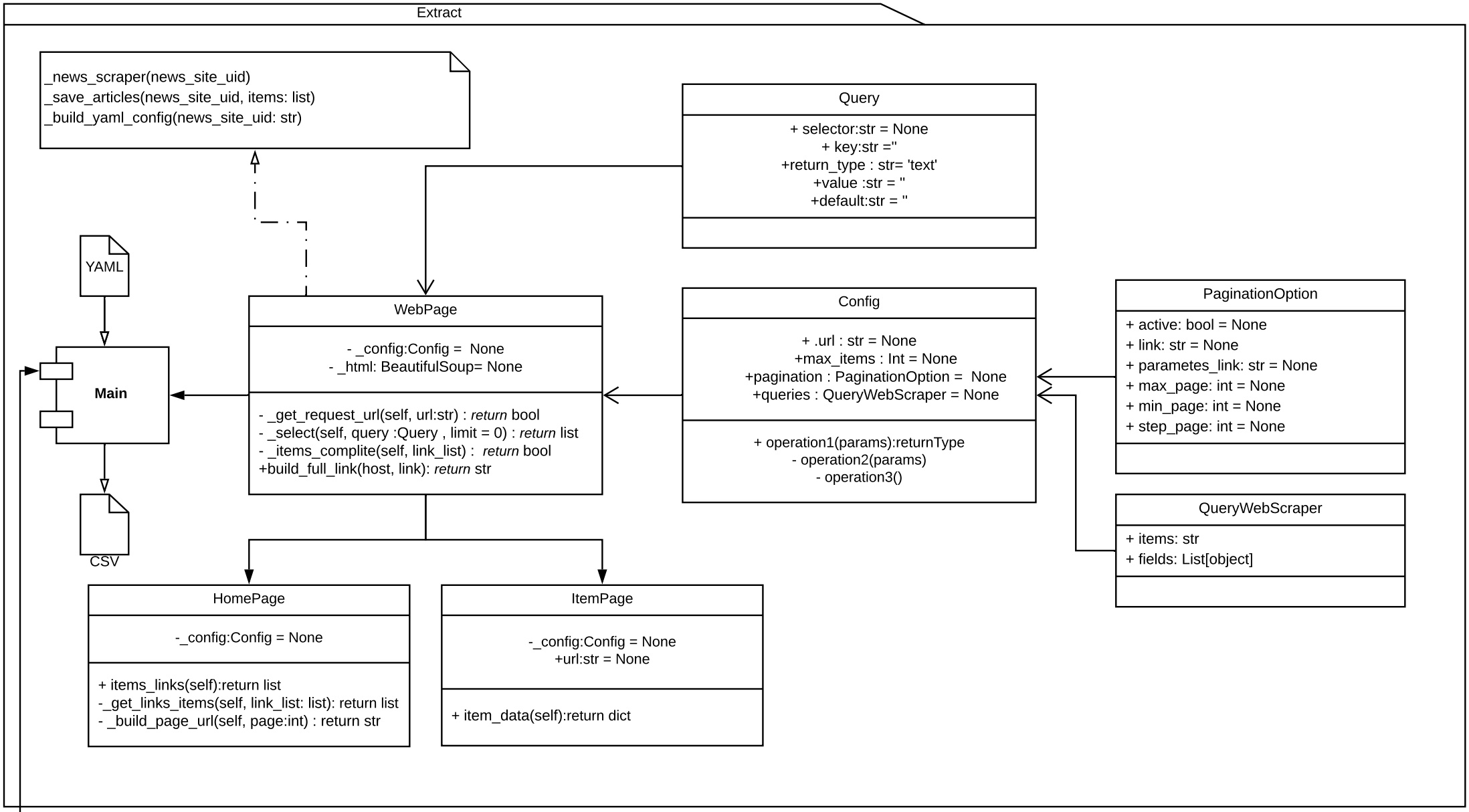
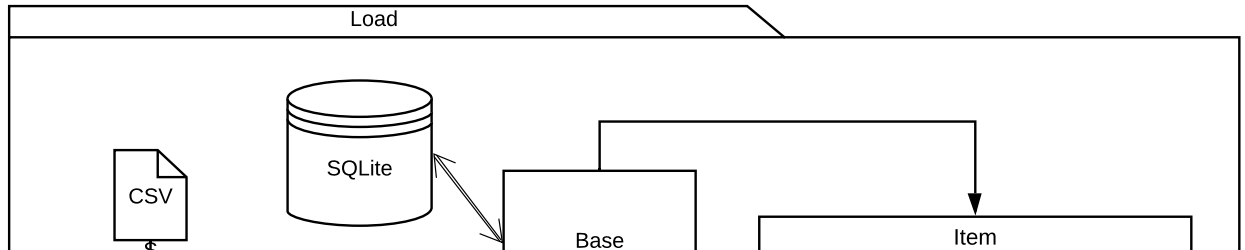
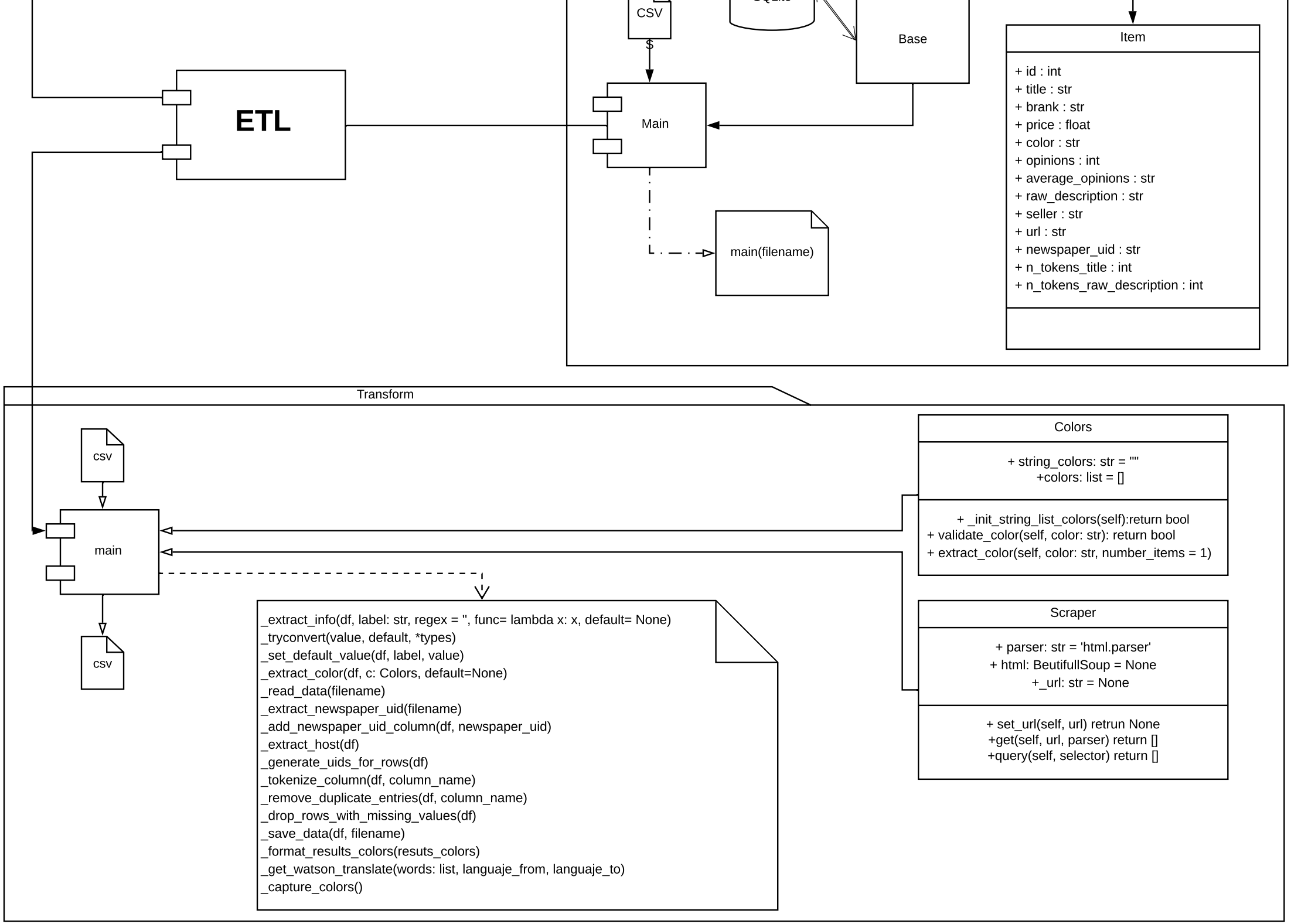

Extract
_news_scraper(news_site_uid)
_save_articles(news_site_uid, items: list)
_build_yaml_config(news_site_uid: str)
Query
+ selector:str = None
+ key:str =''
+return_type : str= 'text'
+value :str = ''
+default:str = ''
YAML
Main
CSV
WebPage
- _config:Config = None
- _html: BeautifulSoup= None
- _get_request_url(self, url:str) : return bool
- _select(self, query :Query , limit = 0) : return list
- _items_complite(self, link_list) : return bool
+build_full_link(host, link): return str
Config
+ .url : str = None
+max_items : Int = None
+pagination : PaginationOption = None
+queries : QueryWebScraper = None
+ operation1(params):returnType
- operation2(params)
- operation3()
PaginationOption
+ active: bool = None
+ link: str = None
+ parametes_link: str = None
+ max_page: int = None
+ min_page: int = None
+ step_page: int = None
QueryWebScraper
+ items: str
+ fields: List[object]
HomePage
-_config:Config = None
+ items_links(self):return list
-_get_links_items(self, link_list: list): return list
- _build_page_url(self, page:int) : return str
ItemPage
-_config:Config = None
+url:str = None
+ item_data(self):return dict


Load
CSV
SQLite
Base
Item


ETL
CSV
$
Base
Main
main(filename)
Item
+ id : int
+ title : str
+ brank : str
+ price : float
+ color : str
+ opinions : int
+ average_opinions : str
+ raw_description : str
+ seller : str
+ url : str
+ newspaper_uid : str
+ n_tokens_title : int
+ n_tokens_raw_description : int
Transform
csv
main
csv
_extract_info(df, label: str, regex = '', func= lambda x: x, default= None)
_tryconvert(value, default, *types)
_set_default_value(df, label, value)
_extract_color(df, c: Colors, default=None)
_read_data(filename)
_extract_newspaper_uid(filename)
_add_newspaper_uid_column(df, newspaper_uid)
_extract_host(df)
_generate_uids_for_rows(df)
_tokenize_column(df, column_name)
_remove_duplicate_entries(df, column_name)
_drop_rows_with_missing_values(df)
_save_data(df, filename)
_format_results_colors(resuts_colors)
_get_watson_translate(words: list, languaje_from, languaje_to)
_capture_colors()
Colors
+ string_colors: str = ""
+colors: list = []
+ _init_string_list_colors(self):return bool
+ validate_color(self, color: str): return bool
+ extract_color(self, color: str, number_items = 1)
Scraper
+ parser: str = 'html.parser'
+ html: BeutifullSoup = None
+_url: str = None
+ set_url(self, url) retrun None
+get(self, url, parser) return []
+query(self, selector) return []