1 Definicion del problema

1.1 Entender el problema

1.2 Arsenal

- Paradigma: orientado a objetos
- Lenguaje: Python
- Herramientas: API, JSON

1.3 Requisitos funcionales

Obtener el clima de la ciudad a donde se va a viajar

1.4 Requisitos no funcionales

Garantizar la eficacia, seguridad al procesar los datos y la resistencia a fallos del programa.

2 Analisis del problema

Tenemos como entrada un archivo csv con los datos de cada vuelo que contiene el nombre del aereopuerto en version codigo de la ciudad, longitud y latitud, y como entrada del usuario que seleccione la ciudad a la cual va ir. Como salida tenemos datos json que son convertidos a strings legibles y sintetizadas para una mejor comprension El modelo de datos es principalmente a traves de matrices que funcionan como tablas hash donde en cada casilla almacenan la informacion requerida, estas determinan el desempeño en una complejidad de acceso constante aunque ocupan gran espacio, es por eso que tenemos que ver que tamaño nos conviene en relacion tamaño-colisiones donde se busca minimizar ambos, se ajusta a los requisitos funcionales porque al final guarda informacion a la cual posteriormente vamos a acceder como ya se menciono en un costo constante y nos da los datos que requerimos para el siguiente paso o mostrar la informacion dependiendo lo que se requiere. No puede haber algo mejor ya que el acceso es constante y si puede haber algo peor como lo son las listas, pilas, colas o incluso un arreglo donde la

informacion almacenada no tiene ninguna relacion con el indice donde esta guardada, son peores porque la busqueda dentro de estas estructuras es de tiempo lineal ya que hay que buscar uno a uno. $\bigcup_1^n A$

3 Seleccion de la mejor alternativa

4 Pseudocodigo

```
1: if you love me then
2:     kiss me!
3: else if you like me then
4:     kiss me!
5: else
6:     kiss me?
7: end if
```

5 Mantenimiento

6 Pruebas