Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Selección de vuelos

Flight selection

Miguel Bravo Arvelo

La Laguna, 11 de septiembre de 2020

D. **Juan José Salazar González**, con N.I.F. 43.356.435-D profesor Titular de Universidad adscrito al Departamento de Estadística, I.O. y Computación de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

“Selección de Vuelos”

ha sido realizada bajo su dirección por D. **Miguel Bravo Arvelo**,

con N.I.F. 51.152.438-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de septiembre de 2020

Agradecimientos

Querría agradecer a Juan José Salazar González por ofrecer la posibilidad de realizar este proyecto y también querría agradecer a mis padres y mi hermana por su acompañamiento y por ser mi fuente de inspiración en la vida, a ellos les quiero dar mi sincero agradecimiento por apoyarme durante esta etapa.

Licencia

Resumen

El objetivo de este proyecto consiste en desarrollar una herramienta que optimiza los recursos necesarios para la realización de entrevistas a vuelos realizados en Canarias para cumplir con las entrevistas periódicas del Frontur en cada uno de los aeropuertos canarios, tarea que lleva siendo realizada desde el 2009 por el ISTAC, hasta donde se conoce, la selección de vuelos y la asignación de encuestadores se realiza manualmente por parte de encargados en los institutos de estadística. No se tiene constancia de ninguna herramienta informática que ayude a los decisores a gestionar sus recursos durante la planificación de estas encuestas, a pesar de que se realiza mensualmente en institutos autonómicos.

Por ello se ha decidido desarrollar una herramienta informática que automatice el proceso teniendo en cuenta que el usuario final no es de un perfil técnico y además, ya que se desconoce el sistema operativo prioritario usado por los empleados de la organización la implementación debe de intentar ser lo más independiente del sistema en la medida posible.

Palabras clave: Turismo, ISTAC, FronTur, Optimización Combinatoria, Data Science

Abstract

The objective of this project consists on developing a tool that optimizes the necessary resources necessary for the completion of polls done to flights on the Canary Islands so it accomplishes the periodic Frontur interviews on each airport of the Canary Islands, task that has been done since 2009 by the ISTAC, from where is known, the selection of flights and the assignment of pollsters is done manually by workers on the statistic institutes. There is no known evidence of any software tool used to aid the decision makers on their resource management during the planification of said polls, despite it being done monthly on autonomic institutes.

Due to this it has been decided to develop a software tool to automate the process having on account that the end user is not one of a technical background and also, because the proprietary operating system of the employees of the organization is unknown the implementation should try to be as independent of the system as possible.

Keywords: Tourism, ISTAC, FronTur, Combinatorial Optimization, Data Science

Índice general

Índice de figuras

Capítulo 1 Introducción

1.1 Motivación

La empresa de Aeropuertos Españoles y Navegación Aérea(AENA) periódicamente determina una semana en la que se deben realizar las encuestas con un número determinado de encuestadores, para llevar esta tarea a cabo Aena proporciona la información generada por GESLOT, una herramienta de coordinación de los sistemas operacionales de los aeropuertos la cual suministra información de las franjas horarias autorizadas para cada vuelo, a las entidades involucradas en la realización de las encuestas, en este caso el ISTAC.

Cada vuelo tiene sus aeropuertos de salida y llegada. Los aeropuertos están agrupados por países en el caso de ser vuelos internacionales. Los destinos nacionales(aeropuertos) se consideran cada uno como un país, salvo los canarios que se estudian como un único país, dadas las características que se le aplican a la Comunidad Autónoma de Canarias (CAC) se puede desarrollar una herramienta desarrollada para resolver este caso, con la posibilidad de poder llegar a generar una herramienta útil que pueda llegar a ser usada en la gestión de esta tarea.

1.2 Alcance

Diseñar un módulo con la capacidad de resolver el problema y desarrollar herramientas basadas en este para proporcionar interfaces de uso amigables al usuario final.

1.3 Entidades relacionadas

El Instituto de Estadística Canario (ISTAC) es el órgano central del sistema estadístico autonómico y centro oficial de investigación del Gobierno de Canarias de estadística de la Comunidad Autónoma de Canarias(CAC), y que entre otras, le asigna las siguientes funciones:

- Proveer información estadística con independencia técnica y profesional información estadística de interés de la Comunidad Autonómica de Canarias
- Coordinar la actividad estadística pública, siendo responsable de la promoción, gestión y coordinación de la actividad estadística pública de la Comunidad Autónoma de Canarias

Frontur es el acrónimo dado a la encuesta de movimientos turísticos en fronteras y consiste en la operación estadística de la Subdirección General de Conocimientos y Estudios Turísticos que recoge datos relativos a la entrada en España de visitantes no residentes en España.

La estadística de Movimientos turísticos en fronteras canarias (FronTur-Canarias)

ofrece datos mensuales de los turistas y excursionistas entrados en cada una de las islas canarias, permitiendo la comparación con los datos nacionales y de otras comunidades autónomas.

El Instituto de Turismo de España(Turespaña) organismo autónomo de la Administración General del Estado encargado de la promoción en el exterior de España como destino turístico, que en el año 2009 organizó un convenio con el ISTAC para llevar a cabo la Encuesta de Movimientos Turísticos en Fronteras de Canarias.

Aena es la empresa pública española que gestiona los aeropuertos de interés general en España, así como la navegación aérea en algunos privados y algunas bases aéreas de régimen mixto con las Fuerzas Armadas de España.

El Instituto Nacional de Estadística es un organismo encargado de la actividad estadística pública, expresamente la realización de las operaciones estadísticas como censos demográficos y económicos, cuentas nacionales, estadísticas demográficas y sociales, indicadores económicos y sociales, coordinación y mantenimiento de los directorios de empresas, formación del Censo Electoral y demás.

1.4 Principales conceptos y definiciones

Según la metodología de 2016 sobre la estadística de movimientos turísticos en fronteras de Canarias los principales objetivos de la encuesta son:

I. Estimar el número de turistas procedentes del extanjero entrados en cada una de las islas según sus características.
II. Estimar el número de turistas procedentes del resto del territorio nacional entrados en cada una de las islas, según sus características.
III. Estimar el número de excursionistas entrados en cada una de las islas.

En relación con un país dado, pueden distinguirse tres formas de turismo:

- **Turismo interno**: el que comprende las actividades de los residentes en un país determinado que viajan y permanecen en lugares sólo en ese país, pero fuera de su entorno habitual.
- **Turismo receptor**: el que comprende las actividades de los no residentes de un país determinado que viajan y permanecen en lugares de ese país fuera de su entorno habitual.
- **Turismo emisor**: el que comprende las actividades de los residentes de un país determinado que viajan y permanecen en lugares, fuera de ese país y fuera de su entorno habitual.

En términos de las estadísticas turísticas regionales de Canarias denominaremos:

- **Turismo interno de Canarias**: el que comprende las actividades de los residentes en Canarias que viajan y permanecen en lugares de Canarias, pero fuera de su entorno habitual.
- **Turismo receptor de Canarias**: el que comprende las actividades de los no residentes en Canarias que viajan y permanecen en lugares de la región. Incluye tanto a los turistas residentes en el extranjero como turistas residentes en comunidades autónomas del resto de España.
- **Turismo emisor de Canarias**: el que comprende las actividades de los residentes en Canarias que viajan y permanecen en lugares, fuera de Canarias y fuera de su entorno habitual.

1.5 Diseño muestral y tratamiento de datos

Con el fin de poder cumplir con los objetivos de la Encuesta de Movimientos Turísticos en Fronteras en Canarias (FronTur-Canarias) se ha venido realizando desde 2009 una ampliación de la muestra estatal en la Comunidad Autónoma de Canarias. Para ello el Instituto Canario de Estadística (ISTAC) suscribió un convenio de colaboración con Turespaña como responsable en ese momento de FronTur. Las labores de recogida de datos de esa ampliación se realizaba conjuntamente con el Instituto de Estudios Turísticos (IET) e incluía la captura de datos de vuelos nacionales, con el fin de estimar el turismo nacional recibido en Canarias.

Tras el traspaso de responsabilidades, en 2015 el ISTAC suscribió un nuevo convenio de colaboración con el INE para la ampliación de la muestra de FronTur sobre turismo internacional en la Comunidad Autónoma de Canarias (CAC), al objeto de disponer de un tamaño muestral suficiente para desagregar la estadística por islas y por países de residencia. En ese sentido la muestra total del ámbito de Canarias queda constituida por la muestra diseñada por el INE y una muestra complementaria diseñada por el ISTAC. En este nuevo acuerdo la captura de datos de la ampliación muestral es realizada por el ISTAC, que a su vez es el responsable de la recogida de información de vuelos nacionales que queda fuera del acuerdo.

Dado que la estadística es mensual y no es posible llevar a cabo procedimientos de verificación de igualdad de estimaciones entre las muestras capturadas separadamente, para la integración de la muestra ampliada en el fichero nacional, el INE remite al ISTAC el fichero con la muestra general con el fin de que éste lo integre con la muestra ampliada y calcule los factores de elevación correspondientes, calibrando a las principales estimaciones nacionales en la Comunidad Autónoma.

La muestra final que se usa en FronTur-Canarias se compone de los vuelos seleccionados por el INE y la selección complementaria que realiza el Instituto Canario de Estadística (ISTAC) para cumplir con los objetivos asociados a FronTur-Canarias.

La muestra combinada de FronTur-Canarias se compone de encuestas a pasajeros en vuelos nacionales y vuelos internacionales con origen un aeropuerto canario.

A su vez, también se compone de los pasajeros aéreos en vuelos internacionales con origen en otro aeropuerto de España y de los pasajeros marítimos en los puertos canarios, siempre que estos pasajeros declaren haber sido turistas con destino principal o excursionistas en las Islas Canarias. Este tipo de pasajeros es detectado por el INE en su muestra nacional.

1.5.1 Evaluación y ajustes de datos

En el caso de los pasajeros canarios, éstos se incorporan como unidades muestrales pero la información que se recoge en la encuesta es de carácter básico. Esta información es necesaria para obtener los porcentajes de tipología de pasajeros en los vuelos. Por lo tanto la muestra se compone de las siguientes tipologías de unidades de última etapa:

I. Pasajeros residentes fuera de España, que denominaremos como “extranjeros”
   A. **Pasajeros extranjeros** en vuelos internacionales con origen Canarias
   B. **Pasajeros extranjeros** en vuelos nacionales con origen Canarias
   C. **Pasajeros extranjeros** en vuelos internacionales con origen en otros

aeropuertos de España

D. **Pasajeros extranjeros** en barcos que visitan Canarias

II. Pasajeros residentes en España, pero no en Canarias, que denominaremos como “españoles”

A. **Pasajeros españoles** en vuelos nacionales con origen Canarias

B. **Pasajeros españoles** en vuelos internacionales con origen Canarias

III. Pasajeros residentes en Canarias, que denominaremos como “canarios”

A. **Pasajeros canarios** en vuelos nacionales con origen Canarias

B. **Pasajeros canarios** en vuelos internacionales con origen Canarias

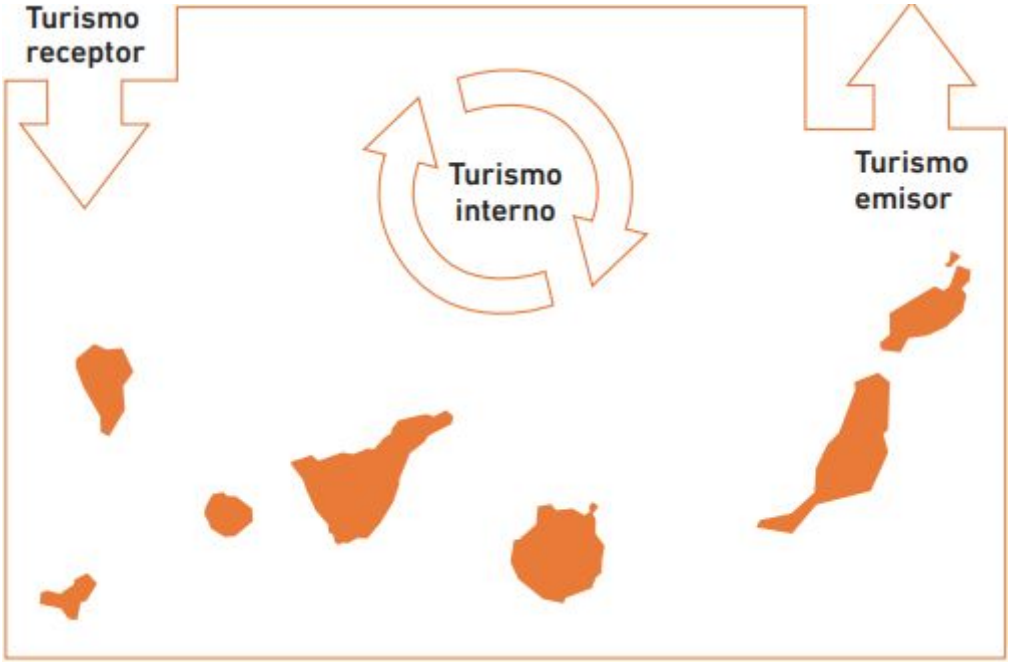


Figura 1.1: Concepto de turismo

Capítulo 2 Descripción del problema

2.1 Conceptos del problema

Para cada país de destino se definen tamaños mínimos de representación de la muestra combinada. La tabla siguiente indica los mínimos necesarios para representar un estrato.

Viajeros por estratos	Tamaño muestral mínimo
N < 60	20
De 60 a 999	80
De 1.000 a 9.999	200
De 10.000 a 24.999	400
De 25.000 a 39.999	500
De 40.000 a 59.000	600
60.000 y más	700

Figura 2.1: Tamaños mínimos de representación

Parte de las muestras pueden venir dadas parcialmente en el caso de que ya las haya hecho el INE se reduce el número mínimo de entrevistas a realizar y la cantidad de vuelos disponibles para entrevistar.

Se asume también que cada encuestador adapta sus tiempos de comida y descanso a las planificaciones que se le asigne dentro de su jornada y que van a estar disponibles todos los días de la encuesta.

Para obtener el número de asientos de un vuelo se necesita información adicional, ya que solo se especifica el tipo de avión y para conseguirla se ha acudido a Wikipedia para obtener los datos.

IATAcode	asientos	modelo
319	141	Airbus A319
320	180	Airbus A320-100/200
321	200	Airbus A321-100/200
32A	220	Airbus A323
32B	220	Airbus A323
32S	220	Airbus A318/319/320/321
332	288	Airbus A330-200
333	295	Airbus A330-300
73C	189	Boeing 737
73G	189	Boeing 737-700 pax
73H	189	Boeing 737-800 pax
737	189	Boeing 737
734	189	Boeing 737-400 pax
752	239	Boeing 757-200 pax
767	278	Boeing 767
76W	278	Boeing 767
AT7	72	Aerospatiale/Alenia ATR72
ATR	72	Aerospatiale/Alenia ATR42 / ATR72
BE4	10	Beechcraft
CRK	50	Canadair Regional Jet
DH4	68	De Havilland Canada DHC-8-400 Dash 8Q
E95	122	Embraer E-195
EP3	100	Embraer E-190
H28	100	British Aerospace (Hawker Siddeley)
M83	172	McDonnell Douglas MD83

Figura 2.2: Ficha de aviones

La resolución del problema de selección de vuelos a encuestar se realiza sin el conocimiento del número de pasajeros que viajarán en cada vuelo, que se estima como un porcentaje del número de asientos o ocupación, tampoco se conoce el número de pasajeros que van a aceptar ser entrevistados, por ello se crea otro factor llamado éxito.

Con los datos anteriores se puede generar el parámetro que va a ser usado en el modelo y viene dado de esta manera:

$$encuestas_i = num\_asientos_i \times ocupacion \times exito$$

Figura 2.3: Formula número de encuestas

La realización de cada entrevista consume tiempo al encuestador, por ello se ha añadido el parámetro de velocidad, que representa la cantidad de segundos necesarios para llevar a cabo una entrevista.

Y dados la velocidad y el número de encuestas viables de un vuelo se puede calcular la siguiente variable usada por el modelo:

$$consumo_i = encuestas_i \times velocidad$$

Figura 2.4: Consumo de encuestas

El tiempo de pasar de encuestar de un vuelo a otro también viene parametrizado como el tiempo de descanso entre vuelos, este valor se mide en minutos y es una restricción que hace viable la ruta generada por el modelo aunque este parámetro es el mismo entre todos los vuelos, una medida tomada por la ausencia de información.

2.2 Modelo matemático

El problema de este trabajo consiste en resolver un problema de programación lineal (también llamada optimización lineal), que trata de conseguir el mejor resultado

dadas unas restricciones y un objetivo ya sea maximizar o minimizar, en un modelo matemático estos requisitos están representados como relaciones lineales.

Los programas lineales se representan de manera canónica de la siguiente manera:

$$\mathbf{c}^{\mathrm{T}}\mathbf{x}$$
$$A\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq \mathbf{0}$$

Figura 2.5: Representación canónica de programas lineales

2.2.1 Variables del problema

Existen 4 variables en este modelo, todas ellas booleanas, es decir, que solo pueden tomar valor verdadero o falso y son las siguientes:

$$x_{ik} = \begin{cases} 1 & \text{si y sólo si el encuestador } k \text{ debe trabajar sobre } i \\ 0 & \text{en otro caso.} \end{cases} \quad (i \in I, k \in K)$$

$$y_i^1 = \begin{cases} 1 & \text{si y sólo si el vuelo } i \text{ a 1 único encuestador} \\ 0 & \text{en otro caso.} \end{cases} \quad (i \in I)$$

$$y_i^2 = \begin{cases} 1 & \text{si y sólo si el vuelo } i \text{ a 2 encuestadores} \\ 0 & \text{en otro caso.} \end{cases} \quad (i \in I)$$

$$z_p = \begin{cases} 1 & \text{si y sólo si logramos las encuesas mínimas del país } p \\ 0 & \text{en otro caso.} \end{cases} \quad (p \in P)$$

Figura 2.6: Variables del modelo

La variable ***z*** hace que el problema consiga obtener una solución siempre, aunque si llegase a ser imposible se recurre a registros donantes de encuestas anteriores.

Se pueden llegar a necesitar registros donantes en dos circunstancias:

- Un país sólo tiene 1 vuelo en esa semana, y sólo pueden obtenerse menos del tamaño muestral mínimo para dicho país.
- Algunos países (por determinar) tienen vuelos únicos que no se pueden entrevistar con los encuestadores disponibles.

2.2.2 Restricciones del problema

Dada una semana y un aeropuerto, el problema de selección de vuelos consiste en determinar qué vuelos debe realizar cada encuestador en cada día de la semana y en ese aeropuerto para obtener el máximo posible de encuestas.

Supongamos que los vuelos tienen una ocupación del 80% y el 50% de los entrevistados accede a participar, sabiendo que la velocidad de las encuestas duran 30 segundos cada una y el permutar entre vuelos consume 10 minutos.

En este caso veremos que par de vuelos pueden ser entrevistados por un mismo trabajador:

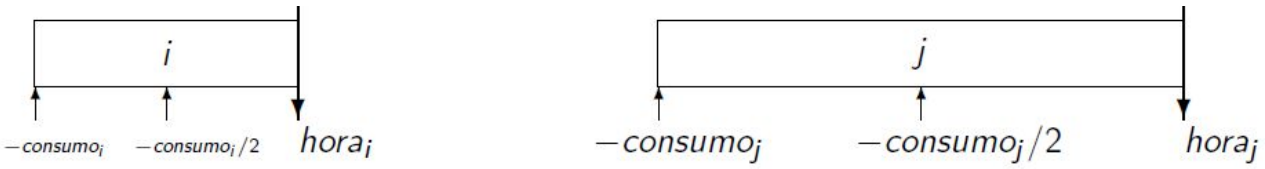


Figura 2.7: Ejemplo de horas de salida

Si los vuelos **i** y **j** están tuviesen horas de salida muy próximas o la diferencia de tiempo entre ellos es aproximadamente mayor a la jornada laboral del trabajador entonces no las hace el mismo encuestador:

$$hora_i > hora_j - consumo_j/2 - descanso \quad \text{o}$$
$$hora_i - consumo_i/2 - descanso + jornada < hora_j$$

Figura 2.8: Condiciones de restricción 1

Entonces:

$$x_{ik} + x_{jk} \leq 1$$

Figura 2.9: Restricción 1

En el caso en el que solo se quiera asignar un entrevistador por vuelo esta sería la única restricción aplicada para resolver el problema.

En el caso en el que se permita a los entrevistadores hacer las entrevistas juntos se aplicarán las siguientes restricciones:

Si el vuelo **j** lo hace 1 encuestador, puede hacer **i** si no son próximos a menos que **i** lo haga otro encuestador:

$$hora_i > hora_j - consumo_j - descanso$$

Figura 2.10: Condiciones de restricción 2

Entonces:

$$x_{ik} + x_{jk} \leq 1 + \sum_{l \in K \setminus \{k\}} x_{il}$$

Figura 2.11: Restricción 2

Si el vuelo **i** lo hace 1 encuestador, si no existe una diferencia de tiempo aproximadamente mayor a la jornada laboral salvo que el vuelo **j** lo haga otro entrevistador:

$$hora_i - consumo_i - descanso + jornada < hora_j$$

Figura 2.12: Condiciones de restricción 3

Entonces:

$$x_{ik} + x_{jk} \leq 1 + \sum_{l \in K \setminus \{k\}} x_{jl}$$

Figura 2.13: Restricción 3

Se debe añadir otra restricción para que un vuelo no sea entrevistado por una única persona y dos personas a la vez al mismo tiempo:

$$y_i^1 + y_i^2 \leq 1$$

Figura 2.14: Restricción 4

Y además se debe añadir otra restricción para que todos los entrevistadores de un vuelo entren a ser entrevistados por un solo entrevistador o por un grupo de dos entrevistadores:

$$\sum_{k \in K} x_{ik} = y_i^1 + 2 \cdot y_i^2$$

Figura 2.15: Restricción 5

La próxima restricción es la que consigue que el modelo pueda conseguir una solución siempre:

$$\sum_{i \in I_p} encuestas_i \cdot (y_i^1 + y_i^2) \geq (NumMin_p - INE_p) \cdot z_p$$

Figura 2.16: Restricción 6

La idea de esta restricción es que el número de encuestas realizadas sea superior al mínimo que se pide, pero en caso de no ser posible la variable **z** tomará como valor 0 y entonces siempre se podrá cumplir la restricción ya que las variables $\mathbf{y_1}$ e $\mathbf{y_2}$ son variables binarias y el número de encuestas es mayor o igual a 0.

La función objetivo de este modelo consiste en maximizar la cantidad de encuestas en la muestra valorando negativamente los países que queden sin su mínimo de encuestas dando doble valor a las encuestas de vuelos entrevistados por 2 encuestadores.

$$\sum_{i \in I} encuestas_i \cdot (y_i^1 + 2 \cdot y_i^2) \quad - \quad 100 \cdot \sum_{p \in P} NumMax_p \cdot (1 - z_p)$$

Figura 2.17: Función objetivo

Capítulo 3 Desarrollo de la aplicación

3.1 Estructura del programa

3.1.1 Entorno de la aplicación

Todo el desarrollo se hizo en un entorno virtual, esto permite la gestión de los paquetes instalados en proyectos distintos. El módulo venv es el que viene incluido en la librería estándar en versiones superiores a Python 3.3 y es el módulo recomendado para crear entornos virtuales con el comando **python -m venv env** siendo el segundo argumento la localización en la que crear el entorno virtual.

Tras haber terminado se genera un directorio con la siguiente estructura:

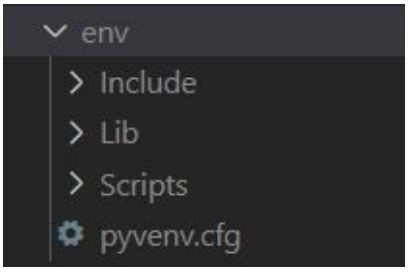


Figura 3.1: Entorno virtual

Los paquetes se guardan en la carpeta Lib, los scripts para activar y desactivar el entorno se encuentran en Scripts y la versión de python utilizada y demás variables se encuentran en **pyvenv.cfg**.

3.1.2 Estructura del directorio

- Las capturas hechas en esta explicación se han hecho desde la raíz del directorio del proyecto.

La estructura del paquete es la siguiente:

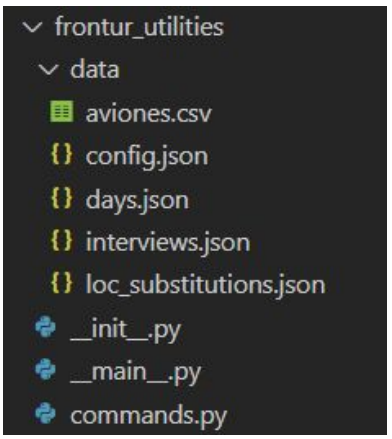


Figura 3.2: Estructura del paquete

Aquí se puede observar el enfoque usado para almacenar los archivos de configuración y demás información que no tiene porqué saber el usuario final, toda esta información estará incluida al descargar el paquete en pypi.

__init__.py es el punto de entrada desde el que se obtienen los métodos a exportar cuando se incluye un paquete desde un módulo y **__main__.py** es el punto de entrada cuando el módulo es llamado desde la consola con python.

La estructura de los test es la siguiente:

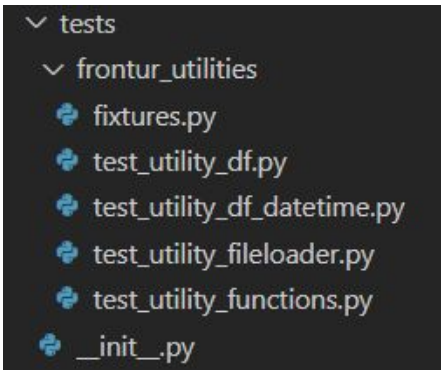


Figura 3.3: Directorio de tests

Es necesario que el directorio tenga el archivo **__init__.py** para poder ser usado con pytest, la herramienta escogida para realizar los tests.

3.1.3 Estructura de la información del programa

La información tratada en este proyecto viene a tratarse mayoritariamente de fechas, y en ello yace un problema que es el tratar con los formatos, ya que puede darse el caso de que la fecha sea la siguiente:

11/9/2020

Bajo interpretación podría decirse que esta fecha es el ***11 de septiembre***, pero interpretandolo de la manera en la que se hace de manera general en *Estados Unidos* la fecha sería el ***9 de noviembre***, que es el formato que Pandas usa por defecto, para paliar con esta problemática se decidió establecer por defecto el formato día/mes/año con la posibilidad de que el usuario pudiese cambiarlo en la configuración.

Teniendo este trasfondo se decidió adoptar este formato para indicar los días:

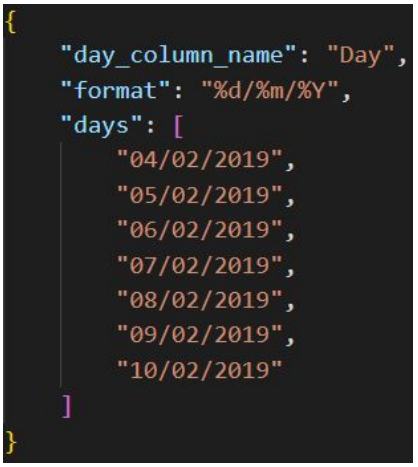


Figura 3.4: Fichero de días

Siendo el parámetro day_column_name la columna del Dataframe de la que se que van a escoger los días.

El fichero que recoge el número mínimo de entrevistas sigue el siguiente formato:

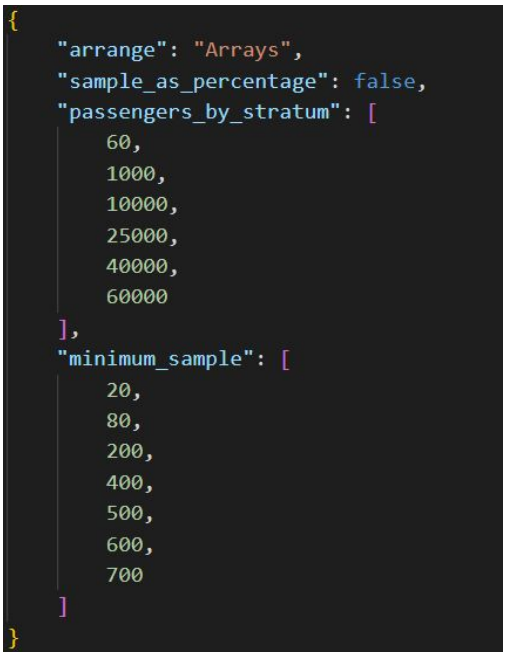


Figura 3.5: Archivo de entrevistas mínimas

El motivo de estar almacenados de esta manera es que el uso que se le va a dar a estos intervalos será la búsqueda normalmente de valores intermedios por ello se usará el algoritmo de bisección para buscar el índice en el que se encuentra el valor intermedio y luego acceder al valor de la muestra mínima con la posición obtenida.

El fichero que recoge todos los cambios de valores necesarios está hecho de manera en que se almacenen en un array y cada objeto/diccionario contenga el nombre de la columna con los valores que se van a buscar y el nombre de la columna en la que se va a reemplazar la información con el valor correspondiente:

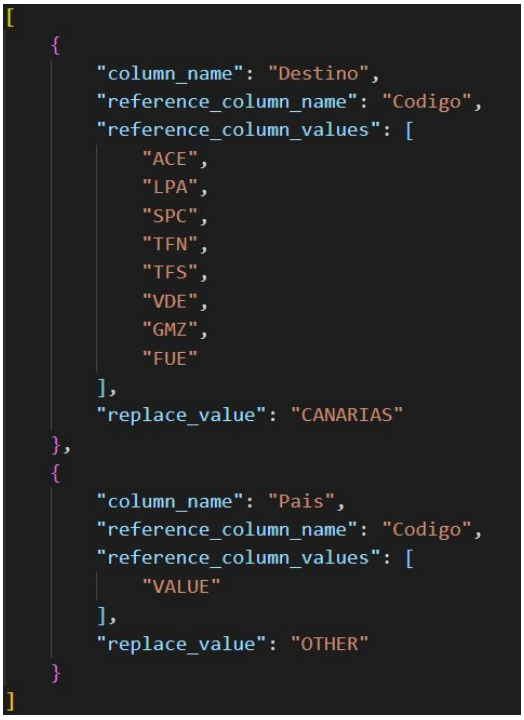


Figura 3.6: Archivo de sustituciones

Las demás configuraciones no requieren de ningún tratamiento específico ya que ambos archivos de configuración y de aviones consisten únicamente de entradas de valores de acceso común.

3.1.4 Documentación

La documentación fue generada con pdoc3 haciendo uso de docstrings en el formato de google, soportado por esta herramienta, con ello podemos generar información más legible, por ejemplo escribiendo el siguiente docstring:

```
"""Function that selects rows by a given dictionary
with the available days

Args:
    data_frame (pandas.DataFrame): Source DataFrame
    json_data (dict): Dictionary with the parameters required to do the map operation
    date_format (str, optional): Parameter key to the date format.
    available_days (str, optional): Parameter key to the available days.
    column_name (str, optional): Parameter key to the source column.
    alt_format (str, optional): Alternative date format.
    days (list, optional): List of dates that are going to be searched.

Returns:
    pandas.DataFrame: DataFrame with the selected dates
"""
```

Figura 3.7: Docstring

Los docstrings son atributos especiales que tienen los objetos como los módulos, métodos, funciones y clases almacenados en **__doc__** que contiene la documentación del objeto, esta información es procesada por herramienta generando una página de documentación con el comando **pdoc --html --output-dir build my_package**:

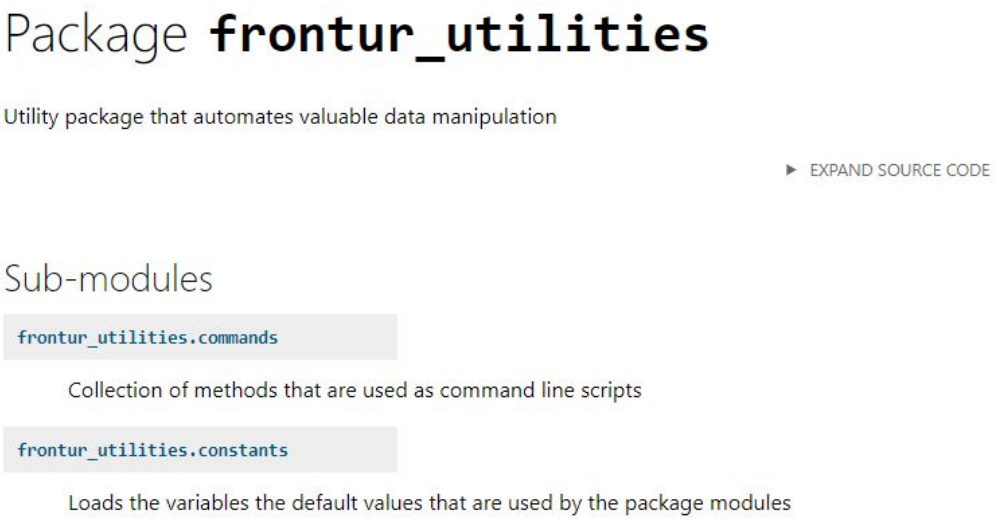


Figura 3.8: Página de documentación

3.2 Tratamiento de datos previo

3.2.1 Obtención de la información

El programa necesita recolectar información de varias fuentes.

I. En primer lugar necesita el archivo del GESLOT con la información referida

a los vuelos

II. Cada entrada en los vuelos contiene el modelo de avión, pero no el número de asientos que tiene, esta información viene dada por el archivo **aviones.csv**

III. Los días en los que se van a realizar las entrevistas

IV. El aeropuerto en el que se realizarán las entrevistas

V. Opcionalmente un fichero con sustituciones de valores

VI. Opcionalmente un fichero de salida

3.2.2 Construcción del DataFrame

En primera instancia el programa necesita obtener el fichero GESLOT en alguna de las extensiones soportadas en excel o csv, teniendo la ubicación del archivo se usa el método load_agenda del modulo **utility_fileloader.py**

```
if extension in const.SUPPORTED_EXCEL:
    data_frame = pandas.read_excel(file_path, date_parser=date_parser)
elif extension in const.SUPPORTED_CSV:
    try:
        data_frame = pandas.read_csv(file_path, date_parser=date_parser)
    except UnicodeDecodeError:
        guessed_encoding = guess_encoding(open(file_path, 'rb'))
        uf.eprint("Decoding error using: utf-8\nAutomatic guess:", guessed_encoding)
        data_frame = pandas.read_csv(file_path, date_parser=date_parser, encoding=guessed_encoding)
```

Figura 3.9: Apertura de archivo

Dentro de la función se decide a qué método llamar y en caso de que surja un error de decodificación se usa el módulo chardet para detectar la codificación del fichero pasándole como valor los bytes del fichero.

En el siguiente paso se selecciona el aeropuerto con el que se va a trabajar:

```
data_frame = data_frame.loc[lambda frame: frame[target_col] == airport]
if data_frame.empty:
    utility.eprint(f'Selected airport "{airport}" is invalid or not found.')
```

Figura 3.10: Selección de aeropuerto

En caso de haber indicado un archivo de sustituciones se realiza la operación, esto se consigue con el método apply del DataFrame que es el método que se usa para realizar operaciones sobre arrays de 1 dimensión y que recibe como parámetro la función que va a aplicar y el eje en el que lo va a realizar que en general es sobre las columnas:

```
for substitution_map in json_data:
    data_frame[substitution_map[json_column_name]] = data_frame.apply(
        lambda row: replace_value(
            row,
            substitution_map,
            json_column_name=json_column_name,
            json_reference_column_name=json_reference_column_name,
            json_reference_column_values=json_reference_column_values,
            json_replace_value=json_replace_value
            ),
        axis='columns')
```

Figura 3.11: Sustitución de valores

Usando el formato indicado anteriormente se itera sobre cada uno de los valores, en este caso diccionarios de python, y se le pasa como función una lambda con la llamada al método replace_value con los parámetros necesarios, este método luego devuelve el valor que corresponde:

```
if row[trad_dict[json_reference_column_name]] in trad_dict[json_reference_column_values]:
    return trad_dict[json_replace_value]
else:
    return row[trad_dict[json_column_name]]
```

Figura 3.12: Método de sustitución

Tras este paso se añade información de los aviones, este proceso requiere de un tratamiento adicional de la información ya que los tipos de aeronave suelen tener valores como AT7 y 73C existen códigos en la misma columna que tienen únicamente caracteres numéricos, por ejemplo el 320, y esto hace que su tipo de dato pase a ser entero, para ello se debe de imponer que todos los valores de la columna pasen a ser cadenas para continuar con el proceso y ya que sin esta condición se puede perder información por columnas que no casan.

Una vez modificados los tipos se aplica una unión externa de los valores, así se puede llegar a conocer si quedaron valores sin unir en el Dataframe objetivo y si este llegase a ser el caso producirse un mensaje de error advirtiendo al usuario.

```
planes = df_fileloader.load_agenda(file_path)
data_frame[target_col] = data_frame[target_col].astype(str)
planes[target_col] = planes[target_col].astype(str)
data_frame = pandas.merge(data_frame, planes, how='outer', on=[target_col], indicator=True)
unmatched = data_frame.query('_merge == "left_only"').groupby([target_col]).size().reset_index(name='count')
```

Figura 3.13: Añadido de información de aviones

Luego se transforma el Dataframe para que pase a tener intervalos de fechas con este formato:

Dia_semana	Opera_desde	Opera_hasta
D	04/11/2018	24/03/2019
M J	01/11/2018	28/03/2019
V	15/02/2019	29/03/2019

Figura 3.14: Intervalo de fechas

A tener días concretos, esto se consigue formateando la columna Dia_semana para que no tenga caracteres en blanco y luego se expanden los intervalos de las fechas, esto llevaría a tener una array de entradas por entrada del Dataframe aumentando así su dimensionalidad, por ello se tendría que volver a reducir la dimensión de la información contenida en el Dataframe:

```
data_frame[target_col] = data_frame.apply(lambda row: re.sub(r"\s+", '', row[target_col]), axis='columns')
dict_lists = data_frame.apply(lambda row: df_datetime.expand_date_intervals(row), axis='columns')
return pandas.pandas.DataFrame(utility.flatten(dict_lists))
```

Figura 3.14.1: Expansión de intervalos horarios

Y el último paso del proceso sería la selección de fechas:

```
days = frozenset(df_datetime.conv_to_datetime(json_data[available_days], alt_format))
return data_frame.loc[data_frame[json_data[column_name]].isin(days)]
```

Figura 3.15: Selección de días

El método **.loc** de los Dataframes usa etiquetas para leer y escribir datos, pero también puede contener aritmética para seleccionar las filas deseadas.

Tras haber terminado todo este proceso se termina con un Dataframe con la información necesaria, en la siguiente figura se muestra un Dataframe con información válida, pero cabe tener en cuenta que podría llegar a componerse de valores adicionales:

	Unnamed: 0	Unnamed: 1	Destino	Codigo	Hora_Salida	Aeronave	Num_vuelo	Pais	Escala	Origen	Day	asientos	modelo
0	W18	Destino	AMSTERDAM/SCHIPHOL	AMS	10:10:00	73H	TRA5686	CANARIAS	NaN	ACE	10/02/2019	189	Boeing 737-800 pax
1	W18	Destino	AMSTERDAM/SCHIPHOL	AMS	10:45:00	73H	TRA5686	CANARIAS	NaN	ACE	07/02/2019	189	Boeing 737-800 pax
2	W18	Destino	AMSTERDAM/SCHIPHOL	AMS	10:45:00	73H	TRA5686	CANARIAS	NaN	ACE	05/02/2019	189	Boeing 737-800 pax
3	W18	Destino	AMSTERDAM/SCHIPHOL	AMS	19:35:00	73H	TRA5684	CANARIAS	NaN	ACE	09/02/2019	189	Boeing 737-800 pax
4	W18	Destino	BELFAST / INTERNACIONAL	BFS	07:05:00	73H	RYR3498	SEGUNDO	NaN	ACE	08/02/2019	189	Boeing 737-800 pax
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2795	W18	Destino	GOTEBORG /LANDVETTER	GOT	12:45:00	76W	TOM3645	SUECIA	NaN	LPA	09/02/2019	278	Boeing 767 ??
2796	W18	Destino	GOTEBORG /LANDVETTER	GOT	14:30:00	76W	TOM3313	SUECIA	NaN	LPA	06/02/2019	278	Boeing 767 ??
2797	W18	Destino	ESTOCOLMO/ARLANDA	ARN	13:30:00	76W	TOM3747	SUECIA	NaN	TFS	10/02/2019	278	Boeing 767 ??
2798	W18	Destino	GOTEBORG /LANDVETTER	GOT	13:05:00	76W	TOM3785	SUECIA	NaN	TFS	10/02/2019	278	Boeing 767 ??
2799	W18	Destino	OSLO / GARDERMOEN	OSL	12:40:00	32Q	NVR158	NORUEGA	BLL	ACE	08/02/2019	220	Airbus ??

[2800 rows x 13 columns]

Figura 3.16: Dataframe formateado

Los campos que va a usar el modelo de optimización del programa son las horas de salida del vuelo, el número de pasajeros del vuelo y el día que va a salir.

3.3 Implementación del modelo de optimización

3.3.1 Estructura de la información

Para almacenar la información se ha hecho uso de las dataclasses de python que vienen a ser clases regulares de python enfocadas a almacenar información, y no tanto a la lógica. Estas se encargan de interpretar la información dada por el Dataframe usado para instanciarlas, éstas recogen vistas del Dataframe, generan información y guardan parámetros necesarios para el modelo.

Se han usado 3 dataclasses para albergar la información:

- **SolverParameters**, que toma el Dataframe generado en el punto anterior y se encarga de generar las instancias de cada país y de cada entrevistador. Además guarda los parámetros tiempo límite de ejecución del programa, número de horas de la jornada laboral y tiempo de espera entre vuelos
- **Country**, esta dataclass recibe como parámetro la vista correspondiente a los vuelos pertenecientes de un país, el índice al que corresponde, el nombre del país y el número de viajantes con destino a ese país. Además se encarga de generar el número mínimo de entrevistas.
- **Flight**, que recibe una única entrada por vista de ahí obtiene la información sobre el número del vuelo, el número de asientos que tiene, el día y la hora de salida, el número de entrevistas que se pueden efectuar y el tiempo que consume hacerlas.

El motivo por el cual se usa este enfoque es por la flexibilidad que otorga ante futuros cambios y la conveniencia de ser una clase dedicada a almacenar el estado.

3.3.2 Implementación del modelo

Para la implementación se hizo uso de or-tools aprovechado también la expresividad de python y sus comprensiones de listas, generalmente se usan para instanciar arrays, pero en este caso se usarán para añadir restricciones al modelo.

Para empezar a añadir restricciones al modelo se necesita instanciar un solver:

```
solver = pywraplp.Solver('FronturSolver', pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING)
```

Figura 3.17: Instanciación del solver

OR-Tools provee de varias interfaces para resolver problemas de programación de enteros mixtos. El solver CBC(Coin-or-branch and cut) viene incluido en OR-Tools, aunque también se pueden instalar solvers de terceros si compilas los ficheros fuente desde el propio ordenador para usar SCIP, GLPK o Gurobi.

Dependiendo del caso el solver puede llegar a tardar bastante tiempo en dar una solución, por ello se añade un limite de tiempo para encontrar la solución.

```
solver.SetTimeLimit(int(sol_par.execution_time_limit) * 1000)
```

Figura 3.18: Límite de ejecución

Se inicializan las variables del problema:

```
x = { (i,k) : solver.BoolVar('x[%i, %i]' % (i, k)) for i in range(sol_par.num_flights) for k in pollsters }
y1 = { (i) : solver.BoolVar('y1[%i]' % (i)) for i in range(sol_par.num_flights) }
y2 = { (i) : solver.BoolVar('y2[%i]' % (i)) for i in range(sol_par.num_flights) }
z = { (p) : solver.BoolVar('z[%i]' % (p)) for p in range(sol_par.num_countries) }
```

Figura 3.19: Instanciación de variables

El número de instancias de cada variable depende del número de entrevistadores, vuelos y países.

Se añade la restricción para que un vuelo no sea entrevistado por una única persona y dos personas a la vez al mismo tiempo:

```
[solver.Add(y1[i.index] + y2[i.index] <= 1) for i in sol_par.flights]
```

Figura 3.20: Adición restricción 1

La restricción para que el número mínimo de entrevistas se lleve a cabo:

```
[solver.Add(
    sum(i.surveys * (y1[i.index] + y2[i.index]) for i in p.flights)
    >= (p.min_polls - p.ine_polls) * z[p.index]
    ) for p in sol_par.countries]
```

Figura 3.21: Adición restricción 2

La restricción para que un mismo entrevistador no pueda entrevistar dos vuelos muy próximos entre sí:

```
if (i.flight_hour > j.flight_hour - j.time_consumed/2 - sol_par.rest_time) \
    or (i.flight_hour - i.time_consumed/2 < j.flight_hour - sol_par.workday_time):
        [ solver.Add( x[i.index, k] + x[j.index, k] <= 1 ) for k in pollsters]
```

Figura 3.22: Adición restricción 3

Estas son las restricciones que hacen posible hacer entrevistas a vuelos próximos si hay otro entrevistador que las haga:

```
elif i.flight_hour > j.flight_hour - j.time_consumed - sol_par.rest_time:
    [ solver.Add(
        x[i.index, k] + x[j.index, k]
        <= 1 + sum(x[i.index, k_not] for k_not in pollsters if k_not != k)
    ) for k in pollsters ]
```

Figura 3.23: Adición restricción 4

```
elif i.flight_hour - i.time_consumed < j.flight_hour + sol_par.workday_time:
    [ solver.Add(
        x[i.index, k] + x[j.index, k]
        <= 1 + sum(x[j.index, k_not] for k_not in pollsters if k_not != k)
        ) for k in pollsters ]
```

Figura 3.24: Adición restricción 5

La función a maximizar con la penalización negativa por no entrevistar el mínimo de países:

```
solver.Maximize(
    sum(f.surveys*(y1[f.index] + y2[f.index]*2) for f in sol_par.flights)
    - 100*sum(p.num_travelers*(1-z[p.index]) for p in sol_par.countries)
    )
```

Figura 3.25: Adición función objetivo

3.4 Testeo del proyecto

El testeo del proyecto ha usado el framework pytest debido a su facilidad para crear pequeños tests con aserciones detalladas y la posibilidad de poder incluir las funcionalidades del módulo unittest desde el primer momento, necesario para hacer mocks.

3.4.1 Pytest fixtures

Los fixtures permiten a las funciones de test recibir fácilmente y trabajar con objetos específicos pre-inicializados sin tener que encargarse de los detalles de importar, configurar y liberar los recursos, estos pueden ser accedidos por las funciones de testeo a través de parámetros.

En la suite de tests del programa se declararon 2 fixtures que son usados en la mayor parte de los test cases en **fixtures.py**, en ellos se declara el Dataframe que van a compartir todos los test y el diccionario de parámetros que usan los métodos, para declararlos solo se necesita el decorador **@pytest.fixture**.

```
@pytest.fixture
def json_data():
    return {
        # Values used to substitute rows
```

Figura 3.26: Creación fixture

Y para usarlos solo es necesario importarlos:

```
from fixtures import df, json_data

def test_replace_value(df, json_data):
    # Assertions use Dataframe instances of only one row
```

Figura 3.27: Uso fixture

3.4.2 Mocks

En algunos casos los test llaman a un método que manipula archivos usando las librerías de Pandas, algo que no es necesario testear y que ralentiza y consume los recursos, la solución a esto es hacer uso de mocks, objetos pre-programados con especificaciones sobre las llamadas y valores de retorno.

En este caso los usaremos para que la llamada a la librería de Pandas solo devuelva un valor específico acelerando así los tests.

```
@mock.patch(
    'frontur_utilities.utility_fileloader.pandas.read_csv',
    return_value='csv',
    autospec=True)
def test_load_csv_agenda(mock_method, ext, tmpdir):
    file = tmpdir.join('filename.' + ext)
    actual = utility_fileloader.load_agenda(file.strpath)
```

Figura 3.28: Mocking

En el ejemplo se pueden observar varios elementos, primero el mock, que recibe tres parámetros, el ámbito del módulo desde el que se llama al método objetivo, el valor que retorna el método y el parámetro autospec que asegurar que los parámetros que llaman a la función son correctos. Pero además hace uso del parámetro tmpdir que es una fixture incluida en pytest que provee de un directorio temporal único para el test que lo invoca.

3.4.3 Parametrización

Otra característica incluida en pytest es el decorador **@pytest.mark.parametrize** que permite la parametrización de los argumentos de las funciones de los tests.

```
@pytest.mark.parametrize('expected, value', [
    ([0], 'L'),
    ([1], 'M'),
    ([2], 'X'),
    ([3], 'J'),
    ([4], 'V'),
    ([5], 'S'),
    ([6], 'D'),
    ([0, 1, 2, 3, 4], 'LMXJV'),
    ([2, 4], 'XV'),
    ([0, 3, 4, 5, 6], 'LSDJV')
])
def test_parse_workdays(expected, value):
    assert expected == df_datetime.parse_workdays(value)
```

Figura 3.29: Parametrización

Esto hace que el código sea más limpio y simple de entender eliminando código duplicado de la prueba

3.4.4 Cobertura de los tests

La cobertura conseguida por los tests desarrollados cumplen con las funcionalidades requeridas para el funcionamiento correcto del programa:

Coverage report: 70%

Module ↑	statements	missing	excluded	coverage
frontur_utilities__init__.py	7	0	0	100%
frontur_utilities__main__.py	4	4	0	0%
frontur_utilities\commands.py	33	15	0	55%
frontur_utilities\constants.py	16	0	0	100%
frontur_utilities\extract_methods.py	37	24	0	35%
frontur_utilities\utility_df.py	22	4	0	82%
frontur_utilities\utility_df_datetime.py	42	0	0	100%
frontur_utilities\utility_fileloader.py	38	13	0	66%
frontur_utilities\utility_functions.py	13	4	0	69%
Total	**212**	**64**	**0**	**70%**

coverage.py v5.2.1, created at 2020-08-31 06:40 +0100

Figura 3.30: Cobertura

3.5 Desarrollo con Github

Github es una plataforma para el desarrollo de software y versión de controles usando Git, Git es una herramienta para gestionar los cambios del código en el proceso de desarrollo de software y Github junto con este servicio provee varias características, entre ellas la gestión de tareas que ha sido una herramienta utilizada durante el desarrollo.

3.5.1 Configuración previa del repositorio

El repositorio tiene información privada guardada en los secretos del propio repositorio siendo el usuario y contraseña de pypi y el token de acceso personal (PAT) de github para poder accionar eventos de flujo de trabajo

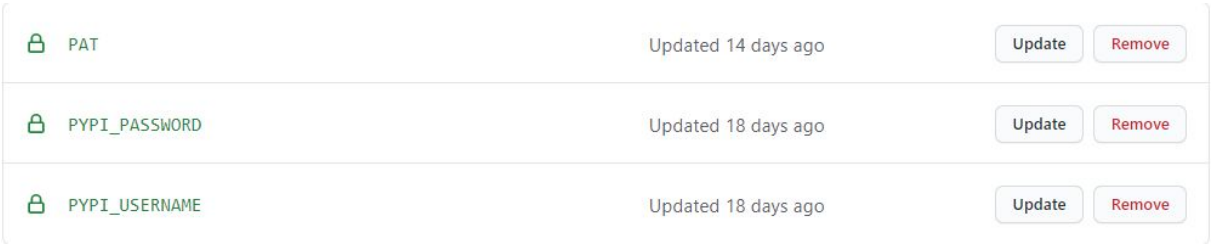


Figura 3.31: Github secrets

Los tokens de acceso personal(PAT) son una alternativa al método de identificación de usuario y contraseña cuando se usa la api de Github o la línea de comandos en la que se pueden establecer los permisos del token, en este caso se usarán en los workflows ya que el token por defecto de github no acciona otros workflows.

3.5.2 Automatización con workflows

Github dispone de flujos de trabajo, estos son procesos automatizados personalizables para la automatización del ciclo de vida del desarrollo del proyecto definidos bajo el directorio **.github/workflows**.

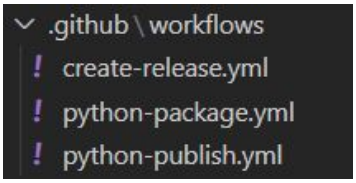


Figura 3.32: Github workflows

Estas tres tareas llevan a cabo las siguientes labores:

- **Testeo multiplataforma** - Accionado tras subir los archivos con modificaciones en los archivos fuente de python creando una serie de trabajos que corren en las últimas versiones de ubuntu, mac os y windows con las versiones 3.7 y 3.8 de python
- **Publicación de nuevos lanzamientos** - Accionado tras añadir tags a las versiones del proyecto, Git tiene la habilidad de etiquetar puntos específicos en el historial del repositorio como punto de interés con **git tag v1.0**, tras haber marcado los puntos de interés estos se deben subir aparte de los habituales push con **git push origin --tags** que creará el lanzamiento correspondiente a la versión del tag identificado con el token de acceso del administrador del repositorio(PAT) para accionar los flujos de trabajo dependientes de este
- **Publicación de la distribución en pypi** - Dependiente de que se realice un lanzamiento nuevo del proyecto generando la distribución del proyecto con setuptools y subiendo la distribución a pypi con la identificación de usuario establecida en el entorno del sistema

3.6 Distribución de la aplicación en pypi

3.6.1 Creación de la distribución

Un módulo/paquete de python generalmente debe de seguir las siguientes restricciones:

- Todo en minuscula
- Debe de ser único en pypi, incluso si quieres que el paquete no sea público
- Las separaciones del nombre deben de ser '_'

En el caso de este proyecto todos los paquetes siguen la convención de **frontur_***.

Es importante que cada paquete tenga asociada una licencia, la licencia que usa el proyecto es la MIT, una licencia simple y permisiva, deja que otros usuarios puedan hacer casi cualquier cosa con el proyecto, como crear y distribuir versiones propias de código cerrado.

En la creación de la distribución se establecieron las mismas dependencias a las usadas en el entorno usando el comando ***pip freeze > requirements.txt.***

```
chardet==3.0.4
click==7.1.2
numpy==1.19.1
pandas==1.1.0
python-dateutil==2.8.1
pytz==2020.1
six==1.15.0
```

Figura 3.33: Semantic versioning

Los paquetes de distribución fueron generados con las últimas versiones de setuptools y wheel usando los flujos de trabajo de github los cuales generan la carpeta dist con dos archivos:

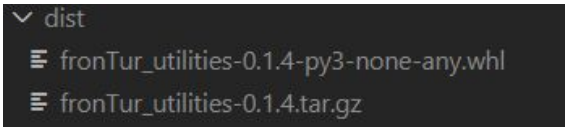


Figura 3.34: Carpeta de distribución

- **.tar.gz** - que contiene los archivos fuente del lanzamiento generado con el parámetro *'sdist'*(*Source Distribution*) que recibe ***setup.py***
- **.whl** - Un formato de distribución que contiene ficheros y metadatos que solo necesitan ser movidos a la localización del sistema correcta para ser instalado generado con el parámetro *'bdist_wheel'*(*Build Distribution*)

El motivo por el que se generan dos archivos es porque las nuevas versiones de pip prefieren descargar archivos **.whl**, pero retrocederá a la descarga de archivos fuente si es necesario.

Este directorio no está incluido en el repositorio, se genera dentro del flujo de trabajo y luego se sube a pypi.

3.6.2 Método de guardado de configuración

Entre los métodos convencionales existe la filosofía de crear un directorio oculto en la carpeta de usuario o en la carpeta root, en este caso se ha optado por guardar la configuración dentro del mismo paquete, esto es posible gracias al archivo MANIFEST.in

3.6.3 Comandos

El paquete tiene implementados comandos implementados con click (Command Line Creation Interface), un paquete que contiene decoradores de funciones de python que abstrae facetas de la implementación de comandos independientes de la plataforma.

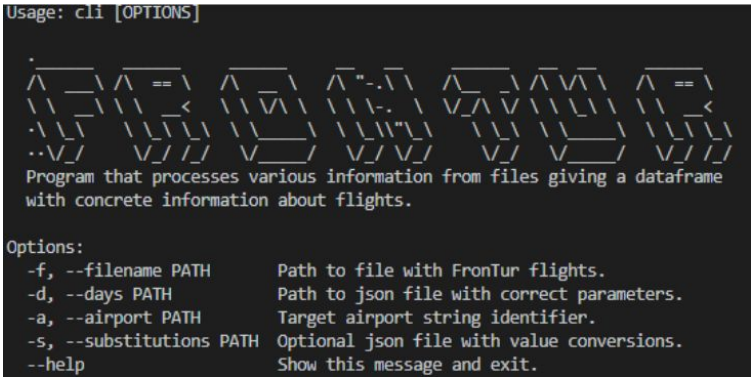


Figura 3.35: Mensaje de ayuda comando

Los comandos se encuentran en el archivo **commands.py** y funciona de manera que si se quiere añadir un nuevo comando solo se tiene que añadir al grupo de comandos cli:

```python
@click.group()
def cli():
    """Method that can be called my other modules to use the commands on this file"""
    pass
```

Figura 3.36: Grupo de comandos

Dichos comandos han sido añadidos como puntos de entrada desde **setup.py**

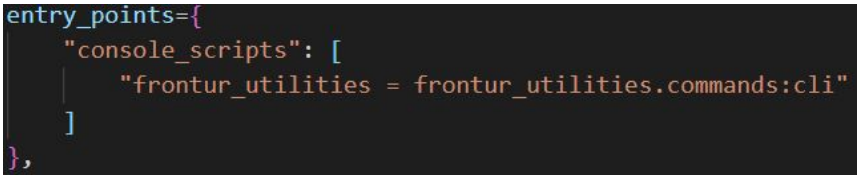


Figura 3.37: Punto de entrada

Los comandos disponibles son los dos necesarios para generar la información necesaria y el modelo que usa la información generada para resolver el problema, además se añadió un comando para poder editar el archivo de configuración en caso de ser necesario.

Capítulo 4 Desarrollo de la interfaz gráfica

4.1 Diseño de la arquitectura

La arquitectura del proyecto sigue el patrón de Modelo Vista Controlador, Kivy tiene un lenguaje propio con el que se pueden implementar las vistas para facilitar la separación con la lógica del modelo.

La estructura del directorio es la siguiente:

Figura 4.1: Carpeta controlador

Existe un controlador por fase de programa.

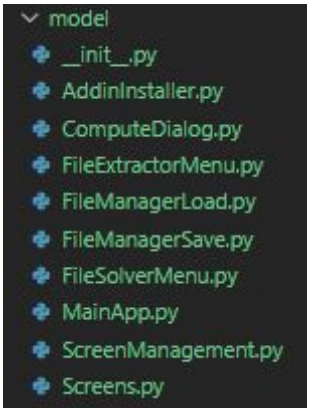


Figura 4.2: Carpeta modelo

En estos modelos se alberga la lógica de las vistas

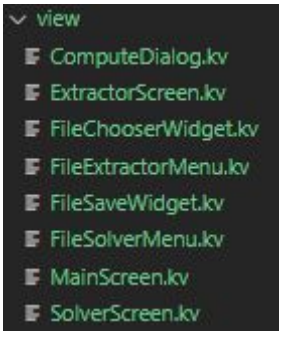


Figura 4.3: Carpeta vista

Las vistas escritas en el lenguaje de Kivy.

4.1.1 Asincronía en la aplicación

Ya que la aplicación en ocasiones requiere de un tiempo de cómputo intensivo dependiendo de la entrada y el estado en el que se encuentra el proceso se muestra por pantalla está diseñado para ser dinámico, el hecho que solo haya un hilo de ejecución en el programa hace que la interfaz no funcione de manera adecuada, por ello se requiere de un wrapper para hacer la llamada a los métodos en un hilo aparte, la solución es la creación de un componente que muestra el estado de ejecución en una pantalla.

La solución fue crear un componente al que se le asignará cualquier método con un yield que devolviese una cadena por cada paso de ejecución, es decir, funciones generadoras de python.

```
class ComputeDialog(FloatLayout):
    text_output = ObjectProperty(None)
    cancel = ObjectProperty(None)
    loading_method = ObjectProperty(None)
    callback = ObjectProperty(None)
    thread = ObjectProperty(None)

    def compute(self, instance):
        self.thread = threading.Thread(target=self.method)
        self.thread.start()

    def method(self):
        from datetime import datetime
        for step in self.loading_method():
            self.container.text += f'[{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}] ' + step + '\n'
        self.continue_btn.disabled = False
        self.callback()
```

Figura 4.4: Implementación asincronía interfaz

4.2 Implementación de la interfaz

Para la parte de la interfaz se usó Kivy, una librería de código abierto escrita en python para un desarrollo rápido de aplicaciones que pueden ser ejecutadas en Linux, Windows, OS X, Android e IOS. Hace que el mismo código sea ejecutado en todas las plataformas soportadas, de uso libre bajo la licencia de MIT.

4.2.1 Elementos de la aplicación

En primer lugar están las Screens o pantallas de Kivy que sobre ellas se hacen las transiciones:

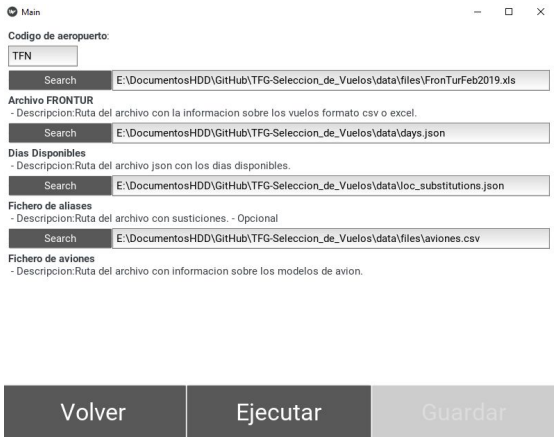


Figura 4.5: Pantalla de aplicación

Estas albergan los menús de cada aplicación:

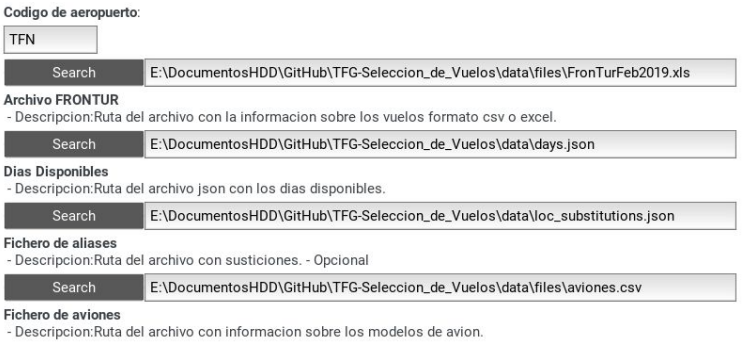


Figura 4.6: Menú de aplicación

Los menús contienen los componentes customizados que son usados repetidas veces en otras vistas como FileChooserWidget que sirve para seleccionar archivos:

Figura 4.7: Widget aplicación

Se compone de un marco de texto no editable y un botón de búsqueda que al pulsarlo aparece popup para seleccionar el archivo:

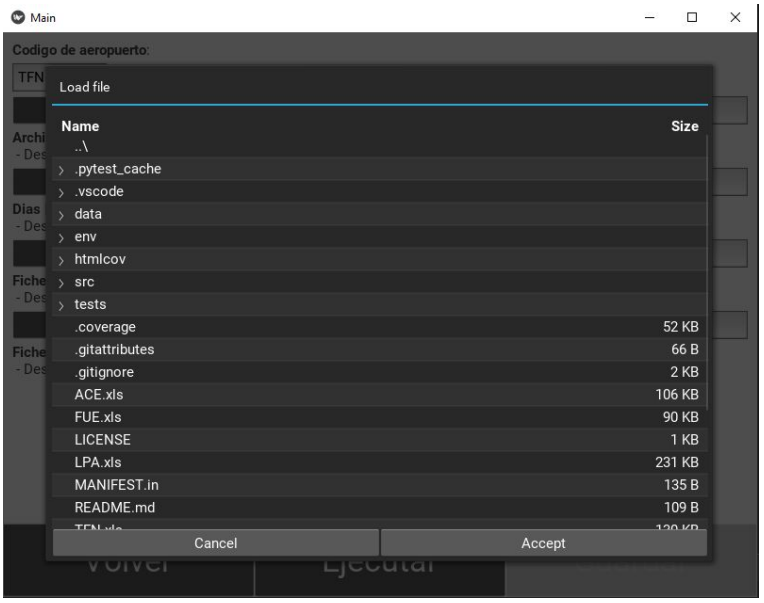


Figura 4.8: Explorador de archivos

Tras seleccionar el archivo el marco de texto se actualiza con la dirección absoluta del fichero.

Una vez seleccionados los archivos se produce la operación correspondiente a la pantalla en la que esté:

Figura 4.9: Opciones pre-ejecución

La opción de guardado no estará disponible hasta que haya ejecutado satisfactoriamente el resultado, el progreso de la ejecución se mostrará en un popup con mensajes que muestran el proceso de ejecución.

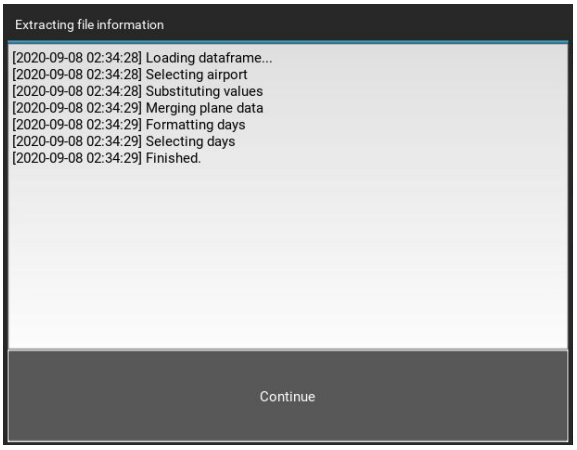


Figura 4.10: Diálogo de computo

Una vez completado la opción de guardado estará disponible:

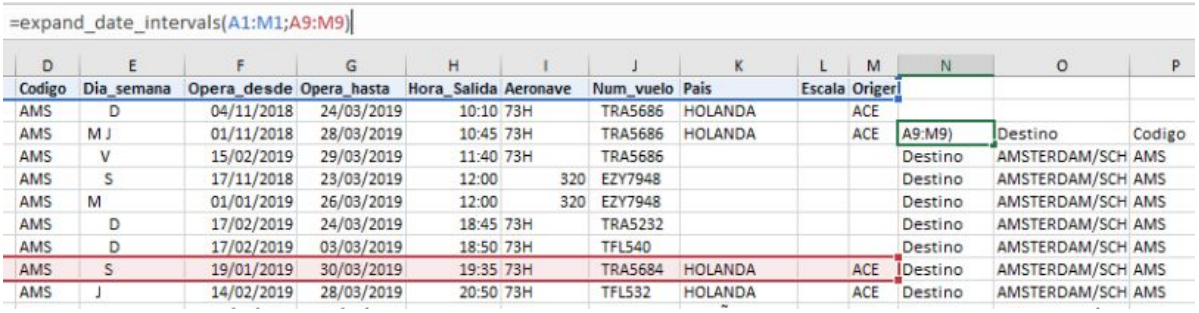
Figura 4.11: Opciones post-ejecución

Con esta opción se puede guardar el resultado de la operación en formato excel o csv en el disco.

Capítulo 5 Desarrollo del add-in de Excel

5.1 Desarrollo del addin

La implementación del add in fue hecha usando xlwings, una librería de python que provee una api para definir nuestros UDF's (User Defined Functions), usa una licencia BSD de 3 cláusulas que no impide el desarrollo previsto de las funcionalidades. xlwings soporta dependencias opcionales que recomienda encarecidamente, entre ellas Pandas, esto hace que se integre de manera natural en el proyecto.



	D	E	F	G	H	I	J	K	L	M	N	O	P
	Codigo	Dia_semana	Opera_desde	Opera_hasta	Hora_Salida	Aeronave	Num_vuelo	Pais	Escala	Origen			
	AMS	D	04/11/2018	24/03/2019	10:10	73H	TRA5686	HOLANDA		ACE			
	AMS	M J	01/11/2018	28/03/2019	10:45	73H	TRA5686	HOLANDA		ACE	A9:M9)	Destino	Codigo
	AMS	V	15/02/2019	29/03/2019	11:40	73H	TRA5686				Destino	AMSTERDAM/SCH	AMS
	AMS	S	17/11/2018	23/03/2019	12:00	320	EZY7948				Destino	AMSTERDAM/SCH	AMS
	AMS	M	01/01/2019	26/03/2019	12:00	320	EZY7948				Destino	AMSTERDAM/SCH	AMS
	AMS	D	17/02/2019	24/03/2019	18:45	73H	TRA5232				Destino	AMSTERDAM/SCH	AMS
	AMS	D	17/02/2019	03/03/2019	18:50	73H	TFL540				Destino	AMSTERDAM/SCH	AMS
	AMS	S	19/01/2019	30/03/2019	19:35	73H	TRA5684	HOLANDA		ACE	Destino	AMSTERDAM/SCH	AMS
	AMS	J	14/02/2019	28/03/2019	20:50	73H	TFL532	HOLANDA		ACE	Destino	AMSTERDAM/SCH	AMS

Figura 5.1: Hoja Excel

La intención del desarrollo del addin fue la de dar la posibilidad a las personas que se dediquen a usar Excel de poder usar las funcionalidades desarrolladas en este proyecto.

Las funciones implementadas para excel están implementadas en **__init__.py** ya que cualquier método que no esté definido en el punto de entrada(que sea importado desde otro módulo) no funcionará de manera correcta.

Para definir una función del addin se necesita poner el decorador **@xw.func** en primer lugar, luego para definir los parámetros **@xw.arg** con el nombre del parámetro y los parámetros para especificar el tipo de dato que sería:

```
@xw.func
@xw.arg('data_frame', pd.DataFrame, header=True, index=False, dates=datetime.datetime)
@xw.ret(index=False, expand='table')
def expand_date_intervals(data_frame):
```

Figura 5.2: Decoradores xlwings

En este caso la función recibiría un Dataframe como argumento, pero Excel utiliza un formato propio para el tiempo, este da un valor entero que comienza a partir del 1 de enero del año 1900 y para que sea compatible con los métodos implementados se tienen que decodificar:

```
def decode_date(value):
    if isinstance(value, (float, int)):
        return datetime.date(1900,1,1) + datetime.timedelta(days=int(value))
    elif isinstance(value, str):
        from dateutil import parser
        return parser.parse(value)
```

Figura 5.3: Método de decodificación

Además de decodificarlo también tiene que codificarlo de la manera apropiada para que pueda ser mostrado en la hoja de trabajo:

```
if 'Hora_Salida' in data_frame:
    data_frame['Hora_Salida'] = data_frame.apply(lambda row:
        str(row['Hora_Salida']).split(' ')[-1], axis='columns')
if 'Day' in data_frame:
    data_frame['Day'] = data_frame.apply(lambda row:
        row['Day'].strftime('%d/%m/%Y'), axis='columns')
```

Figura 5.4: Método de codificación

5.2 Asincronía en la aplicación

Xlwings soporta desde la versión 0.14.0 funciones asíncronas, al activar esta funcionalidad asignando el parámetro async_mode a 'threading' permite mientras se realiza el cómputo de la llamada a la función se pueda seguir usando Excel mientras espera a que la función termine de retornar el valor, tras terminar la celda o celdas se actualizan automáticamente.

```
@xw.func(async_mode='threading')
@xw.arg('data_frame', pd.DataFrame, header=True, index=False, dates=datetime.datetime)
@xw.ret(index=False, expand='table')
```

Figura 5.5: xlwings async decorator

Esta funcionalidad no cambia la manera de llamarla desde Excel ni requiere tratar con los valores recibidos de manera distinta.

Capítulo 6 Uso de las herramientas

El proyecto consiste de 3 herramientas en total como se ha visto hasta ahora, en este capítulo se mostrará en donde se encuentran, como son descargadas y como se hace uso de ellas.

6.1 frontur_utilities

El paquete frontur_utilities se encuentra hospedado en la plataforma TestPypi una instancia separada del Python Package Index que permite probar las herramientas de distribución sin afectar al Index principal.

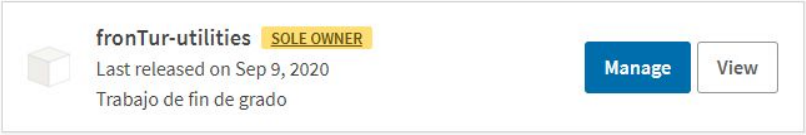


Figura 6.1: frontur_utilities en pypi

De igual manera también se puede encontrar el código en Github accesible de manera pública.

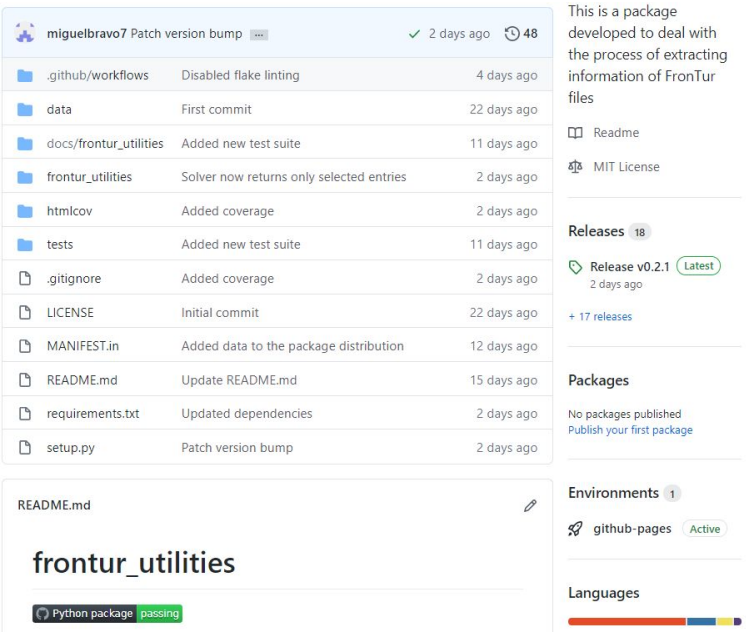


Figura 6.2: frontur_utilities en Github

Para poder instalarlo se necesita pip y python 3.7 o superior de 64 bits, cumpliendo los requisitos anteriores se procede a instalar con:

pip install -i https://test.pypi.org/simple frontur_utilities

Tras haberlo instalado aparte de poder importar sus métodos en otros módulos también ofrece un comando de consola que ofrece tres posibilidades:

process el comando que hace el tratamiento de la información para generar los

valores que usa el solver, las únicas opciones necesarias son **infile** que toma como argumento la ruta del archivo con la información sobre Frontur y **airport** que es la cadena de texto que corresponde al código de aeropuerto deseado, los demás parámetros son generados de manera automática.

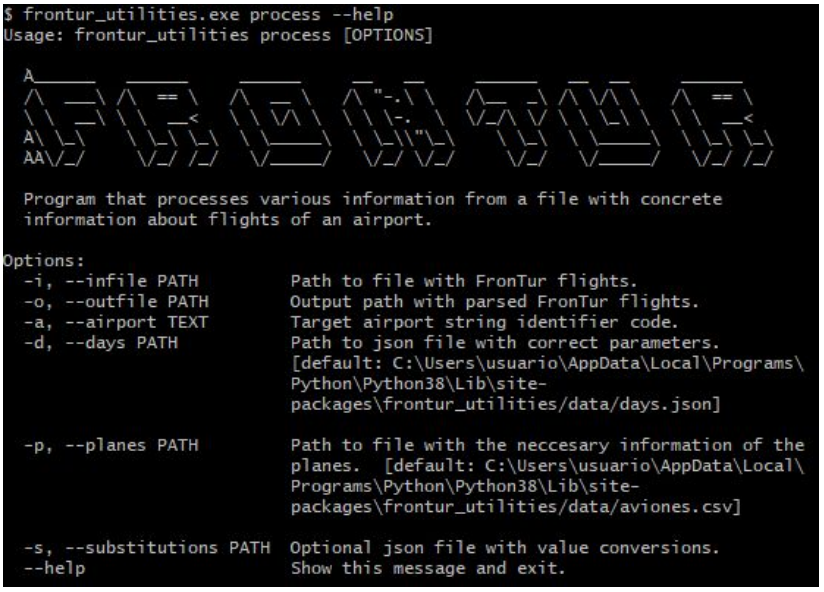


Figura 6.3: Comando process

solver comando que recibe el archivo ya sea csv o excel con los valores necesarios usando la opción **infile**.

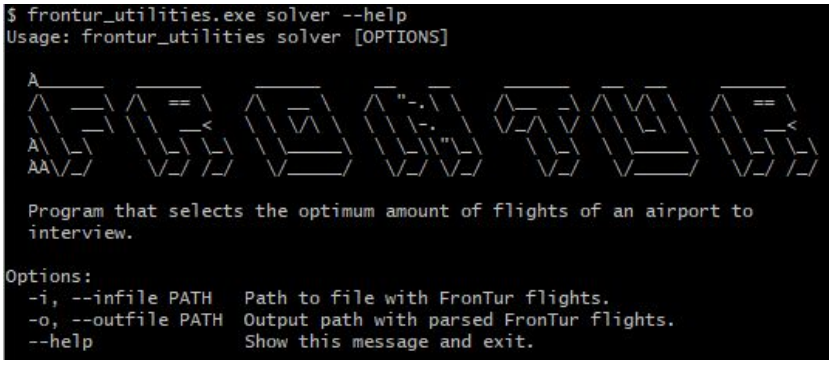


Figura 6.4: Comando solver

edit_conf que abre el archivo de configuración del paquete con el editor de texto predeterminado u opcionalmente se puede especificar por opciones el editor con el que abrirlo.

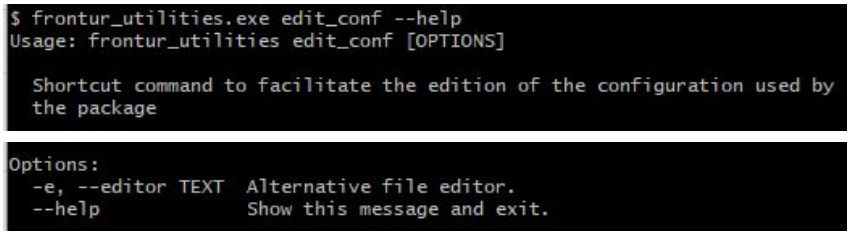


Figura 6.5: Comando edit_conf

6.2 frontur_excel_addin

El paquete frontur_excel_addin también se encuentra en TestPypi:

Figura 6.6: frontur_excel_addin en pypi

Pero en este caso hace uso del hecho de ser un paquete instalado en el sistema por pypi con otro propósito, ya que el método de importado de un módulo en python empieza buscando en primer lugar en la ruta de módulos instalados, una vez instalado frontur_excel_addin podemos importarlo desde cualquier ruta dentro de la máquina.

Para poder usar el addin se debe de instalar usando el comando:

xlwings addin install

Tras ello aparecerá la siguiente pantalla:

Figura 6.7: Comando de instalación addin xlwings

Tras ello aparecerá una nueva pestaña llamada xlwings:

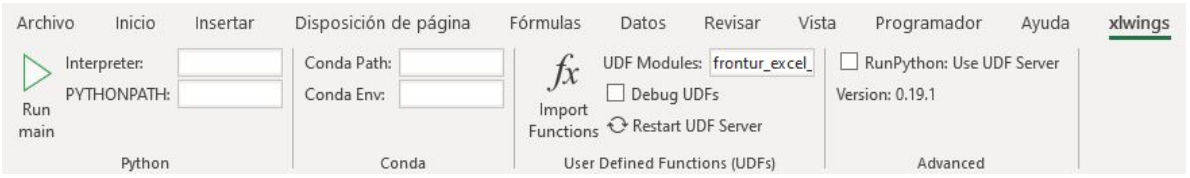


Figura 6.8: Pestaña xlwings

En ella dentro del apartado User Defined Functions (UDF's) aparece la casilla en la que se debe de escribir el nombre del módulo que contiene las funciones e importarlas, la importación en este programa sigue el mismo procedimiento que el que hace python, el archivo de igual manera podría haber estado dentro del mismo directorio y haber funcionado de la misma manera, pero este enfoque no es flexible, ya que todos los archivos que llegasen a requerir de los métodos definidos tendrían que ser movidos a la misma carpeta en la que estuviese el archivo con las funciones definidas.

6.3 frontur_gui

En este caso la interfaz gráfica no es un paquete descargable con pip, la manera de poder usarlo es descargarlo desde github desde cualquiera de los medios que ofrece la plataforma, una vez hecho se instalan los paquetes en requirements.txt con:

pip install -r requirements.txt

En caso de surgir errores en la instalación de Kivy la solución más común es la siguiente:

pip install kivy[base] kivy_examples --pre --extra-index-url https://kivy.org/downloads/simple/

Tras haber hecho la instalación solo es necesario usar el siguiente comando:

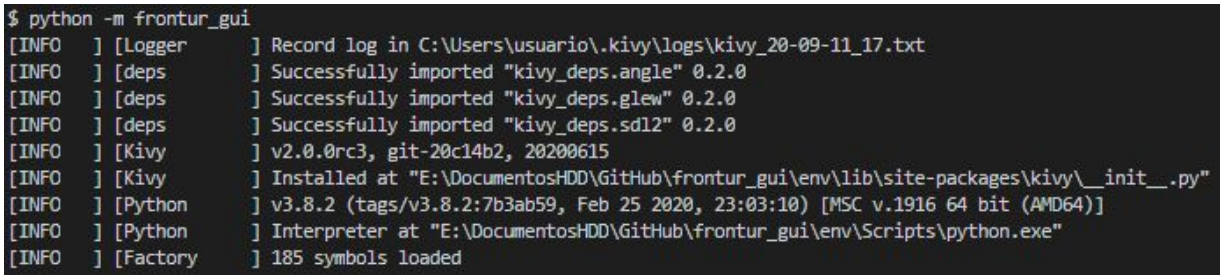


Figura 6.9: Lanzamiento de la interfaz

Tras esto se podrá ver la pantalla principal de la interfaz.

Capítulo 7 Experiencia computacional

Los resultados de los siguientes experimentos fueron hechos sobre la semana del 4 al 10 de febrero de 2019 con 2 encuestadores, 80% de ocupación, 60% de éxito, 30 segundos por encuesta, 10 minutos de descanso entre vuelos, 8 horas de jornada, 7 días de trabajo y objetivo que favorece 2 encuestadores sobre cada vuelo. El límite de ejecución es de 15 minutos.

Tenerife Norte

Sin grupos

Núm países 15 - Núm vuelos 621
Número de variables = 2499
Número de restricciones = 16923
Valor Óptimo objetivo = -7749.0
Tiempo = 22.347 seconds
Vuelos seleccionados = 123
Uso de memoria: 15.4+ KB

Con grupos

Núm países 15 - Núm vuelos 621
Número de variables = 2499
Número de restricciones = 56027
Valor Óptimo objetivo = -8218.0
Tiempo = 350.383 seconds
Vuelos seleccionados = 94
Uso de memoria: 11.8+ KB

Lanzarote

Sin grupos

Núm países 24 - Núm vuelos 495
Número de variables = 2004
Número de restricciones = 9644
Valor Óptimo objetivo = 14590.0
Tiempo = 15.31 seconds
Vuelos seleccionados = 123
Uso de memoria: 15.4+ KB

Con grupos

Núm países 24 - Núm vuelos 495
Número de variables = 2004
Número de restricciones = 36934
Solución no encontrada a tiempo
Tiempo = 886.855 seconds
Uso de memoria: 50.4+ KB

Fuerteventura

Sin grupos

Núm países 25 - Núm vuelos 399
Número de variables = 1621
Número de restricciones = 6443
Valor Óptimo objetivo = 12968.0
Tiempo = 10.837 seconds
Vuelos seleccionados = 118
Uso de memoria: 14.8+ KB

Con grupos

Núm países 25 - Núm vuelos 399
Número de variables = 1621
Número de restricciones = 25155
Solución no encontrada a tiempo
Tiempo = 887.841 seconds
Uso de memoria: 40.6+ KB

La Palma

Sin grupos

Núm países 25 - Núm vuelos 1105
Número de variables = 4445
Número de restricciones = 47601
Valor Óptimo objetivo = 6495.0
Tiempo = 481.332 seconds
Vuelos seleccionados = 148
Uso de memoria: 18.5+ KB

Con grupos

Núm países 25 - Núm vuelos 1105
Número de variables = 4445
Número de restricciones = 179565
Solución no encontrada a tiempo
Tiempo = 821.301 seconds
Uso de memoria: 112.4+ KB

Tenerife Sur

Sin grupos

Núm países 26 - Núm vuelos 638
Número de variables = 2578
Número de restricciones = 14192
Valor Óptimo objetivo = 16348.0
Tiempo = 42.261 seconds
Vuelos seleccionados = 137
Uso de memoria: 17.1+ KB

Con grupos

Núm países 26 - Núm vuelos 638
Número de variables = 2578
Número de restricciones = 61628
Solución no encontrada a tiempo
Tiempo = 871.926 seconds
Uso de memoria: 64.9+ KB

Capítulo 8 Conclusiones y líneas futuras

8.1 Conclusiones

Durante el desarrollo de este trabajo se han implementado herramientas que automatizan la labor de llevar a cabo una tarea repetitiva haciendo uso de tecnologías ya existentes siendo estas Pandas, or-tools, Kivy y demás, para resolver problemas de programación lineal con un diseño que posibilita futuras extensiones que cuenta además con varias formas de interactuar con la aplicación, durante el desarrollo se han experimentado con varias herramientas de desarrollo de código con distintos objetivos que han hecho que el resultado final termine siendo más completo.

Se ha estudiado cómo están organizadas las instituciones de investigación gubernamentales dedicadas a recaudar información sobre el propio territorio y generar información útil a la comunidad.

Se ha aprendido a incorporar procesos de cómputo intensivo en aplicaciones interactivas para una experiencia ininterrumpida para el usuario.

8.2 Líneas futuras

Una posible línea futura que pudiese llegar a seguir el proyecto sería la de ampliar los casos de uso del modelo matemático ya sea incorporando modificaciones el modelo en sí o hacer uso de metaprogramación para adaptarlo a otros casos específicos y generar código de manera dinámica para que se ajuste a las necesidades del usuario, además también de poder colaborar en el futuro con el ISTAC para llevar a cabo el proyecto y conseguir que el proyecto llegue a añadir valor a esta institución.

También se podría investigar el poder mejorar el sistema para poder calcular la distancia o tiempo entre puntos de embarque o vuelos para darle al modelo información más fidedigna sobre el problema que resuelve.

Capítulo 9 Summary and Conclusions

9.1 Summary

During the development of this work tools have been implemented to automatize the labor of a repetitive task making use of existing technologies those being Pandas, or-tools, Kivy and others, to solve linear programming problems with a design that allows future extension that also counts with various ways of interacting with the application, during development various tools of code development have been experimented on with different purposes that made a more well rounded final result.

It has been studied how government research institutions dedicated to collect information of its own territory and generate useful information to the community are organized.

It has been learnt to incorporate compute-heavy processes onto interactive applications for an uninterrupted user experience.

9.2 Conclusions

A possible conclusion is that the project may go on would be to broaden the use cases of the mathematical model either by modifying the model itself making use of metaprogramming to adapt it to other specific cases, generating code dynamically to adjust to the user necessities, in addition to being able to collaborate in the future with the ISTAC to carry out with the project and manage to make the project add value to this institution.

It could be studied the possibility of improving the system by implementing a method to calculate the distance or time between points of embark or flights to provide the model grounded information about the problem it resolves.

Capítulo 10 Presupuesto

Para el desarrollo de este proyecto ha sido hecho con software libre como Pandas y con servicios también gratuitos como Github, esto abarata el presupuesto sustancialmente, los únicos costos vienen dados por el equipo de desarrollo y el salario del desarrollador que emplea un total de 300 horas y su salario está alrededor de 20€ por hora.

Tipo de recurso	Precio
Salarios	3000€
Equipo de desarrollo	900€
Subtotal	3900€

Tabla 7.1: Resumen de tipos

Capítulo 11 Apéndice

11.1 Modelo de optimización

```
x = { (i,k) : solver.BoolVar('x[%i, %i]' % (i, k)) for i in range(sol_par.num_flights)
                                                    for k in pollsters }
y1 = { (i) : solver.BoolVar('y1[%i]' % (i)) for i in range(sol_par.num_flights) }
y2 = { (i) : solver.BoolVar('y2[%i]' % (i)) for i in range(sol_par.num_flights) }
z = { (p) : solver.BoolVar('z[%i]' % (p)) for p in range(sol_par.num_countries) }

[solver.Add(y1[i.index] + y2[i.index] <= 1) for i in sol_par.flights]
[solver.Add(
      sum(x[i.index, k] for k in pollsters)
      == y1[i.index] + 2*y2[i.index]
      ) for i in sol_par.flights]
[solver.Add(
      sum(i.surveys * (y1[i.index] + y2[i.index]) for i in p.flights)
      >= (p.min_polls - p.ine_polls) * z[p.index])
      for p in sol_par.countries]
[solver.Add( x[i.index, k] + x[j.index, k] <= 1 ) for k in pollsters]
[solver.Add(
      x[i.index, k] + x[j.index, k]
      <= 1 + sum(x[i.index, k_not] for k_not in pollsters if k_not != k)
      ) for k in pollsters]
[solver.Add(
      x[i.index, k] + x[j.index, k]
      <= 1 + sum(x[j.index, k_not] for k_not in pollsters if k_not != k)
      ) for k in pollsters]

solver.Maximize(
      sum(f.surveys*(y1[f.index] + y2[f.index]*2) for f in sol_par.flights) -
      100*sum(p.num_travelers*(1-z[p.index]) for p in sol_par.countries))
```

Anexos

Repositorio del proyecto frontur_utilities

https://github.com/miguelbravo7/frontur_utilities

Repositorio del proyecto frontur_excel_addin

https://github.com/miguelbravo7/frontur_excel_addin

Repositorio del proyecto frontur_gui

https://github.com/miguelbravo7/frontur_gui

Bibliografía

[1] ISTAC información - 11/09/2020

http://www.gobiernodecanarias.org/istac/istac/

[2] FronTur información - 11/09/2020

http://estadisticas.tourspain.es/es-ES/estadisticas/frontur/Paginas/default.aspx

[3] FronTur Canarias Metodología - 11/09/2020

http://www.gobiernodecanarias.org/istac/descargas/E16028B/metodologia_FRONTUR.pdf

[4] INE información- 11/09/2020

https://www.ine.es/ss/Satellite?L=es_ES&c=Page&cid=1254735910183&p=1254735910183&pagename=INE%2FINELayout

[5] GESLOT información - 11/09/2020

https://www.slotcoordination.es/csee/Satellite/Slots/es/Page/1237545152190/1237544440201/

[6] Turespaña - 11/09/2020

https://es.wikipedia.org/wiki/Turespa%C3%B1a

[7] Git Wikipedia - 11/09/2020

https://en.wikipedia.org/wiki/Git

[8] Github Personal Access Token - 11/09/2020

https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token

[9] Github Workflows - 11/09/2020

https://docs.github.com/es/actions/configuring-and-managing-workflows/configuring-a-workflow

[10] Python virtual environments- 11/09/2020

https://docs.python.org/3/library/venv.html

[11] Pytest - 11/09/2020

https://docs.pytest.org/en/stable/

[12] Parametrizing fixtures and test functions - 11/09/2020

https://docs.pytest.org/en/stable/parametrize.html#pytest-mark-parametrize

[13] Unittest Mocks - 11/09/2020

https://docs.python.org/3/library/unittest.mock.html#the-patchers

[14] Pytest tmpdir - 11/09/2020

https://docs.pytest.org/en/stable/tmpdir.html

[15] Python documentation tool - 11/09/2020

https://pdoc3.github.io/pdoc/doc/pdoc/#gsc.tab=0

[16] Python or-tools - 11/09/2020

https://developers.google.com/optimization/introduction/python

[17] OR-Tools MIP Solvers - 11/09/2020

https://developers.google.com/optimization/mip/integer_opt

[18] Wikipedia Linear programming - 11/09/2020

https://en.wikipedia.org/wiki/Linear_programming

[19] Pandas DataFrame - 11/09/2020

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

[20] Python dataclasses - 11/09/2020

https://docs.python.org/3/library/dataclasses.html

[21] Coverage tool - 11/09/2020

https://coverage.readthedocs.io/en/coverage-5.2.1/

[22] Python packaging - 11/09/2020

https://packaging.python.org/tutorials/packaging-projects/

[23] License chooser - 11/09/2020

https://choosealicense.com/

[24] MANIFEST file - 11/09/2020

https://packaging.python.org/guides/using-manifest-in/

[25] Python click package - 11/09/2020

https://click.palletsprojects.com/en/7.x/

[26] Kivy homepage - 11/09/2020

https://kivy.org/#home

[27] xlwings UDF’s- 11/09/2020

https://docs.xlwings.org/en/stable/udfs.html