Ottimizzazione dei Garanti Accademici

Realizzato da Colli Simone e Merenda Saverio Mattia

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 doc2xlsx Namespace Reference

Variables

- bool all_empty = True
- df = pd.DataFrame(rows, columns=["CORSO DI STUDIO","CORSO DI STUDIO", "CODICE U-GOV", "CLASSE", "PRESIDENTE", "NOTE", "DOCENZA DI RIFERIMENTO", "DIPARTIMENTO"])
- str dipartimento = ""
- doc = Document(file_path)
- str file_path = "../dataset/originali/elenco_2024-2025.docx"
- index
- list row = [cell.text.strip() for cell in row.cells]
- list rows = [ ]
- list tabella = [ ]
- list tabelle = [ ]

4.1.1 Variable Documentation

4.1.1.1 all_empty

```
bool all_empty = True
```

4.1.1.2 df

```
df = pd.DataFrame(rows, columns=["CORSO DI STUDIO","CORSO DI STUDIO", "CODICE U-GOV", "CLASSE",
"PRESIDENTE", "NOTE", "DOCENZA DI RIFERIMENTO", "DIPARTIMENTO"])
```

4.1.1.3 dipartimento

```
str dipartimento = ""
```

4.1.1.4 doc

```
doc = Document(file_path)
```

4.1.1.5 file_path

```
str file_path = "../dataset/originali/elenco_2024-2025.docx"
```

4.1.1.6 index

```
index
```

4.1.1.7 row

```
list row = [cell.text.strip() for cell in row.cells]
```

4.1.1.8 rows

```
list rows = []
```

4.1.1.9 tabella

```
list tabella = []
```

4.1.1.10 tabelle

```
list tabelle = []
```

4.2 hard-tester Namespace Reference

Functions

- init_corsi (filename)
- init_corsi_matricole (filepathCorsi, filepathProf)
- init_matricole (filename)
- main ()
- run ()
- write (filters_corsi)

  FUNZIONI PRINCIPALI ##.

Variables

- str dataset_corsi_dir = 'dataset/corsi/'

  VARIABILI GLOBALI ##################################.
- str filepathCorsi = dataset_corsi_dir + 'codici-corsi.csv'
- str filepathProf = dataset_corsi_dir + 'codici-matricole.csv'
- int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
- str path_coperture = "dataset/coperture.xlsx"
- str path_docenti = "dataset/docenti.xlsx"
- str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
- list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx", "dataset/immatricolati/CU.↩ xlsx"]

4.2.1 Function Documentation

4.2.1.1 init_corsi()

```
init_corsi (
              filename )
```

```
Inizializza e salva i corsi con descrizione e codice tipo corso.

@param filename: Nome del file di output per i corsi.
@type filename: str
```

4.2.1.2 init_corsi_matricole()

```
init_corsi_matricole (
              filepathCorsi,
              filepathProf )
```

```
Inizializza i file dei corsi e delle matricole.

@param filepathCorsi: Percorso per il file dei corsi.
@type filepathCorsi: str
@param filepathProf: Percorso per il file delle matricole.
@type filepathProf: str
```

4.2.1.3 init_matricole()

```
init_matricole (
              filename )
```

```
Inizializza e salva le matricole dei docenti non a contratto.

@param filename: Nome del file di output per le matricole.
@type filename: str
```

4.2.1.4 main()

```
main ( )
```

```
Funzione principale che esegue l'inizializzazione dei corsi e delle matricole,
filtra i corsi e gestisce i test con esecuzione di script esterni.
```

4.2.1.5 run()

```
run ( )
```

```
Esegue un processo esterno (script bash) e analizza l'output.

@return: -1 se l'output contiene "UNSATISFIABLE", 1 se contiene "OPTIMUM FOUND", 0 per altri casi.
@rtype: int
```

4.2.1.6 write()

```
write (
              filters_corsi )
```

FUNZIONI PRINCIPALI ##.

4.2.2 Variable Documentation

4.2.2.1 dataset_corsi_dir

```
str dataset_corsi_dir = 'dataset/corsi/'
```

VARIABILI GLOBALI ###################################.

4.2.2.2 filepathCorsi

```
str filepathCorsi = dataset_corsi_dir + 'codici-corsi.csv'
```

4.2.2.3 filepathProf

```
str filepathProf = dataset_corsi_dir + 'codici-matricole.csv'
```

4.2.2.4 NUMERO_MINIMO_DI_INSEGNAMENTI

```
int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
```

4.2.2.5 path_coperture

```
str path_coperture = "dataset/coperture.xlsx"
```

4.2.2.6 path_docenti

```
str path_docenti = "dataset/docenti.xlsx"
```

4.2.2.7 path_docenti_a_contratto

```
str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
```

4.2.2.8 paths_immatricolati

```
list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx",
"dataset/immatricolati/CU.xlsx"]
```

4.3 main Namespace Reference

Functions

- clean_text (text)
- init_corsi (filename)
- init_corsi_matricole (filepathCorsi, filepathProf)
- init_matricole (filename)

  FUNZIONI PRINCIPALI ######################################.
- main ()

Variables

- int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
- str path_coperture = "dataset/coperture.xlsx"

  VARIABILI GLOBALI ################################.
- str path_docenti = "dataset/docenti.xlsx"
- str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
- str path_elenco_allegato = "dataset/elenco_allegato.xlsx"
- list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx", "dataset/immatricolati/CU.↩ xlsx"]

4.3.1 Function Documentation

4.3.1.1 clean_text()

```
clean_text (
              text )
```

4.3.1.2 init_corsi()

```
init_corsi (
              filename )
```

```
Inizializza e salva un file con i codici e le descrizioni dei corsi.

Questa funzione:
1. Carica i dati delle coperture.
2. Crea una colonna "Overview" che unisce la descrizione del corso e il tipo di corso.
3. Estrae i codici dei corsi e rimuove i duplicati.
4. Filtra i corsi sulla base dei codici unici e salva i risultati nel file specificato.

Args:
    filename (str): Nome del file di output dove salvare i codici dei corsi.

:return: None
:rtype: None
```

4.3.1.3 init_corsi_matricole()

```
init_corsi_matricole (
              filepathCorsi,
              filepathProf )
```

```
Inizializza i file relativi ai corsi e alle matricole.

Questa funzione:
1. Inizializza e salva i dati relativi ai corsi e alle matricole.
2. Se i file di output non esistono, li crea e li salva.
3. Visualizza un messaggio di completamento e termina l'esecuzione.

Args:
    filepathCorsi (str): Percorso del file di output per i corsi.
    filepathProf (str): Percorso del file di output per le matricole.

:return: Nessuno. Il programma termina dopo l'esecuzione.
:rtype: None
```

4.3.1.4 init_matricole()

```
init_matricole (
              filename )
```

FUNZIONI PRINCIPALI ##.

```
Carica e salva le matricole dei docenti non a contratto.

Questa funzione:
1. Carica le matricole dei docenti dal file `path_docenti`.
2. Rimuove i duplicati e le salva.
3. Carica i dati dalle coperture e filtra in base alle matricole dei docenti.
4. Salva i dati filtrati nel file specificato da `filename`.

Args:
    filename (str): Nome del file di output dove salvare le matricole dei docenti.

:return: None
:rtype: None
```

4.3.1.5 main()

```
main ( )
```

```
Funzione principale per l'elaborazione e la gestione dei dati.

La funzione esegue le seguenti operazioni:
1. Inizializza i dataset dei corsi e delle matricole se non esistono.
2. Filtra e salva i dati dei corsi, docenti e coperture.
3. Genera i file richiesti in base ai parametri specificati (corsi, docenti, coperture, immatricolati).

- Crea e gestisce i filtri per i corsi in base ai parametri di input.
- Gestisce i file di dati e li salva nella struttura appropriata.

:return: Nessuno. Il flusso di elaborazione termina.
:rtype: None
```

4.3.2 Variable Documentation

4.3.2.1 NUMERO_MINIMO_DI_INSEGNAMENTI

```
int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
```

4.3.2.2 path_coperture

```
str path_coperture = "dataset/coperture.xlsx"
```

VARIABILI GLOBALI ##################################.

4.3.2.3 path_docenti

```
str path_docenti = "dataset/docenti.xlsx"
```

4.3.2.4 path_docenti_a_contratto

```
str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
```

4.3.2.5 path_elenco_allegato

```
str path_elenco_allegato = "dataset/elenco_allegato.xlsx"
```

4.3.2.6 paths_immatricolati

```
list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx",
"dataset/immatricolati/CU.xlsx"]
```

4.4 modules Namespace Reference

Namespaces

- namespace course_parser
- namespace dataset_loader
- namespace dataset_manager

4.5 modules.course_parser Namespace Reference

Classes

- class CourseParser

4.6 modules.dataset_loader Namespace Reference

Classes

- class DatasetLoader

4.7 modules.dataset_manager Namespace Reference

Classes

- class DatasetManager

4.8 post-proc Namespace Reference

Variables

- answerBlocks = re.findall(r"(Answer: \d+\n.*?Optimization: -?\d+)", content, re.DOTALL)
- args = parser.parse_args()
- codice_corso = entry["Codice corso"]
- content = file.read()
- bool DEV = False
- df_coperture = pd.read_excel(path_coperture, engine="openpyxl", dtype=str)
- df_docenti = pd.read_excel(path_docenti, engine="openpyxl", dtype=str)
- df_excel = pd.DataFrame(excel_data, columns=["Cognome e Nome", "Matricole", "Codice Corso", "Des. Corso"])
- df_jolly = pd.DataFrame(jolly_data, columns=["Codice Corso", "Des. Corso", "Numero di Jolly"])
- encoding
- engine
- ensure_ascii
- list excel_data = [ ]
- str excel_output_path = "garanti.xlsx"
- False

- garanti = defaultdict(list)
- help
- indent
- index
- list jolly_data = [ ]
- str jolly_output_path = "contratti.xlsx"
- json_file
- last_block = answerBlocks[-1]
- matches = re.findall(r"garante\((\d+),(\d+),\d+,(\w+)\)", last_block)
- list matricole = [int(docente[0]) for docente in docenti if docente[1] != "c"]
- nome_corso
- nome_docenti
- num_jolly = len(docenti) - len(matricole)
- str output_path = "result.json"
- parser = argparse.ArgumentParser(description="Estrai l'ultima risposta da un file Clingo e genera una struttura dati.")
- str path_coperture = "../dataset/coperture.xlsx"
- str path_docenti = "../dataset/docenti.xlsx"
- list result = [ ]
- str
- type

4.8.1 Variable Documentation

4.8.1.1 answerBlocks

```
answerBlocks = re.findall(r"(Answer:  \d+\n.*?Optimization:  -?\d+)", content, re.DOTALL)
```

4.8.1.2 args

```
args = parser.parse_args()
```

4.8.1.3 codice_corso

```
codice_corso = entry["Codice corso"]
```

4.8.1.4 content

```
content = file.read()
```

4.8.1.5 DEV

```
bool DEV = False
```

4.8.1.6 df_coperture

```
df_coperture = pd.read_excel(path_coperture, engine="openpyxl", dtype=str)
```

4.8.1.7 df_docenti

```
df_docenti = pd.read_excel(path_docenti, engine="openpyxl", dtype=str)
```

4.8.1.8 df_excel

```
df_excel = pd.DataFrame(excel_data, columns=["Cognome e Nome", "Matricole", "Codice Corso",
"Des.  Corso"])
```

4.8.1.9 df_jolly

```
df_jolly = pd.DataFrame(jolly_data, columns=["Codice Corso", "Des.  Corso", "Numero di Jolly"])
```

4.8.1.10 encoding

```
encoding
```

4.8.1.11 engine

```
engine
```

4.8.1.12 ensure_ascii

```
ensure_ascii
```

4.8.1.13 excel_data

```
list excel_data = []
```

4.8.1.14 excel_output_path

```
excel_output_path = "garanti.xlsx"
```

4.8.1.15 False

```
False
```

4.8.1.16 garanti

```
garanti = defaultdict(list)
```

4.8.1.17 help

```
help
```

4.8.1.18 indent

```
indent
```

4.8.1.19 index

```
index
```

4.8.1.20 jolly_data

```
list jolly_data = []
```

4.8.1.21 jolly_output_path

```
jolly_output_path = "contratti.xlsx"
```

4.8.1.22 json_file

```
json_file
```

4.8.1.23 last_block

```
last_block = answerBlocks[-1]
```

4.8.1.24 matches

```
matches = re.findall(r"garante\((\d+),(\d+),\d+,(\w+)\)", last_block)
```

4.8.1.25 matricole

```
list matricole = [int(docente[0]) for docente in docenti if docente[1] != "c"]
```

4.8.1.26 nome_corso

```
nome_corso
```

Initial value:

```
00001 =  df_coperture.loc[
00002         df_coperture["Cod. Corso di Studio"] == codice_corso,
00003         "Des. Corso di Studio"
00004     ].values
```

4.8.1.27 nome_docenti

```
nome_docenti
```

Initial value:

```
00001 =  df_docenti.loc[
00002         df_docenti["Matricola"].isin(map(str, matricole)),
00003         "Cognome e Nome"
00004     ].tolist()
```

4.8.1.28 num_jolly

```
num_jolly = len(docenti) - len(matricole)
```

4.8.1.29 output_path

```
str output_path = "result.json"
```

4.8.1.30 parser

```
parser = argparse.ArgumentParser(description="Estrai l'ultima risposta da un file Clingo e
genera una struttura dati.")
```

4.8.1.31 path_coperture

```
str path_coperture = "../dataset/coperture.xlsx"
```

4.8.1.32 path_docenti

```
str path_docenti = "../dataset/docenti.xlsx"
```

4.8.1.33 result

```
result = []
```

4.8.1.34 str

```
str
```

4.8.1.35 type

```
type
```

4.9 sanitize Namespace Reference

Functions

- aggiorna_cod_tipo_corso (row, df_allegato)
- compute_extra_data ()
- compute_remained ()
- merge_elenco_allegato ()
- sanitize ()
- sanitize_codici_corso (df)
- sanitize_coperture ()
- sanitize_docenti ()
- sanitize_elenco_24_25 ()

Variables

- str path_allegato_d = "../dataset/allegato-d-sanitized.xlsx"
- str path_coperture = "../dataset/coperture.xlsx"
- str path_coperture_contratti = "../dataset/docenti_a_contratto.xlsx"
- str path_coperture_rimaste = "../dataset/insegnamenti_senza_docente.xlsx"
- str path_docenti = "../dataset/docenti.xlsx"
- str path_elenco_24_25 = "../dataset/elenco_2024-2025.xlsx"
- str path_elenco_allegato = "../dataset/elenco_allegato.xlsx"
- str path_ns_coperture = "../dataset/originali/coperture.xlsx"
- str path_ns_docenti = "../dataset/originali/docenti.xlsx"
- str path_ns_elenco_24_25 = "../dataset/originali/elenco_2024-2025.xlsx"

4.9.1 Function Documentation

4.9.1.1 aggiorna_cod_tipo_corso()

```
aggiorna_cod_tipo_corso (
              row,
              df_allegato )
```

```
Aggiorna il codice del tipo di corso in base alla descrizione del corso.

La funzione esamina la descrizione del corso e, se corrisponde ad alcune parole chiave specifiche,
aggiorna il codice tipo corso (ad esempio 'LT' o 'LM').

:param row: Una riga del DataFrame contenente le informazioni del corso.
:type row: pandas.Series
:return: Il codice tipo corso aggiornato.
:rtype: str
:raises ValueError: Se la descrizione del corso non corrisponde a nessuna delle regole mappate.
```

4.9.1.2 compute_extra_data()

```
compute_extra_data ( )
```

Calcola i dati extra per i corsi senza docenti e li salva nel file di output.

La funzione esegue le seguenti operazioni:
1. Carica i dati dal file `path_ns_coperture`.
2. Rimuove le righe in cui il campo "Matricola" è vuoto.
3. Converte la "Matricola" in formato stringa.
4. Filtra i dati mantenendo solo i record con matricole che non esistono nel file dei docenti.
5. Sanifica il campo "Cod. Tipo Corso" tramite la funzione `sanitize_codici_corso`.
6. Salva i dati nel file `path_coperture_contratti`.

:return: Nessuno. I dati vengono salvati nel file di output.
:rtype: None

4.9.1.3 compute_remained()

```
compute_remained ( )
```

Calcola i corsi rimasti senza docente e li salva nel file di output.

La funzione esegue le seguenti operazioni:
1. Carica i dati completi dal file `path_ns_coperture`, quelli già assegnati ai docenti dal file `path_copertu
   e quelli assegnati a contratti dal file `path_coperture_contratti`.
2. Esclude i corsi già assegnati a docenti (sia determinati che a contratto) dal set di corsi completi.
3. Rimuove le righe con valori vuoti in tutte le colonne.
4. Salva i corsi rimasti nel file `path_coperture_rimaste`.

:return: Nessuno. I dati dei corsi rimasti vengono salvati nel file di output.
:rtype: None

4.9.1.4 merge_elenco_allegato()

```
merge_elenco_allegato ( )
```

4.9.1.5 sanitize()

```
sanitize ( )
```

Funzione principale per sanificare i dati.

La funzione chiama tutte le operazioni di sanificazione:
- Sanifica i dati dei docenti.
- Sanifica i dati delle coperture.
- Calcola i dati extra per i corsi senza docenti.
- Calcola i corsi rimasti senza docente.

:return: Nessuno. Vengono salvati i dati nei rispettivi file di output.
:rtype: None

4.9.1.6 sanitize_codici_corso()

```
sanitize_codici_corso (
              df )
```

Sanifica i codici dei corsi in base alla descrizione e al tipo del corso.

La funzione sostituisce "L" con "LT" per uniformare il tipo di corso e applica le regole di aggiornamento tramite la funzione `aggiorna_cod_tipo_corso`.

:param df: DataFrame contenente le informazioni dei corsi.
:type df: pandas.DataFrame
:return: DataFrame con i codici tipo corso aggiornati.
:rtype: pandas.DataFrame

4.9.1.7 sanitize_coperture()

```
sanitize_coperture ( )
```

Sanifica i dati di copertura, rimuovendo le righe con matricola vuota e mantenendo solo i record con matricole esistenti nel file dei docenti.

La funzione esegue le seguenti operazioni:
1. Carica i dati dal file `path_ns_coperture`.
2. Rimuove le righe in cui il campo "Matricola" è vuoto.
3. Converte la "Matricola" in formato stringa.
4. Filtra i dati mantenendo solo le matricole esistenti nel file dei docenti.
5. Sanifica il campo "Cod. Tipo Corso" tramite la funzione `sanitize_codici_corso`.
6. Salva i dati sanificati nel file `path_coperture`.

:return: Nessuno. I dati sanificati vengono salvati nel file di output.
:rtype: None

4.9.1.8 sanitize_docenti()

```
sanitize_docenti ( )
```

Sanifica i dati dei docenti, rimuovendo eventuali caratteri non numerici dalle matricole e salvando i risultati nel file di output.

La funzione rimuove i caratteri `.` e `,` dalle matricole e converte il campo in formato stringa prima di salvare il DataFrame nel file Excel `path_docenti`.

:return: Nessuno. I dati dei docenti sanificati vengono salvati nel file di output.
:rtype: None

4.9.1.9 sanitize_elenco_24_25()

```
sanitize_elenco_24_25 ( )
```

4.9.2 Variable Documentation

4.9.2.1 path_allegato_d

```
str path_allegato_d = "../dataset/allegato-d-sanitized.xlsx"
```

4.9.2.2 path_coperture

```
str path_coperture = "../dataset/coperture.xlsx"
```

4.9.2.3 path_coperture_contratti

```
str path_coperture_contratti = "../dataset/docenti_a_contratto.xlsx"
```

4.9.2.4 path_coperture_rimaste

```
str path_coperture_rimaste = "../dataset/insegnamenti_senza_docente.xlsx"
```

4.9.2.5 path_docenti

```
str path_docenti = "../dataset/docenti.xlsx"
```

4.9.2.6 path_elenco_24_25

```
str path_elenco_24_25 = "../dataset/elenco_2024-2025.xlsx"
```

4.9.2.7 path_elenco_allegato

```
str path_elenco_allegato = "../dataset/elenco_allegato.xlsx"
```

4.9.2.8 path_ns_coperture

```
str path_ns_coperture = "../dataset/originali/coperture.xlsx"
```

4.9.2.9 path_ns_docenti

```
str path_ns_docenti = "../dataset/originali/docenti.xlsx"
```

4.9.2.10 path_ns_elenco_24_25

```
str path_ns_elenco_24_25 = "../dataset/originali/elenco_2024-2025.xlsx"
```

4.10 sanitizer-allegato Namespace Reference

Variables

- df
- engine
- str file_path = "../dataset/allegato-d.ods"
- index
- sheet1_df = pd.read_excel(file_path, engine="odf", sheet_name=0, dtype=str)
- sheet2_df = pd.read_excel(file_path, engine="odf", sheet_name=1, dtype=str)

4.10.1 Variable Documentation

4.10.1.1 df

```
df
```

Initial value:

```
00001 =  sheet2_df.merge(
00002     sheet1_df,
00003     left_on=["Area", "Codice area", "Tipo laurea"],
00004     right_on=["Area", "Codice area", "Tipo laurea"],
00005     how="left"
00006 )
```

4.10.1.2 engine

```
engine
```

4.10.1.3 file_path

```
str file_path = "../dataset/allegato-d.ods"
```

4.10.1.4 index

```
index
```

4.10.1.5 sheet1_df

```
sheet1_df = pd.read_excel(file_path, engine="odf", sheet_name=0, dtype=str)
```

4.10.1.6 sheet2_df

```
sheet2_df = pd.read_excel(file_path, engine="odf", sheet_name=1, dtype=str)
```

Chapter 5

Class Documentation

5.1 CourseParser Class Reference

Public Member Functions

- __init__ (self)
- add_courses (self, courses)
- parse (self)

Public Attributes

- parser

5.1.1 Detailed Description

```
Classe gestire il parsing dei corsi utilizzando il modulo argparse.
Consente di aggiungere corsi come argomenti dinamici e di effettuare il parsing
degli argomenti della riga di comando.
```

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
__init__ (
              self )
```

```
Inizializza il parser per i corsi.
Il parser supporta l'argomento `--all` per selezionare tutti i corsi.
```

5.1.3 Member Function Documentation

5.1.3.1 add_courses()

```
add_courses (
              self,
              courses )
```

```
Ogni corso viene aggiunto come un argomento con nome '--<codice>',
dove '<codice>' rappresenta il codice del corso.

:param courses: Dizionario contenente i corsi. La chiave è il codice del corso
                (str), mentre il valore è il nome del corso (str).
:type courses: dict
```

5.1.3.2 parse()

```
parse (
              self )
```

```
Effettua il parsing degli argomenti della riga di comando.

:return: Un oggetto contenente gli argomenti parsati come attributi.
:rtype: argparse.Namespace
```

5.1.4 Member Data Documentation

5.1.4.1 parser

```
parser
```

The documentation for this class was generated from the following file:

- src/modules/course_parser.py

5.2 DatasetLoader Class Reference

Public Member Functions

- __init__ (self, input_file_path="dataset/coperture.xlsx")
- filter_by_values (self, filters, dataset=None, only_prefix=False)
- get_values (self, dataset=None, columns=None)
- save_to_file (self, df, output_file_path, file_format="csv")

Public Attributes

- input_file_path

5.2.1 Detailed Description

Classe per la gestione del caricamento, filtraggio e salvataggio dei dati da file Excel o CSV.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
__init__ (
              self,
              input_file_path = "dataset/coperture.xlsx" )
```

Inizializza la classe DatasetLoader con il percorso al dataset.
:param input_file_path: Path relativo al file Excel da leggere (default: "dataset/coperture.xlsx").

5.2.3 Member Function Documentation

5.2.3.1 filter_by_values()

```
filter_by_values (
              self,
              filters,
              dataset = None,
              only_prefix = False )
```

Filtra i dati in base a più valori specifici in diverse colonne, convertendo tutti i valori in stringhe.
:param filters: Dizionario in cui le chiavi sono i nomi delle colonne e i valori sono liste di valori da filtr
Esempio: {'Cod. Tipo Corso': ['LM'], 'SSD': ['INF/01', 'MAT/05']}.
:param dataset: DataFrame esistente da utilizzare (default: None). Se None, viene caricato il file Excel.
:return: DataFrame contenente solo le righe che soddisfano i criteri di filtraggio.
:raises ImportError: Se il pacchetto openpyxl non è installato.
:raises KeyError: Se una delle colonne specificate nei filtri non esiste nel dataset.
:raises Exception: Per altri errori durante il filtraggio.

5.2.3.2 get_values()

```
get_values (
              self,
              dataset = None,
              columns = None )
```

Carica un file Excel o utilizza un dataset esistente, filtrando opzionalmente per colonne specifiche.
:param dataset: DataFrame esistente da utilizzare (default: None). Se None, viene caricato il file Excel.
:param columns: Lista di colonne da mantenere (esempio: ['Matricola', 'Cognome']).
:return: DataFrame contenente tutte le righe, filtrato per le colonne specificate (se fornite).
:raises ImportError: Se il pacchetto openpyxl non è installato.
:raises KeyError: Se una o più colonne specificate non esistono nel dataset.

5.2.3.3 save_to_file()

```
save_to_file (
              self,
              df,
              output_file_path,
              file_format = "csv" )
```

```
Salva il DataFrame su un file specificato.
:param df: Il DataFrame da salvare.
:param output_file_path: Percorso del file di output.
:param file_format: Formato del file, può essere 'csv' o 'excel'.
:raises ValueError: Se il formato specificato non è 'csv' o 'excel'.
:raises Exception: Per altri errori durante il salvataggio del file.
```

5.2.4 Member Data Documentation

5.2.4.1 input_file_path

```
input_file_path
```

The documentation for this class was generated from the following file:

- src/modules/dataset_loader.py

5.3 DatasetManager Class Reference

Public Member Functions

- __init__ (self, dataset_path="dataset/")
- get_courses (self)
- get_professors (self)
- scrivi_coperture (self, df, filename)
- scrivi_docenti (self, df, filename)
- scrivi_docenti_a_contratto (self, df, filename)
- scrivi_ministeriali (self, df, filename)
- scrivi_presidenti (self, df, filename)

Public Attributes

- dataset_path

5.3.1 Detailed Description

```
Classe per la gestione di file di dataset e la generazione di file ASP strutturati.

La classe permette di:
- Recuperare informazioni sui corsi e sui professori da file dataset.
- Generare file ASP (.lp) contenenti i dati processati e strutturati.
```

5.3.2 Constructor & Destructor Documentation

5.3.2.1 __init__()

```
__init__ (
              self,
              dataset_path = "dataset/" )
```

Inizializza la classe DatasetManager con il percorso della cartella contenente i file di dataset.

:param dataset_path: Percorso alla cartella contenente i file di dataset. Default: "dataset/".
:type dataset_path: str

5.3.3 Member Function Documentation

5.3.3.1 get_courses()

```
get_courses (
              self )
```

Ottiene un dizionario dei corsi basandosi sui file presenti nella cartella del dataset.

:return: Dizionario con chiavi come "Cod. Corso di Studio" e valori come "Overview".
:rtype: dict
:raises FileNotFoundError: Se la cartella del dataset non esiste.

5.3.3.2 get_professors()

```
get_professors (
              self )
```

Ottiene un dizionario che associa ciascun corso ai professori tramite la loro matricola.

:return: Dizionario con chiavi come "Cod. Corso di Studio" e valori come insiemi di matricole.
:rtype: dict
:raises FileNotFoundError: Se la cartella del dataset non esiste.

5.3.3.3 scrivi_coperture()

```
scrivi_coperture (
              self,
              df,
              filename )
```

Genera un file ASP contenente informazioni sui corsi, TAF e SSD.

:param df: DataFrame contenente i dati relativi ai corsi.
:type df: pandas.DataFrame
:param filename: Nome del file di output (senza estensione).
:type filename: str
:raises Exception: Se si verifica un errore durante la scrittura del file.

5.3.3.4 scrivi_docenti()

```
scrivi_docenti (
              self,
              df,
              filename )
```

Genera un file ASP contenente informazioni sui docenti.

```
:param df: DataFrame contenente i dati relativi ai docenti.
:type df: pandas.DataFrame
:param filename: Nome del file di output (senza estensione).
:type filename: str
:raises Exception: Se si verifica un errore durante la scrittura del file.
```

5.3.3.5 scrivi_docenti_a_contratto()

```
scrivi_docenti_a_contratto (
              self,
              df,
              filename )
```

Genera un file ASP contenente informazioni sui docenti a contratto.

```
:param df: DataFrame contenente i dati relativi ai docenti a contratto.
:type df: pandas.DataFrame
:param filename: Nome del file di output (senza estensione).
:type filename: str
:raises Exception: Se si verifica un errore durante la scrittura del file.
```

5.3.3.6 scrivi_ministeriali()

```
scrivi_ministeriali (
              self,
              df,
              filename )
```

Genera un file ASP contenente i parametri ministeriali per i corsi.

```
:param df: DataFrame contenente i dati relativi ai corsi.
:type df: pandas.DataFrame
:param filename: Nome del file di output (senza estensione).
:type filename: str
:raises Exception: Se si verifica un errore durante la scrittura del file.
```

5.3.3.7 scrivi_presidenti()

```
scrivi_presidenti (
              self,
              df,
              filename )
```

5.3.4 Member Data Documentation

5.3.4.1 dataset_path

```
dataset_path
```

The documentation for this class was generated from the following file:

- src/modules/dataset_manager.py

Chapter 6

File Documentation

6.1 src/hard-tester.py File Reference

Namespaces

- namespace hard-tester

Functions

- init_corsi (filename)
- init_corsi_matricole (filepathCorsi, filepathProf)
- init_matricole (filename)
- main ()
- run ()
- write (filters_corsi)

  FUNZIONI PRINCIPALI ##.

Variables

- str dataset_corsi_dir = 'dataset/corsi/'

  VARIABILI GLOBALI ################################.
- str filepathCorsi = dataset_corsi_dir + 'codici-corsi.csv'
- str filepathProf = dataset_corsi_dir + 'codici-matricole.csv'
- int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
- str path_coperture = "dataset/coperture.xlsx"
- str path_docenti = "dataset/docenti.xlsx"
- str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
- list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx", "dataset/immatricolati/CU.↩ xlsx"]

6.2 src/main.py File Reference

Namespaces

- namespace main

Functions

- clean_text (text)
- init_corsi (filename)
- init_corsi_matricole (filepathCorsi, filepathProf)
- init_matricole (filename)

  FUNZIONI PRINCIPALI ##.
- main ()

Variables

- int NUMERO_MINIMO_DI_INSEGNAMENTI = 9
- str path_coperture = "dataset/coperture.xlsx"

  VARIABILI GLOBALI ################################.
- str path_docenti = "dataset/docenti.xlsx"
- str path_docenti_a_contratto = "dataset/docenti_a_contratto.xlsx"
- str path_elenco_allegato = "dataset/elenco_allegato.xlsx"
- list paths_immatricolati = ["dataset/immatricolati/LT.xlsx", "dataset/immatricolati/LM.xlsx", "dataset/immatricolati/CU.↩ xlsx"]

6.3 src/modules/__init__.py File Reference

6.4 src/modules/course_parser.py File Reference

Classes

- class CourseParser

Namespaces

- namespace modules
- namespace modules.course_parser

6.5 src/modules/dataset_loader.py File Reference

Classes

- class DatasetLoader

Namespaces

- namespace modules
- namespace modules.dataset_loader

6.6 src/modules/dataset_manager.py File Reference

Classes

- class DatasetManager

Namespaces

- namespace modules
- namespace modules.dataset_manager

6.7 src/utils/doc2xlsx.py File Reference

Namespaces

- namespace doc2xlsx

Variables

- bool all_empty = True
- df = pd.DataFrame(rows, columns=["CORSO DI STUDIO","CORSO DI STUDIO", "CODICE U-GOV", "CLASSE", "PRESIDENTE", "NOTE", "DOCENZA DI RIFERIMENTO", "DIPARTIMENTO"])
- str dipartimento = ""
- doc = Document(file_path)
- str file_path = "../dataset/originali/elenco_2024-2025.docx"
- index
- list row = [cell.text.strip() for cell in row.cells]
- list rows = [ ]
- list tabella = [ ]
- list tabelle = [ ]

6.8 src/utils/post-proc.py File Reference

Namespaces

- namespace post-proc

Variables

- answerBlocks = re.findall(r"(Answer: \d+\n.*?Optimization: -?\d+)", content, re.DOTALL)
- args = parser.parse_args()
- codice_corso = entry["Codice corso"]
- content = file.read()
- bool DEV = False
- df_coperture = pd.read_excel(path_coperture, engine="openpyxl", dtype=str)
- df_docenti = pd.read_excel(path_docenti, engine="openpyxl", dtype=str)
- df_excel = pd.DataFrame(excel_data, columns=["Cognome e Nome", "Matricole", "Codice Corso", "Des. Corso"])
- df_jolly = pd.DataFrame(jolly_data, columns=["Codice Corso", "Des. Corso", "Numero di Jolly"])
- encoding
- engine
- ensure_ascii
- list excel_data = [ ]
- str excel_output_path = "garanti.xlsx"
- False
- garanti = defaultdict(list)
- help
- indent
- index
- list jolly_data = [ ]
- str jolly_output_path = "contratti.xlsx"
- json_file
- last_block = answerBlocks[-1]
- matches = re.findall(r"garante\((\d+),(\d+),\d+,(\w+)\)", last_block)
- list matricole = [int(docente[0]) for docente in docenti if docente[1] != "c"]
- nome_corso
- nome_docenti
- num_jolly = len(docenti) - len(matricole)
- str output_path = "result.json"
- parser = argparse.ArgumentParser(description="Estrai l'ultima risposta da un file Clingo e genera una struttura dati.")
- str path_coperture = "../dataset/coperture.xlsx"
- str path_docenti = "../dataset/docenti.xlsx"
- list result = [ ]
- str
- type

6.9 src/utils/sanitize.py File Reference

Namespaces

- namespace sanitize

Functions

- aggiorna_cod_tipo_corso (row, df_allegato)
- compute_extra_data ()
- compute_remained ()
- merge_elenco_allegato ()
- sanitize ()
- sanitize_codici_corso (df)
- sanitize_coperture ()
- sanitize_docenti ()
- sanitize_elenco_24_25 ()

Variables

- str path_allegato_d = "../dataset/allegato-d-sanitized.xlsx"
- str path_coperture = "../dataset/coperture.xlsx"
- str path_coperture_contratti = "../dataset/docenti_a_contratto.xlsx"
- str path_coperture_rimaste = "../dataset/insegnamenti_senza_docente.xlsx"
- str path_docenti = "../dataset/docenti.xlsx"
- str path_elenco_24_25 = "../dataset/elenco_2024-2025.xlsx"
- str path_elenco_allegato = "../dataset/elenco_allegato.xlsx"
- str path_ns_coperture = "../dataset/originali/coperture.xlsx"
- str path_ns_docenti = "../dataset/originali/docenti.xlsx"
- str path_ns_elenco_24_25 = "../dataset/originali/elenco_2024-2025.xlsx"

6.10 src/utils/sanitizer-allegato.py File Reference

Namespaces

- namespace sanitizer-allegato

Variables

- df
- engine
- str file_path = "../dataset/allegato-d.ods"
- index
- sheet1_df = pd.read_excel(file_path, engine="odf", sheet_name=0, dtype=str)
- sheet2_df = pd.read_excel(file_path, engine="odf", sheet_name=1, dtype=str)

Index