This file aims to explain the functionalities of the ETL.

1. The ETL must **create** CPT Restaurant items and CPT Food items
2. The ETL must **modify** CPT Restaurant items and CPT Food items
3. The ETL must have an interface where we can **upload** .csv files (or something else if needed)
4. So, the interface should have **4 buttons**, in order to allow us to upload the following files:
   1. NEW CPT Restaurant  
      This file will contain the columns explained in readme\_restaurant file.
   2. NEW CPT Food  
      This file will contain the columns explained in readme\_food file.
   3. MODIFY CPT Restaurant  
      This file will contain all the columns explained in readme\_restaurant file, BUT only some of them will contain a value.  
      The ETL will modify the Custom Field/Taxonomy Terms/Title of the CPT Restaurant only if the column contains a value, while will not modify that ones with empty value
   4. MODIFY CPT Food  
      This file will contain all the columns explained in readme\_food file, BUT only some of them will contain a value.  
      The ETL will modify the Custom Field/Taxonomy Terms/Title of the CPT Food only if the column contains a value, while will not modify that ones with empty value.
5. Since we could wrongly upload files through a button instead of another, the ETL must perform the following **checks:**
   1. File uploaded through the button A (NEW CPT Restaurant) -> before to create a new CPT Restaurant, the ETL must check - row by row - if there is already a CPT Restaurant item with the ‘*Relationship Key*’ value as its custom field.  
        
      If a CPT Restaurant item already present in the DB has the above value, it means that the CPT Restaurant already exists and the ETL must ignore that row!  
      It must also copy that row in a report that we will receive after the ETL process will end, which will indicate us the erroneous rows.
   2. File uploaded through the button B (NEW CPT Food) -> before to create a new CPT Food, the ETL must check - row by row - if there is already a CPT Food item with both:  
        
      - the ‘*Restaurant Key*’ value as value of the custom field ‘*Relationship key*’ in the Restaurant associated (in readme\_food file is explained how to set the custom field ‘*Restaurant Associated*’).   
        
      AND   
        
      - the ‘*Food title*’ value as its title.  
        
      If a CPT Food item already present in the DB has both the above values, it means that the CPT Food already exists and the ETL must ignore that row!  
      It must also copy that row in a report that we will receive after the ETL process will end, which will indicate us the erroneous rows.
   3. File uploaded through the button C (MODIFY CPT Restaurant) -> before to modify a CPT Restaurant, the ETL must check - row by row - if there is already a CPT Restaurant item with the ‘*Relationship Key*’ value as its custom field.  
        
      If a CPT Restaurant item already present in the DB has the above value, it means that the CPT Restaurant already exists and can be modified!  
        
      Otherwise, if a CPT Restaurant with the above value does NOT already exist, ETL must ignore that row!  
      It must also copy that row in a report that we will receive after the ETL process will end, which will indicate us the erroneous rows.
   4. File uploaded through the button D (MODIFY CPT Food) -> before to modify a CPT Food, the ETL must check - row by row - if there is already a CPT Food item with both:  
        
      - the ‘*Restaurant Key*’ value as value of the custom field ‘*Relationship key*’ in the Restaurant associated (in readme\_food file is explained how to set the custom field ‘*Restaurant Associated*’).   
        
      AND   
        
      - the ‘*Food title*’ value as its title.  
        
      If a CPT Food item already present in the DB has both the above values, it means that the CPT Food already exists and can be modified!  
        
      Otherwise, if a CPT Food with the above values does NOT already exist, ETL must ignore that row!  
      It must also copy that row in a report that we will receive after the ETL process will end, which will indicate us the erroneous rows.
6. As per point 5), we will receive a report after each ETL process, and these **report files** will be of 4 types (depends on which ETL process was used).