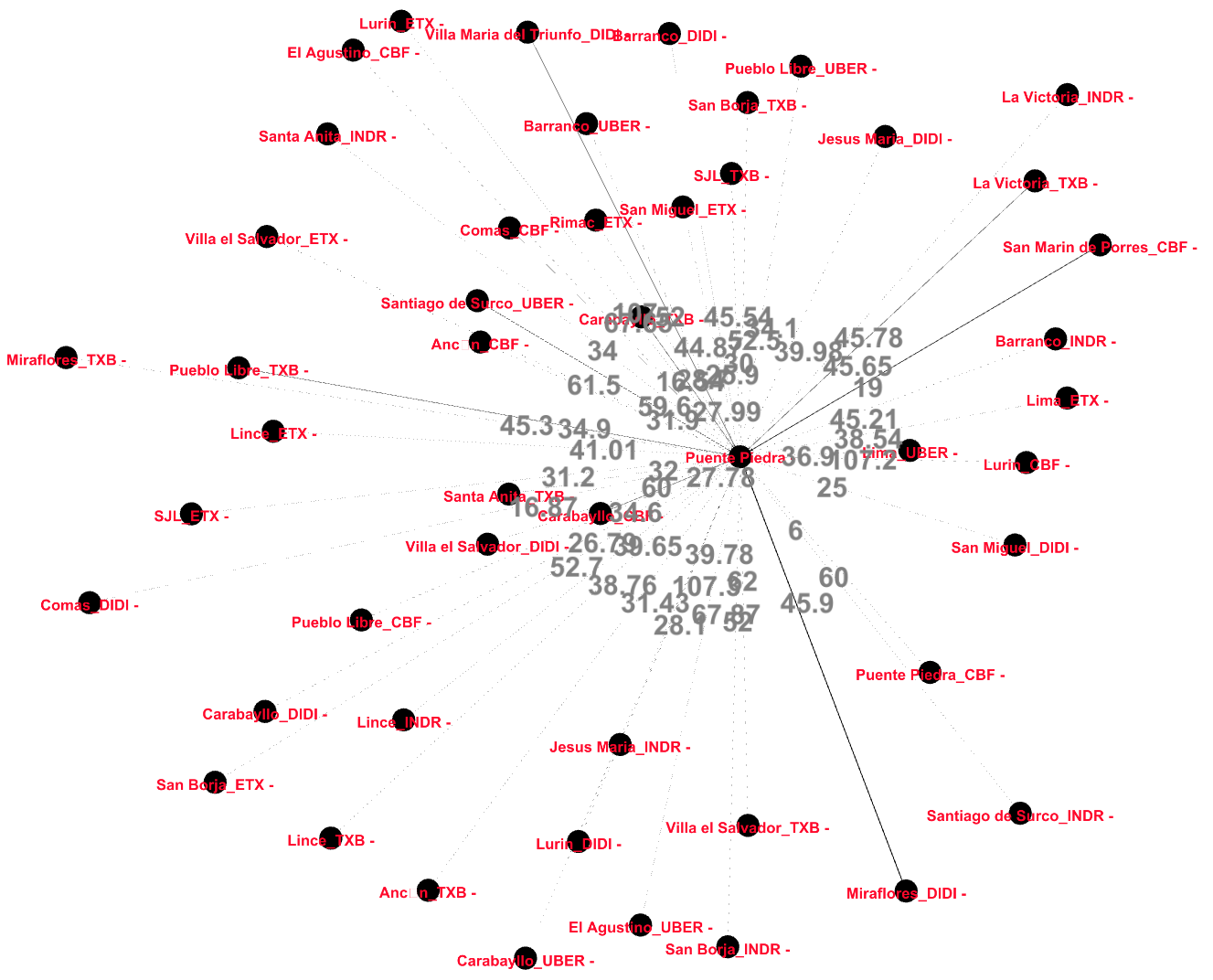
Para este caso, supongamos que el usuario se quiera dirigir al distrito de Puente Piedra, para esto el usuario tendrá la opción de poder digitar en que distrito se encuentra actualmente. Al digitar su distrito en el que se ubica, nuestra aplicación le devolverá una información con la cantidad de aplicaciones de taxis que ofrecen servicio desde el distrito que se encuentra hasta su distrito objetivo o final, con sus respectivos precios. El usuario podrá seleccionar qué precio ofertado le conviene más y con qué aplicación.

Ejemplo de nuestro DataSet Reducido:



Para la creación de nuestro grafo reducido, solamente conectamos los nodos iniciales o puntos de partida al nodo destino o final, con sus respectivas aristas, que vendrían a ser los precios que ofrecen cada aplicación de taxis.

Gráfico de nuestro grafo reducido:



Como se puede observar, el único nodo destino es Puente Piedra, y a partir de este nodo, tiene muchos otros nodos que llegan a él a través de aristas, ofertando diversos precios.

1. Propuesta

Presentamos una propuesta innovadora para facilitar a los usuarios de Lima Metropolitana la búsqueda de la opción de taxi más económica en base a sus necesidades de viaje. Nuestro proyecto, denominado "EasyRoutE", tiene como objetivo principal proporcionar a los pasajeros una herramienta eficiente que compare los precios de las aplicaciones de taxi más populares, como Didi, Uber, EasyTaxi, InDrive y Taxibeat, con el fin de mostrarles las opciones más económicas y convenientes.

El objetivo de EasyRoutE es desarrollar una aplicación que permita a los usuarios ingresar su punto de partida y destino deseado, y obtener una lista de opciones de costos provenientes de diferentes aplicaciones de taxis, mostrando el monto estimado para cada opción. La aplicación se encargará de recopilar y analizar los datos de precios, brindando a los usuarios una forma rápida y sencilla de comparar y seleccionar la opción más económica para su viaje.

Para lograr nuestro objetivo, implementaremos dos técnicas principales: fuerza bruta y metodología de grafos. La técnica de fuerza bruta nos permitirá recopilar y procesar los datos de precios, asegurando que se analicen todas las opciones disponibles. Por otro lado, la metodología de grafos se utilizará para representar (mostrar) las rutas posibles y ofrecer el precio estimado.

EasyRoutE ofrecerá una interfaz intuitiva y fácil de usar. Los usuarios simplemente ingresarán su ubicación actual y su destino deseado, y la aplicación generará una lista de opciones de precios para cada aplicación de taxi. Además de mostrar los costos estimados, también se proporcionará información detallada sobre el tiempo estimado de viaje y la ruta seleccionada.

EasyRoutE es una solución completa y eficiente para los usuarios de Lima Metropolitana que desean ahorrar tiempo y dinero en sus viajes en taxi. Con la capacidad de comparar rápidamente los precios de las principales aplicaciones de taxi y acceder a información relevante sobre las rutas, los pasajeros podrán tomar decisiones informadas y disfrutar de viajes cómodos y económicos.

1. Diseño del aplicativo

**Características de la interfaz:**

* Generamos un diseño de prototipo generado en Figma, para tener una idea clara y concisa de lo que queremos llegar a concretar en nuestro proyecto.
* Logramos implementar dicha idea del Figma a una interfaz creada por Tkinter
* En la interfaz de tkinter, concretamos una interfaz interactiva por el usuario, además de aplicar correctamente la idea principal de nuestro proyecto
* Con respecto a la aplicación, la interfaz solicita al usuario un punto inicial y un punto final, una vez digitada la solicitud, nuestra aplicación muestra todas las opciones disponibles para selección del usuario.

**Diseño y funcionamiento:**

Como previamente redactamos, nuestra interfaz principal está codificada en una librería gráfica estándar de Python, específicamente Tkinter. En ella, el usuario podrá digitar lo que está deseando y con los datos enviados, nuestra aplicación devolverá una lista de opciones que concuerdan con lo requerido por el usuario.

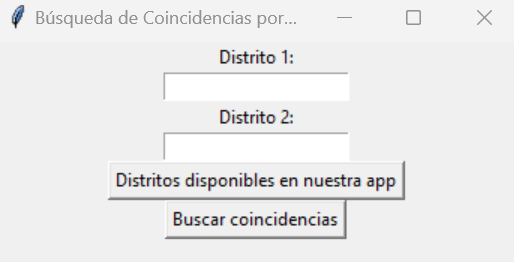
Si por algún motivo, en nuestra base de datos no existe ninguna coincidencia con lo requerido por el usuario, EasyRoute mostrará un mensaje de alerta y procederá a ofrecer rutas alternativas con precios reales.

El filtrado de precios es principalmente para que el usuario seleccione la opción más óptima para beneficio propio.

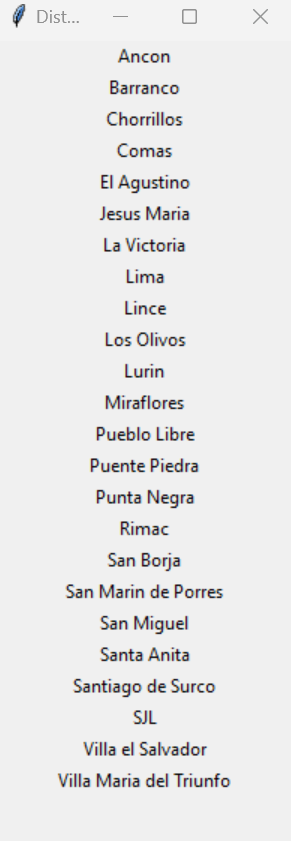
A continuación, presentaremos nuestra propuesta de prototipo diseñado en Figma:

<https://www.figma.com/file/YmXBUbmZFB97Uq8xtoafib/EasyRoutePROTOTYPE?type=design&node-id=0%3A1&t=QwYV86eFR3Y8aUH9-1>

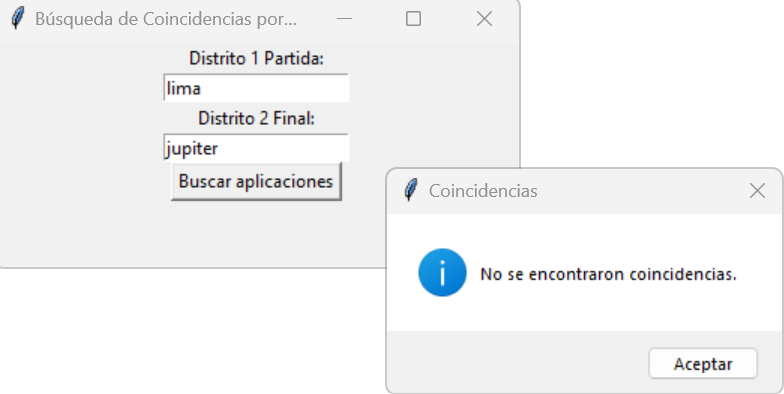
Como consecuencia de un prototipo, presentaremos algunas capturas de pantalla de nuestro interfaz final creado en Tkinter



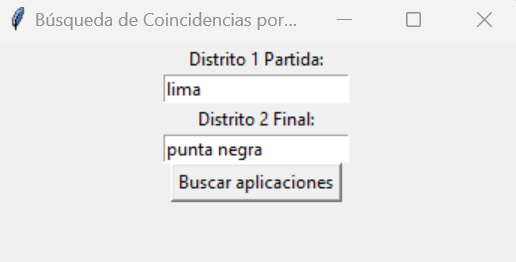
En este apartado se muestra 2 casillas para poder digitar, donde dice “Distrito 1” “Distrito 2” y complementando con 2 botones en el apartado inferior que dicen “Distritos disponibles en nuestra app” y “Buscar coincidencias”.



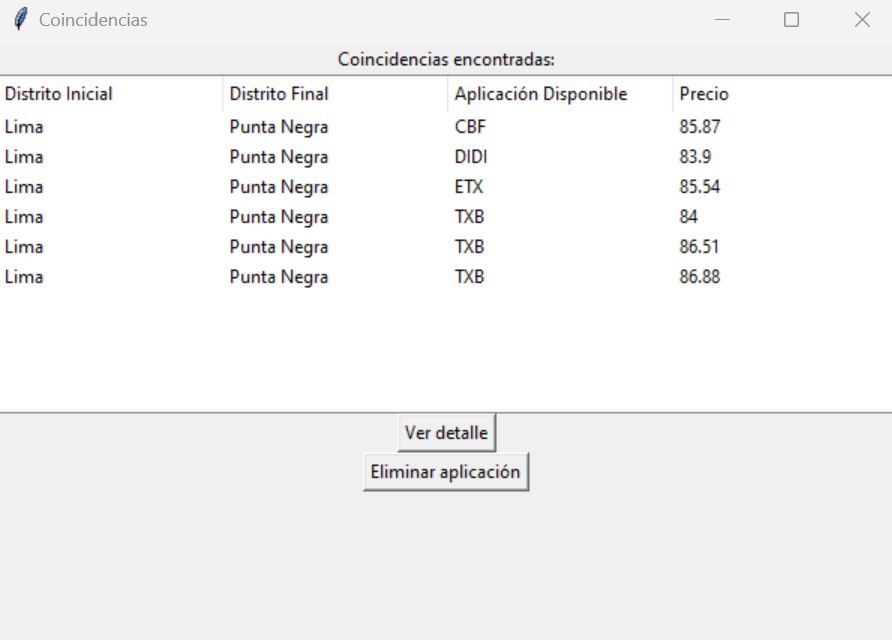
Cuando el usuario seleccione la opción de distritos disponibles en nuestra aplicación, se le mostrará una pestaña, en cual contenga todos los distritos de Lima Metropolitana que podrá digitar para acceder al servicio.



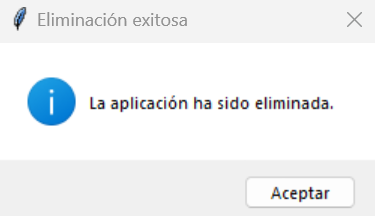
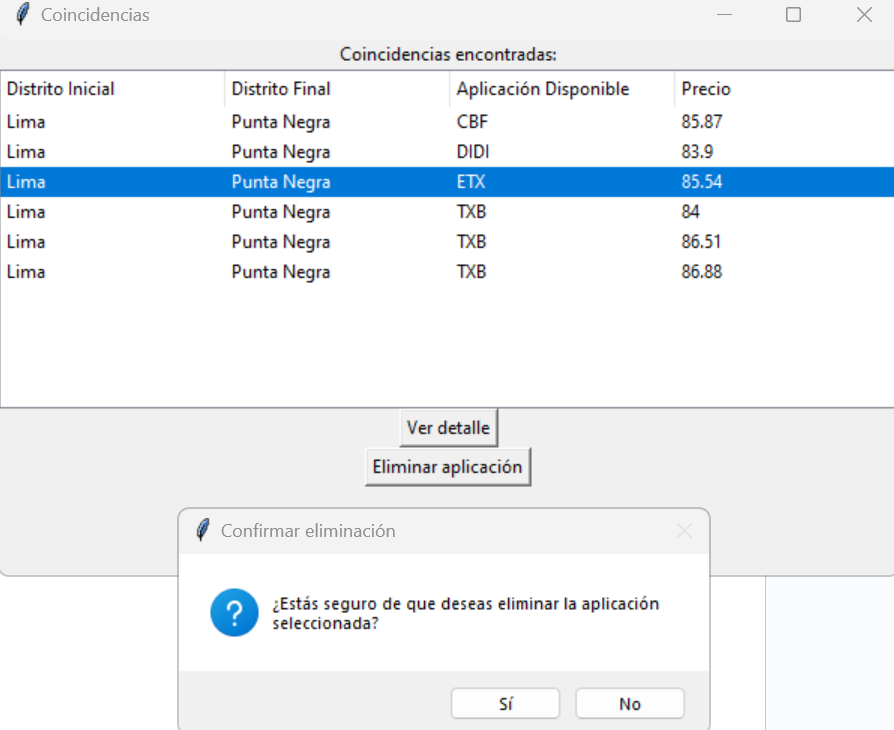
Obviamente en nuestra aplicación gracias a que tiene registrados todos los distritos disponibles, tiene restricciones a si el usuario desea ingresar un distrito inexistente (que no existe) se va a mostrar una ventana de advertencia de que no se encontró dicho distrito.



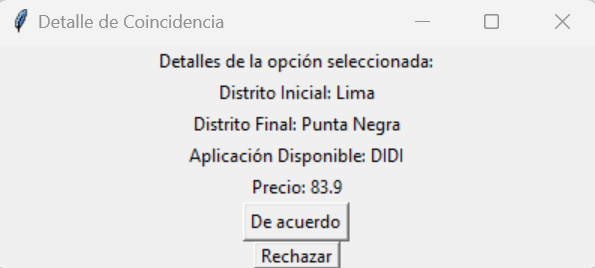
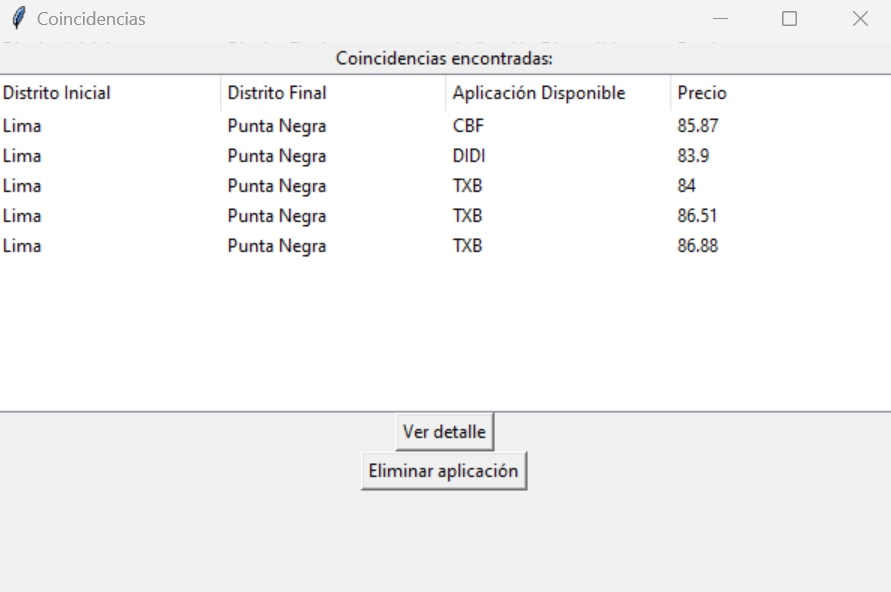
Este es el apartado de búsquedas de coincidencias el usuario podrá digitar el distrito de partida “Distrito 1 Partida:” y digitar el distrito a donde el usuario desea ir “Distrito 2 Final:” y un botón para buscar aplicaciones en el apartado inferior.



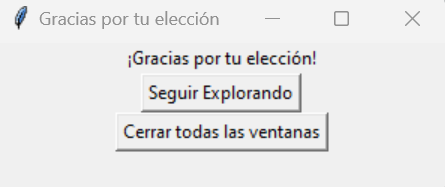
Al poner el distrito de inicio y fin va a parecer un recuadro de “Coincidencia encontradas” que va a mostrar el distrito inicial y distrito final que el usuario colocó, las aplicaciones disponibles para hacer dicho viaje y el precio respectivo de cada aplicación. En la parte inferior se puede ver 2 botones de “Ver detalle” y “Eliminar Aplicación”.



Si el usuario se arrepiente o desea salir de la app solamente le daría click al botón “Eliminar aplicación” y le saldrá una ventana emergente diciendo si esta seguro de realizar dicha acción, al aceptar ya se estaria saliendo.



Volviendo en este apartado si el usuario selecciona donde dice “Lima - Punta Negra - DIDI - 83.9” y le da click al boton de “ver el detalle” le va aparecer otra ventana que mostrará a detalle la opción seleccionada, el distrito inicial y final que colocó, la aplicación que integra en la opción seleccionada y el precio.



Y como última ventana al aceptar el viaje seleccionado, se mostrará una ventana de agradecimiento por el viaje aceptado y con 2 botones extras en la parte inferior para poder seguir explorando o cerrar todas las ventanas.

1. Validación de resultados y pruebas

**Validación de entradas y salidas:**

* + Entradas: Se ingresan diferentes combinaciones de distritos de partida y llegada en la interfaz gráfica.
  + Salidas esperadas: Coincidencias encontradas en el archivo CSV que cumplan con los distritos de partida y llegada ingresados.

**Interpretación de resultados:**

* + Se verifica si las coincidencias mostradas en la tabla corresponden a los distritos de partida y llegada proporcionados.
  + Comprueba si los valores de la columna "Aplicación Disponible" y "Precio" son correctos para las coincidencias encontradas.

**Pruebas de funcionalidad:**

* + Se verifica que se muestre un mensaje de error si el archivo CSV no se encuentra en la ubicación especificada.
  + Se ingresa distritos que no existen en el archivo CSV y verifica que se muestre un mensaje indicando que no se encontraron coincidencias.
  + Se selecciona una coincidencia de la tabla y verifica que se abra una nueva ventana con los detalles de la opción seleccionada.
  + Se intenta eliminar una aplicación seleccionada y verifica que se muestre un mensaje de confirmación. Luego verifica que la aplicación se haya eliminado correctamente de la tabla.

**Pruebas de rendimiento:**

* + Si el archivo CSV contiene una gran cantidad de datos, prueba el rendimiento de la aplicación ingresando diferentes combinaciones de distritos para verificar si la búsqueda de coincidencias se realiza de manera eficiente.

**Conclusiones**

En conclusión, en nuestro experimento utilizamos la técnica de fuerza bruta para encontrar el viaje de menor pago en EasyRoutE. Esta técnica nos permitió recopilar exhaustivamente los precios de diferentes aplicaciones de taxis y seleccionar la opción más económica para los usuarios. Sin embargo, reconocemos que la fuerza bruta puede ser computacionalmente costosa y puede no ser óptima en escenarios con grandes conjuntos de datos. Como trabajos futuros, se podrían investigar enfoques más eficientes, como algoritmos de búsqueda y optimización, que mejoren la velocidad y la escalabilidad del proceso de selección del viaje de menor precio.

Con lo que respecta al programa creado en Python con la interfaz Tkinter, se puede concluir que se aplicó una correcta interfaz gráfica del usuario (GUI) que permite al usuario ingresar los distritos de partida y llegada y realizar la búsqueda de coincidencias. La interfaz incluye botones, etiquetas y cuadros de entrada para interactuar.

Se utilizó el manejo de archivos CSV, el código utiliza la biblioteca “csv” de Python para leer un archivo CSV que contiene datos de partida, llegada, aplicación disponible y precio. Se recorren las filas del archivo CSV y se comparan los distritos de partida y llegada con los ingresados por el usuario para encontrar coincidencias aplicando el algoritmo de fuerza bruta (recorrer todas las coincidencias)

Por último, se concluye que nuestro proyecto ofrece varias formas de interactuar con el usuario. Al seleccionar una coincidencia en la tabla, se puede ver el detalle de la opción seleccionada en una nueva ventana. También se proporciona la opción de eliminar una aplicación seleccionada de la tabla. Además, se muestra un mensaje de agradecimiento al usuario al finalizar la interacción

**Link de la video exposición del proyecto:**

<https://youtu.be/R8lMWfpmuHM>

**Link del repositorio del TF en GitHub:**

<https://github.com/TonyRamosNajar/TF_Complejidad.git>

***Bibliografía***

Grandjean, M. (n.d.). GEPHI – Introduction to Network Analysis and Visualization [new video] | Martin Grandjean. <https://www.martingrandjean.ch/gephi-introduction/>

Orlando. (2021). Grafos | Qué son, tipos, orden y herramientas de visualización. GraphEverywhere. <https://www.grapheverywhere.com/grafos-que-son-tipos-orden-y-herramientas-de-visualizacion/>