DSD+

Maintenance Report

Corso di Ingegneria, Gestione ed Evoluzione del Software

Docente:

Andrea De Lucia

Tutor:

Gilberto Recupito

Partecipanti:

Daniele Fabiano 0522501738

Francesco Paolo D'Antuono 0522501767

Indice

Descrizione del problema

Negli ultimi anni, alcuni studi si sono incentrati sui Data Smell, segnali o indizi sulla presenza di problemi o anomalie latenti nei dati, che richiedono un'attenta e dettagliata analisi. Per questo motivo sono nati alcuni tool che permettono la detection di smell, uno di questi è DSD (Data Smell Detection). Esso si basa sul tool di convalida dei dati open-source Great Expectations e si concentra sul rilevamento degli smell, analizzando un dataset mediante l'utilizzo di tecniche rule-based. DSD+ ha lo scopo di estendere la suite di detection di smell esistente. Inoltre si intende migliorare l'attuale meccanismo di report dei dataset, integrando il calcolo delle metriche di dimensione della qualità dei dati (data quality dimensions). Rispetto al dataset analizzato, sarà possibile visualizzare i risultati dell'esecuzione del tool, con lo scopo di seguire i cambiamenti ottenuti nel tempo a seguito di nuove analisi, per formare una base di indicazioni sul miglioramento/peggioramento di qualità del dataset modificato.

Sistema Attuale

Descrizione del sistema

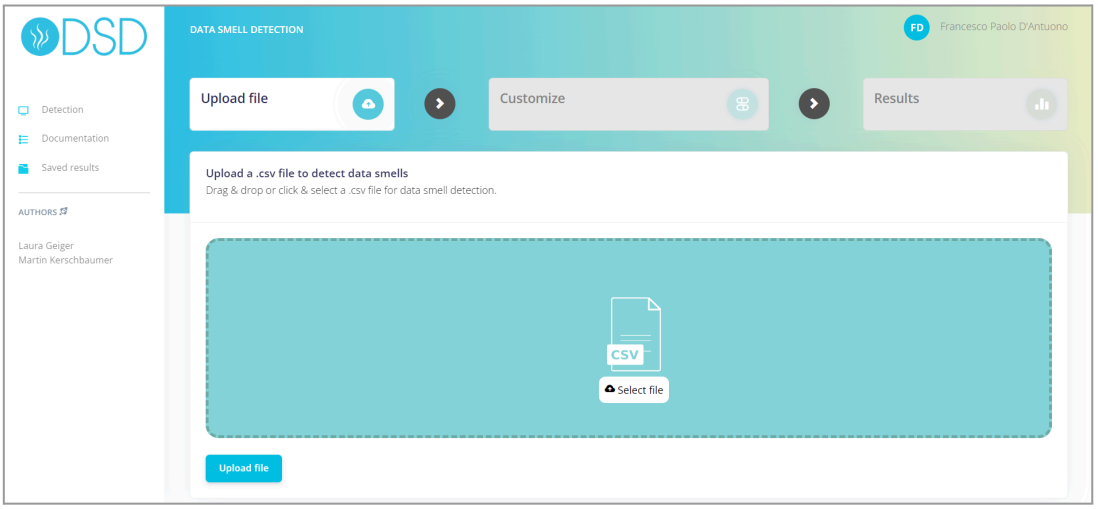
DSD (Data Smell Detection) è un tool sviluppato per consentire l'identificazione di data smell all'interno di dataset. L'implementazione si basa sulla libreria Great Expectations, che è uno strumento open-source progettato per aiutare gli utenti a verificare e documentare la qualità dei dati. Il progetto è stato inizialmente realizzato per la versione python 3.6. Risulta però possibile eseguire il tool sull'ultima versione supportata di python 3.8 (3.8.19). Il tool mette a disposizione una dashboard basata sul template Argon Dashboard Django, che permette di avere una gestione già definita relativamente alla registrazione e autenticazione dell'utente. Inoltre consente di accedere alle funzionalità del sistema che effettuano la detection di data smell sul dataset caricato. Per fornire una maggiore comprensibilità del sistema, sono descritte le principali pagine del sistema.

Link al repository del progetto: https://github.com/CpDant/DSD-plus

Descrizione delle pagine

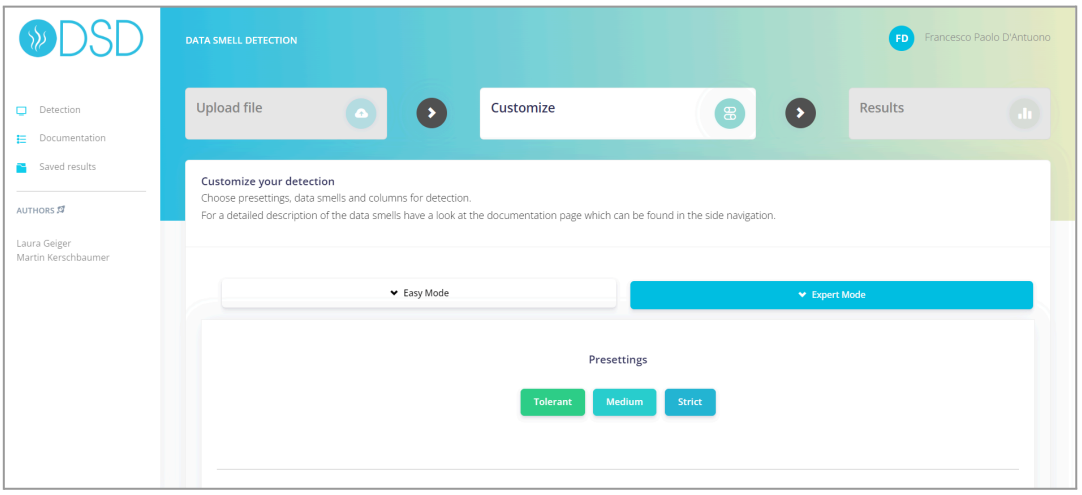
Home page

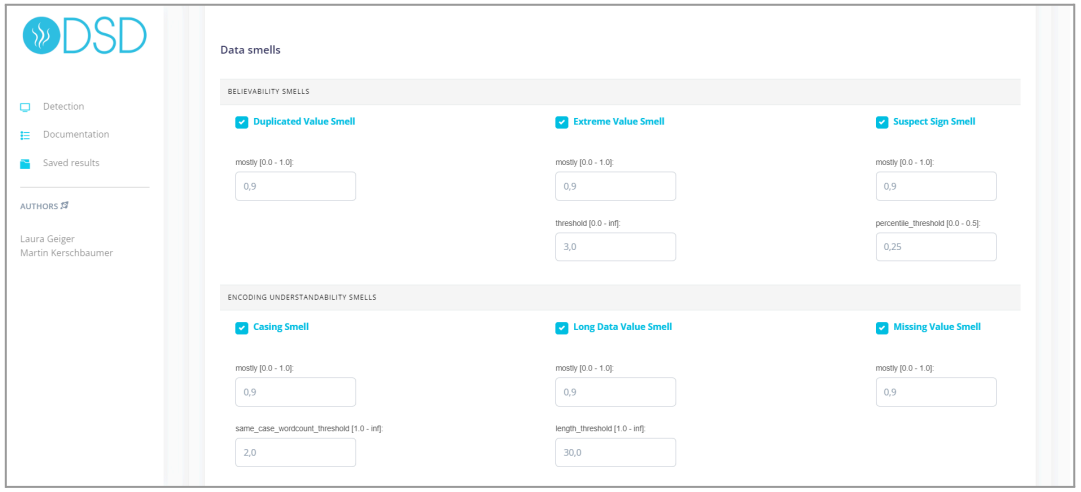
L'immagine di seguito mostra come è composta l'home page. A sinistra vi è una barra laterale dove è possibile accedere alle funzionalità principali della web application. In alto a destra vi è la possibilità di accedere al proprio account e al centro viene descritta la funzionalità principale di detection di data smell, con un riquadro che consente l'inserimento del dataset che verrà analizzato.

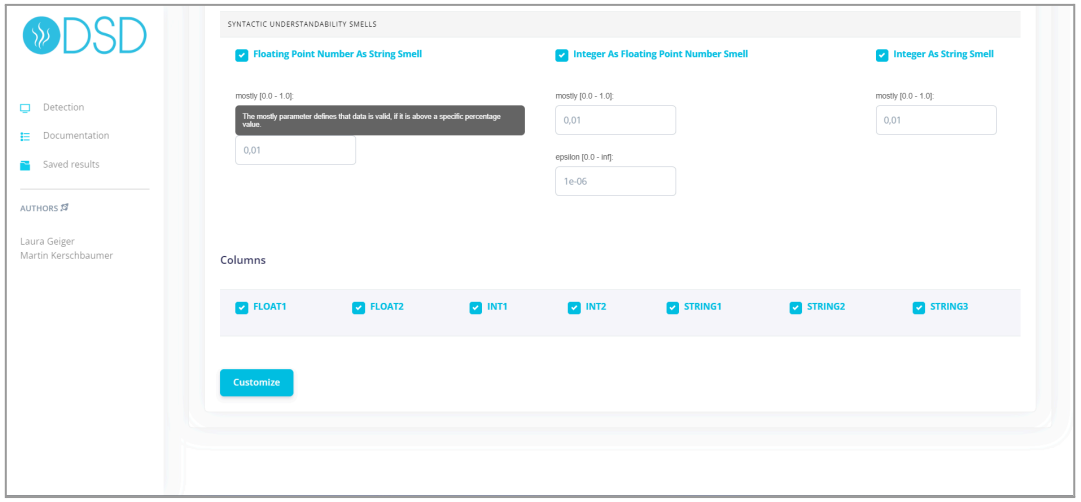


Customization page

In questa pagina vengono deteminati i parametri specifici che verranno utilizzati dal sistema, per effettuare la detection di data smell in un certo tipo. Inoltre si può scegliere anche di detectare un sottoinsieme di smell specifici, evitando di effettuare analisi inutili per quel tipo di dataset. Le immagini seguenti mostrano ciò che è stato descritto in precedenza.

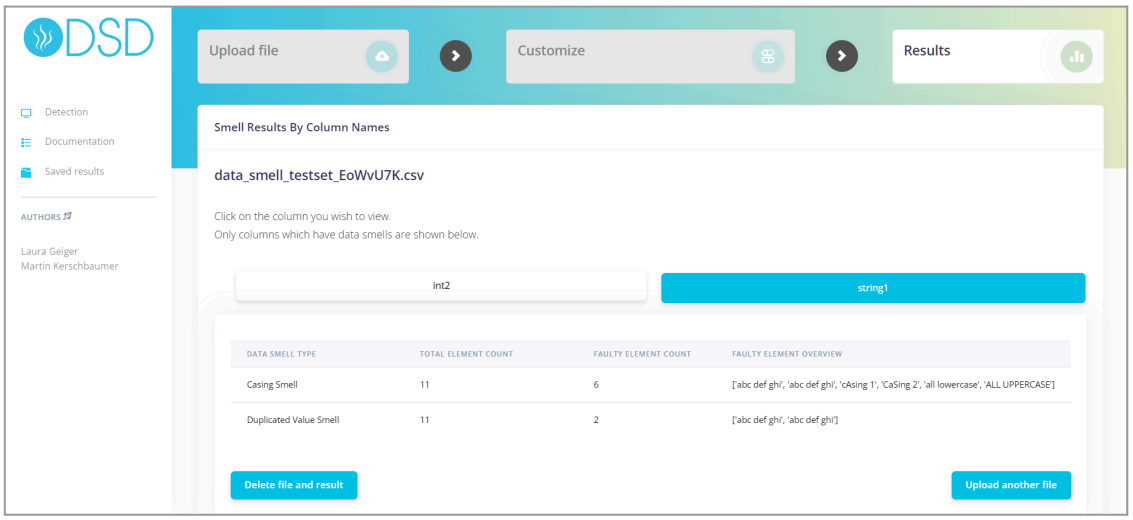






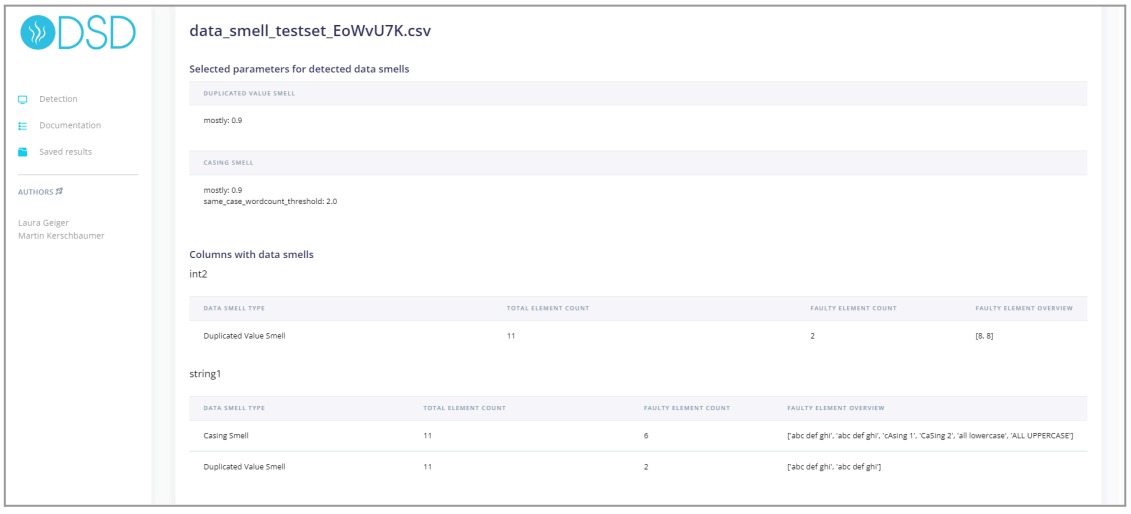
Result page

La seguente immagine mostra la pagina dei risultati della detection appena effettuata. La divisione dei risultati viene fatta per le colonne del dataset.



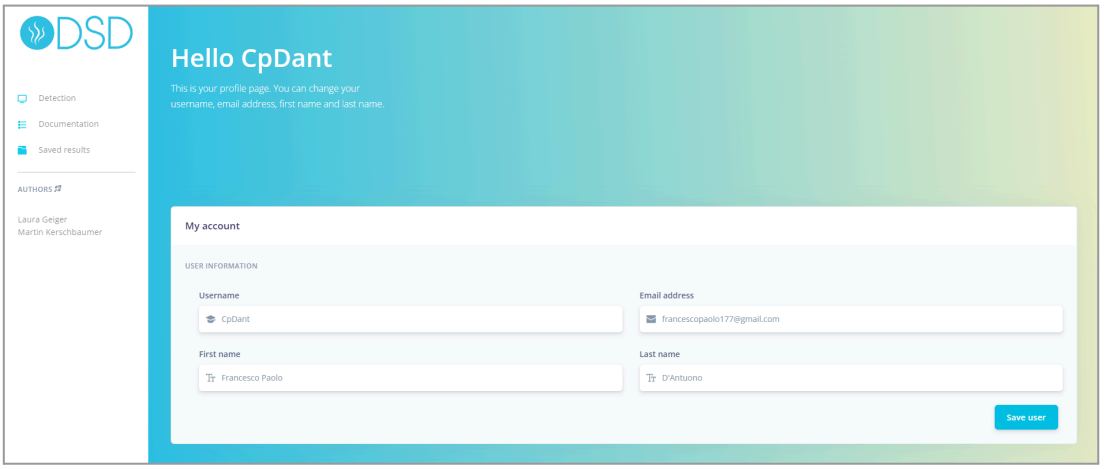
Saved results page

Questa pagina mostra i risultati salvati nel sistema delle detection effettuate in passato, su dataset differenti. L'immagine seguente mostra come vengono visualizzate le informazioni. Da notare la differenza con la Result page, che genera incoerenza.



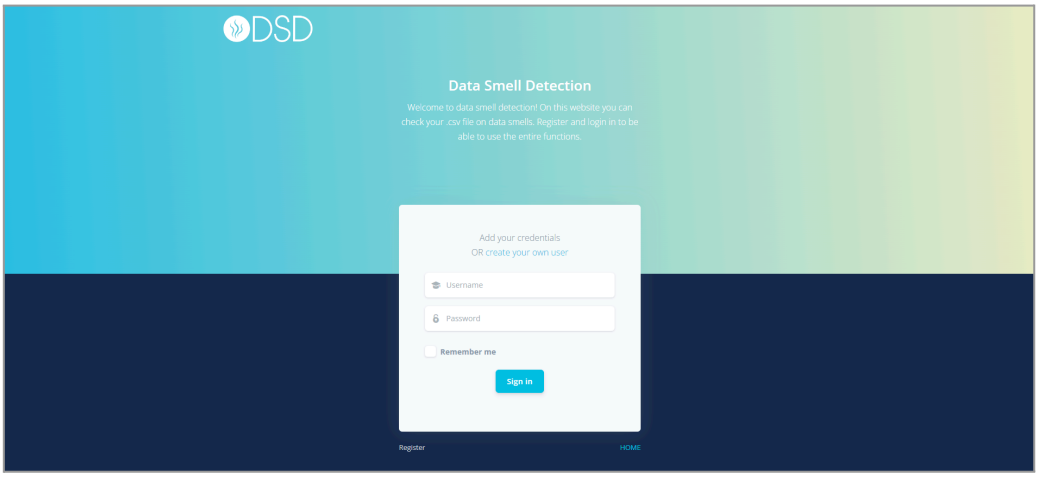
My profile page

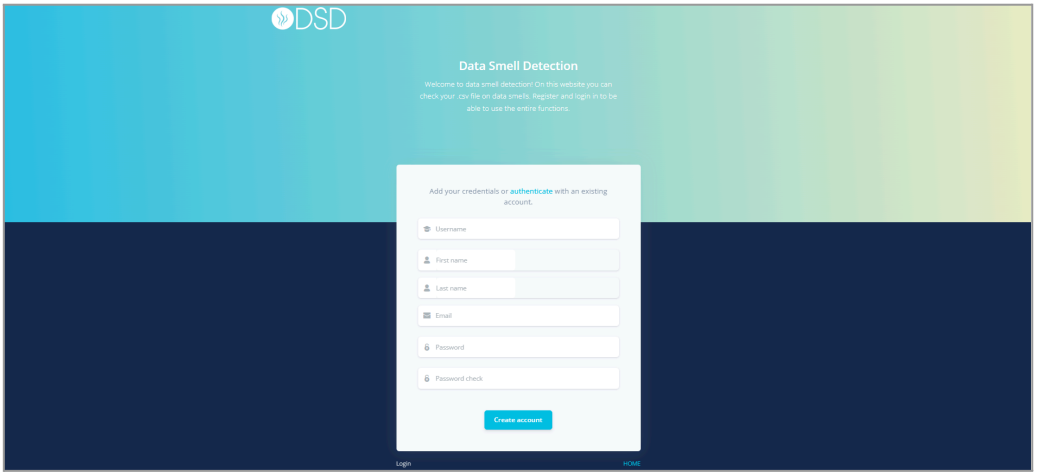
La seguente immagine mostra quali informazioni possono essere visualizzate e modificate dall'utente registrato.



Login page & Sign up page

Le seguenti immagini mostrano la pagina di login e di sign up del sistema, per accedere all'account dell'utente registrato e per registrare un nuovo utente.





Installazione del sistema

L'installazione del sistema è stata descritta nel README del repository e può avvenire in due modalità diverse:

Installazione locale

I passi per eseguire l'installazione locale sono:

1. Installare Python3.8 (se già non è installato nel sistema). Se si tratta di sistemi Windows con WSL Ubuntu seguire i seguenti passaggi:
    a. Aggiornare Ubuntu alla versione più recente con i comandi:
        ```
        sudo apt update, sudo apt upgrade
        ```
    b. Importare su Ubuntu Python 3.8 PPA (una repository che contiene le versioni di Python scaricabili e utilizzabili) con il comando:
        ```
        sudo add-apt-repository ppa:deadsnakes/ppa -y.
        ```
    c. Aggiornare nuovamente Ubuntu per permettere al sistema di riconoscere la nuova repository aggiunta.
    d. Installare Python 3.8 con il comando:
        ```
        sudo apt install python3.8
        ```
    e. Infine, bisogna installare alcuni pacchetti addizionali per far funzionare il tutto correttamente con i seguenti comandi:
        ```
        sudo apt install python3.8-dev,
        sudo apt install python3.8-venv,
        sudo apt install python3.8-distutils
        ```
2. Installare pip con il comando:
    ```
    sudo apt-get install python3.8-pip
    ```
3. Installare il pacchetto virtualenv, che consente di usufruire del virtual environment di python con il comando:
    ```
    sudo pip install virtualenv
    ```
4. Avviare il virtual environment con il comando:
    ```
    virtualenv env
    ```
5. Attivare il virtual environment partendo dalla cartella padre del progetto con il comando:
    ```
    source env/bin/activate
    ```

6. Dopo il passaggio 5, si passa all'installazione dei requisiti e della libreria di detection di data smell. Innanzitutto spostarsi sulla cartella data_smell_detection dalla root del progetto con il comando:

   ```
   cd data_smell_detection
   ```

7. Installare i requisiti con il comando:

   ```
   pip install -r requirements-dev.in
   ```

8. (Facoltativo) Per costruire e distribuire i pacchetti di Python utilizzare il seguente comando:

   ```
   python setup.py install
   ```

NB: Il sistema è configurato per funzionare anche senza l'installazione diretta della libreria, in quanto sono importati i path del progetto della medesima. Nel caso si voglia comunque installare la libreria, ogni volta che si effettua una modifica ad essa è necessario ripetere il passaggio 8.

Per il setup della web application, i passaggi da effettuare sono:

1. Spostarsi sulla cartella *argon-dashboard-django* dalla cartella root del sistema con il comando:

   ```
   cd web_application/argon-dashboard-django/
   ```

2. Installare i moduli richiesti con il comando:

   ```
   pip install -r requirements.txt
   ```

3. Per creare le tabelle del database eseguire i seguenti comandi:

   ```
   python manage.py makemigrations,
   ```

   ```
   python manage.py migrate
   ```

4. Infine, per avviare la web application eseguire il seguente comando:

   ```
   python manage.py runserver
   ```

Installazione containerizzata (Docker)

I passi per riuscire a far funzionare il progetto con Docker sono i seguenti:

1. Assicurarsi di aver installato Docker
2. Posizionandosi sulla root del progetto, per effettuare la build del medesimo, eseguire il seguente comando: `sudo docker-compose build`
3. Avviare la web application con il comando: `sudo docker-compose up -d`

Reverse Engineering

System & Object design

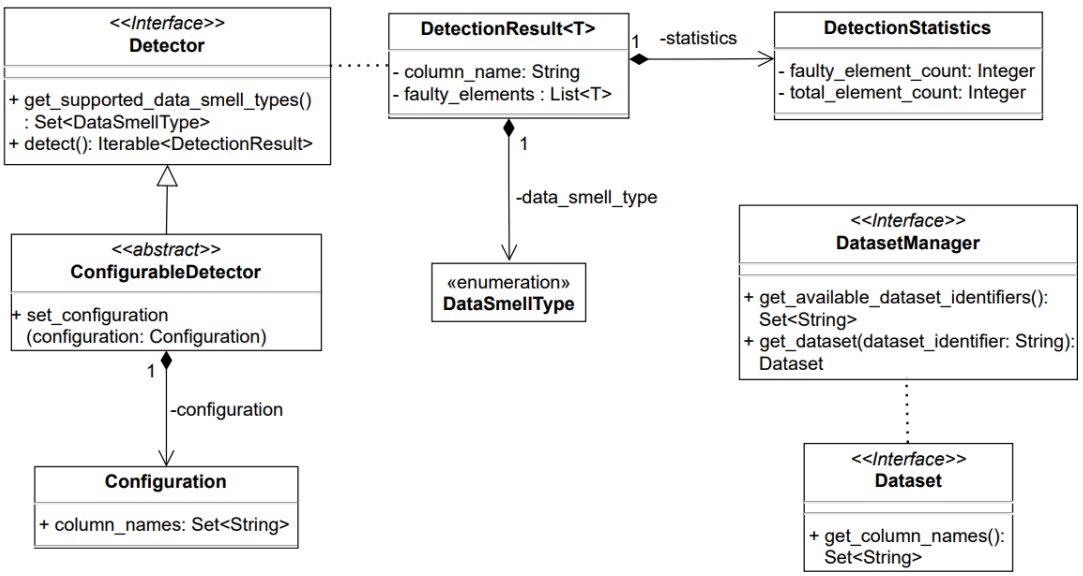
Package

Il sistema è composto da quattro package differenti, che sono:

- Package **core**: Consente di astrarre il processo di rilevamento dei data smell tramite la classe Detector.
- Package **detectors.great_expectations**: Contiene la classe GreatExpectationsDetector. Ci riferiremo a questo package, come package detector.
- Package **detectors.great_expectations.expectations**: Consente di implementare all'interno del sistema i detector specifici dei data smell. Ci riferiremo a questo package, come package expectations.
- Package **web_application**: Consente di implementare la web application, che utilizza la libreria, per far visualizzare i risultati delle detection su un'interfaccia grafica.

Package core

Per descrivere il package core è stato ideato un class diagram che riassume tutti gli oggetti e le dipendenze di questa libreria.



Per comprendere al meglio gli oggetti del sistema, di seguito viene riportata una descrizione degli oggetti in tabelle.

Nome classe	Descrizione	Attributi
Dataset	Interfaccia che rappresenta il dataset sul quale verrà poi effettuata la data smell detection.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_column_names	Consente di prelevare i nomi delle colonne del dataset.	void -> Set<String>

Nome classe	Descrizione	Attributi
DatasetManager	Interfaccia che fornisce metodi per la gestione e la preparazione all'uso del dataset.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_available_dataset_identifiers	Consente di prelevare l'insieme di identificatori di dataset.	void -> Set<String>
get_dataset	Consente di prelevare il dataset specifico passando un identificatore.	dataset_identifier:String -> Dataset

Nome classe	Descrizione	Attributi
DetectionResult	Classe che fornisce un elenco di risultati del processo di detection di data smell.	column_name: String faulty_elements : List<T> statistics: DetectionStatistics data_smell_type: DataSmellType
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

Nome classe	Descrizione	Attributi
DataSmellType	Enum che contiene i data smell che i detector possono trovare nei dataset.	Per ogni tipologia di smell è definito un elemento della Enum
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

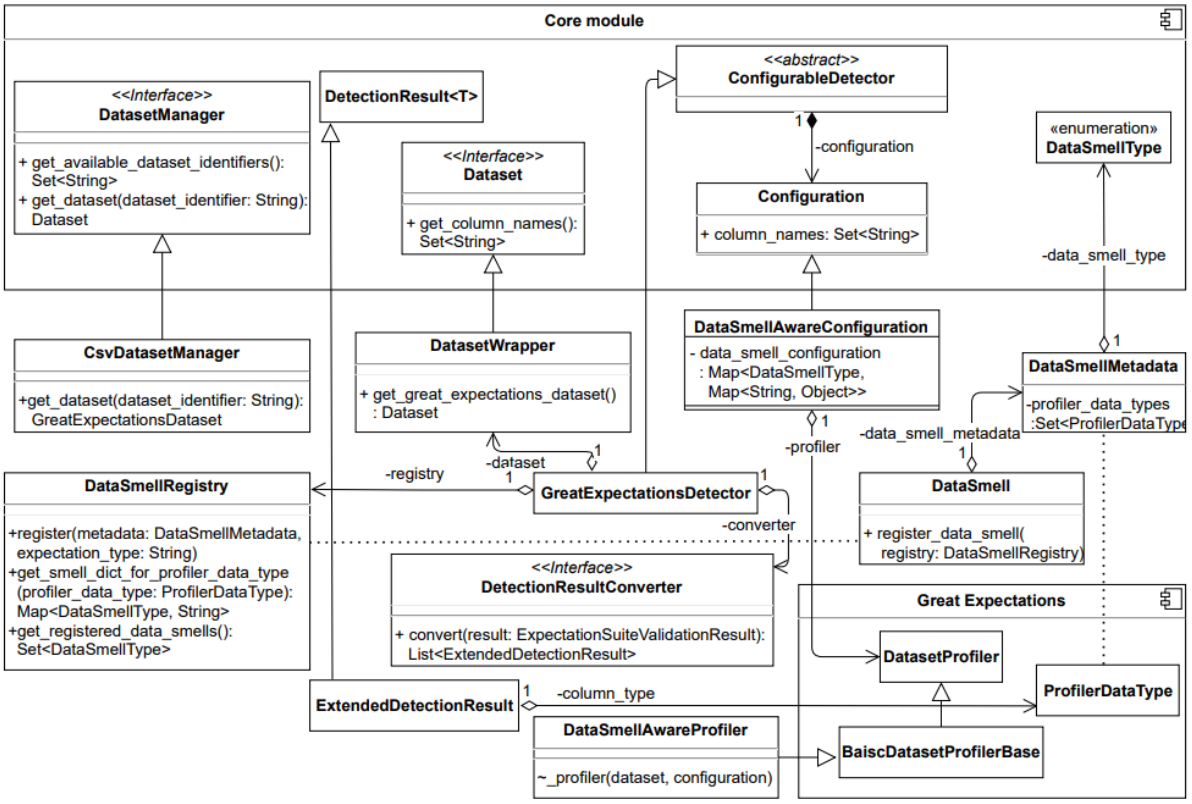
Nome classe	Descrizione	Attributi
Detector	Interfaccia che consente di definire i data smell in un particolare dataset.	
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_supported_data_smell_types	Ritorna un insieme di tipi di data smell che il detector può trovare nei dataset.	void -> Set<DataSmellType>
detect	Esegue la detection e restituisce i data smell trovati sotto forma di risultati di detection.	void -> Iterable<DetectionResult>

Nome classe	Descrizione	Attributi
Configurator	Classe astratta che consente di definire i parametri di configurazione per una detection.	column_names : Set[String]
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

Nome classe	Descrizione	Attributi
ConfigurableDetector	Classe astratta che estende la classe Detector per definire un Detector configurabile.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_configuration	Restituisce la configurazione usata per il detector.	void -> Configuration
set_configuration	Consente di definire una nuova configurazione per il detector.	configuration:Configuration-> void

Package detector

Anche qui è stato ideato un class diagram che descrive la composizione degli oggetti del package. Inoltre, viene descritto anche come gli oggetti del package detector estendono gli oggetti del package core e comunicano con alcuni di essi. Da notare che vi è un package esterno, denominato Great Expectations, che è colui che si occupa di accedere agli oggetti utilizzati per svolgere le attività richieste dalle classi a Great Expectations.



Per comprendere al meglio gli oggetti del sistema, di seguito viene riportata una descrizione degli oggetti in tabella (gli oggetti che già sono stati descritti della libreria core non verranno riportati in questa tabella).

Nome classe	Descrizione	Attributi
DatasetWrapper	Classe che wrappa una istanza di dataset di Great Expectations e estende la classe Dataset. Si trova nel modulo dataset.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_great_expectations_dataset	Restituisce il dataset wrappato.	void -> Dataset

Nome classe	Descrizione	Attributi
CsvDatasetManager (nel codice si chiama FileBasedDatasetManager)	Classe che estende la classe DatasetManager e che consente di creare istanze di DatasetWrapper da file CSV. Si trova nel modulo dataset.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
get_dataset	Consente di ottenere il dataset importato.	dataset_identifier:String -> GreatExpectationsDataset (anche qui nel codice ritorna in realtà il DatasetWrapper)

Nome classe	Descrizione	Attributi
DataSmellMetadata	Classe che permette di descrivere come effettuare la detection di un data smell specifico. Si trova nel modulo datasmell.py.	data_smell_type:DataSmellType profiler_data_types:Set<ProfilerDataType>
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

Nome classe	Descrizione	Attributi
DataSmellRegistry	Classe che permette di salvare le istanze di DataSmellMetadata dei data smell implementati. Si trova nel modulo datasmell.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
register	Salva un nuovo mapping tra un data smell e la corrispondente aspettativa di Great Expectations.	metadata:DataSmellMetadata, expectation_type:String -> //
get_smell_dict_for_profiler_data_type	Permette di prelevare i data smell registrati per uno specifico ProfilerDataType.	profiler_data_type:ProfilerDataType -> Map<DataSmellType.String>
get_registered_data_smells	Permette di prelevare un insieme di tipi di data smell registrati.	-> Set<DataSmellType>

Nome classe	Descrizione	Attributi
DataSmell	Classe progettata per essere una base per le classi che implementano la detection di data smell. Si trova nel modulo datasmell.py.	data_smell_metadata:Optional<DataSmellMetadata>
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
register_data_smell	Semplifica la registrazione delle classi che implementano la detection dei data smell in un DataSmellRegistry.	registry:DataSmellRegistry -> //

Nome classe	Descrizione	Attributi
DataSmellAwareConfiguration	Classe di configurazione che controlla come la detection di data smell viene effettuata dalla classe GreatExpectationsDetector. Inoltre è un’estensione della classe Configuration. Si trova nel modulo detector.py.	data_smell_configuration: Map<DataSmellType, Map<String,Object>>
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

Nome classe	Descrizione	Attributi
DataSmellAwareProfiler	Classe che rappresenta un profiler basato su Great Expectations per la detection di data smell. Lo scopo principale è quello di applicare i data smell registrati in una classe DataSmellRegistry, alle colonne del corrispondente DataSmellType. Si trova nel modulo profiler.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
_profile	Consente di realizzare un ExpectationSuite (descrive i controlli per i data smell, per specifiche colonne in un dataset).	dataset, configuration -> ExpectationSuite

Nome classe	Descrizione	Attributi
ExtendedDetectionResult	Estensione specifica di Great Expectations della classe DetectionResult. Si trova nel modulo converter.py.	column_type:ProfilerDataType expectation_kwargs:Dict<str, Any>
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
//	//	//

Nome classe	Descrizione	Attributi
DetectionResultConverter	Interfaccia che consente di fornire metodi per la conversione da ExpectationSuiteValidationResult a una lista di ExtendedDetectionResult. Si trova nel modulo converter.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
convert	Effettua la conversione da ExpectationSuiteValidationResult a istanze di ExtendedDetectionResult.	result:ExpectationSuiteValidationResult -> List<ExtendedDetectionResult>

Nome classe	Descrizione	Attributi
StandardResultConverter	Classe che implementa l'interfaccia DetectionResultConverter e dunque la conversione associata. Si trova nel modulo converter.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
convert	Implementazione del metodo dell'interfaccia DetectionResultConverter.	validation_suite_result:ExpectationSuiteValidationResult -> List<ExtendedDetectionResult>

Nome classe	Descrizione	Attributi
GreatExpectationsDetector	Sottoclasse della classe ConfigurableDetector. Implementa i metodi definiti sia in Detector che in ConfigurableDetector. Si trova nel modulo detector.py.	//
Metodi		
Nome	**Descrizione**	**Parametri e valori di ritorno**
detect	Esegue la detection e restituisce i risultati.	void -> Iterable<DetectionResult>

Documentazione dei package core e detector

La documentazione del codice dei due package può essere generata tramite l'utilizzo del tool Sphinx, attraverso i seguenti passi:

1. Assicurarsi di aver installato il tool;
2. Posizionarsi nella directory *docs* all'interno di *data_smell_detection*;
3. Eseguire il seguente comando per compilare la documentazione:

   ```
   sphinx-apidoc -o . ../datasmelldetection
   ```

4. Eseguire il seguente comando per ottenere il formato html:

   ```
   make html
   ```

5. Per visualizzare la documentazione del codice, bisogna aprire dal browser il file *index.html*, presente nella directory *_build/html*

Package expectations

In questo package sono inserite, per ogni detector implementato, (e dunque per ogni data smell rilevato) una classe specifica che consente di avere una visione completa dei vari detector. Queste classi utilizzano il pattern di sviluppo di Great Expectations, che consente non solo di implementare le expectations personalizzate, ma anche di testarle, utilizzando una struttura già implementata dalla libreria. Inoltre, ad ogni detector è associato un parametro *mostly*, che consente di definire il threshold specifico, per definire che una certa colonna di un dataset contiene lo smell. Le varie classi presenti nel sistema attuale sono:

- **ExpectColumnValuesToNotContainCasingSmell**: Classe che consente di implementare il detector per il Casing Smell. Oltre al parametro *mostly*, è stato inserito un ulteriore parametro, denominato *same_case_wordcount_threshold*, che permette di controllare quante parole devono essere contenute in una stringa, per essere considerate come casing smell.
- **ExpectColumnValuesToNotContainDuplicatedValueSmell**: Classe che consente di implementare il detector per il Duplicated Value Smell. Utilizza la classe ExpectColumnValuesToBeUnique già implementata da Great Expectations.
- **ExpectColumnValuesToNotContainExtremeValueSmell**: Classe che consente di implementare il detector per l'Extreme Value Smell. Oltre al parametro *mostly*, è stato inserito un ulteriore parametro, denominato *threshold*, che permette di capire quando viene identificato un valore come outlier dal detector, fissando una distanza specifica oltre la quale un numero viene considerato un outlier. Il valore di questo parametro deve essere positivo e di default è 3.
- **ExpectColumnValuesToNotContainFloatingPointNumberAsString Smell**: Classe che consente di implementare il detector per il FloatingPointNumberAsStringSmell.
- **ExpectColumnValuesToNotContainIntegerAsFloatingPointNumberSmell**: Classe che consente di implementare il detector per l'Integer As Floating Point Number Smell. Oltre al parametro *mostly*, è stato utilizzato un ulteriore parametro, denominato *epsilon*, che consente di definire il limite per la differenza assoluta tra il floating point e il suo intero più vicino. Il valore di default di questo parametro è 1e-6.
- **ExpectColumnValuesToNotContainIntegerAsStringSmell**: Classe che consente di implementare il detector per l'Integer As String Smell.

- **ExpectColumnValuesToNotContainLongDataValueSmell**: Classe che consente di implementare il detector per il Long Data Value Smell. Oltre al parametro *mostly*, è stato inserito un ulteriore parametro, denominato *length_threshold*, che permette di determinare il numero di caratteri, per i quali una parola viene considerata “lunga”. Questo parametro deve essere un numero maggiore di 0.
- **ExpectColumnValuesToNotContainMissingValueSmell**: Classe che consente di implementare il detector per il Missing Value Smell.
- **ExpectColumnValuesToNotContainSuspectSignSmell**: Classe che consente di implementare il detector per il Suspect Sign Smell. Oltre al parametro *mostly*, è stato inserito un ulteriore parametro, denominato *percentile_threshold*, che controlla quali quantili sono calcolati. I quantili sono usati per determinare se la maggioranza dei valori in una colonna è positiva o negativa. Il valore di questo parametro deve essere compreso tra 0 e 1.

Tutti questi detector devono essere inseriti all’interno del DataSmellRegistry, con il metodo *register_data_smell()* (descritto nella sezione del package detector). Inoltre, ogni classe avrà *data_smell_metadata* come variabile d’istanza, che consente di settare il tipo di data smell e il tipo di dato delle colonne che verranno analizzate dal detector.

Package web_application

Questo package ha permesso di implementare il front-end. All'interno di esso sono definite le implementazioni riguardanti sia l'autenticazione, sia la logica applicativa che consente di accedere alle librerie interne del sistema le quali effettuano la detection di data smell. Di seguito è riportata la struttura del package con indicate le componenti principali:

- **web_application/**
    - **argon-dashboard-django/**
        - core/ # gestisce i file statici della webapp
            - settings.py # file di configurazione del framework
            - urls.py # definisce gli URL per tutte le pagine
            - static/ # directory che contiene i file statici (css/js)
            - templates/ # template delle pagine da renderizzare
        - app/ # gestisce la logica applicativa
            - urls.py # definisce gli URL per la logica applicativa
            - views.py # mostra le pagine HTML agli utenti
        - authentication/ # gestisce la fase di registrazione
            - urls.py # definisce gli URL per la registrazione
            - views.py # mostra le pagine HTML agli utenti
            - forms.py # definisce i form di autenticazione
        - requirements.txt # file di configurazione per le dipendenze
        - manage.py # script di avvio della webapp

A partire da questo package è stato possibile derivare lo schema per la gestione dei dati persistenti come descritto nella sezione successiva.

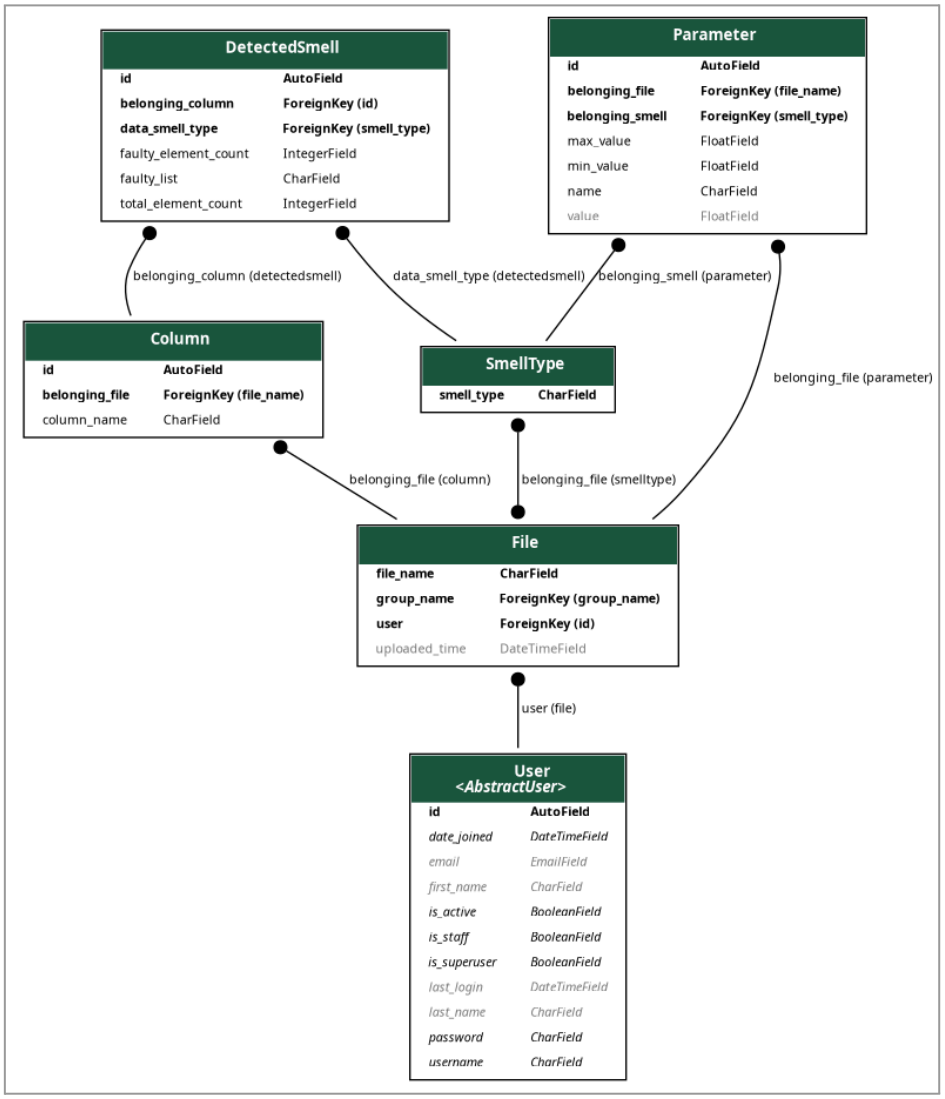
Gestione dei dati persistenti

Per la gestione dei dati persistenti è stato utilizzato un DBMS relazionale denominato SQLite3. L'utilizzo di esso è stato dovuto da diversi fattori:

- È molto veloce;
- È compatto;
- Ha transazioni ACID;
- Semplice da utilizzare;
- È in grado di interpretare stringhe SQL;

Questo tipo di DBMS viene fornito come libreria da Django, che è il framework utilizzato per lo sviluppo della web application. Infatti, la scelta è stata influenzata dal linguaggio di programmazione utilizzato, cioè Python, e dalla facilità di integrazione di questo tipo di DBMS con esso e il framework della web application.

Nella figura seguente vi è rappresentato lo schema del database.



Generazione dello schema dei dati

Il framework Django mette a disposizione un insieme di estensioni utili per facilitare la gestione del sistema. Come visto nella precedente sezione è possibile generare lo schema dei dati, grazie all'estensione Graph models, attraverso i seguenti passi:

1. Assicurarsi di aver installato le django-extensions tramite il comando:

   ```
   pip install django-extensions
   ```

2. All'interno della sezione *INSTALLED_APPS* del file *web-application/core/settings.py* inserire la dicitura *django_extensions*
3. Assicurarsi di aver installato graphviz
4. Installare pygraphviz con il comando:

   ```
   pip install pygraphviz
   ```

5. Posizionarsi nella directory dove si trova *manage.py*:

   ```
   cd web_application/argon-dashboard-django
   ```

6. Per generare lo schema dei dati, eseguire il seguente comando specificando i nomi delle tabelle del database:

   ```
   python manage.py graph_models -a -I DetectedSmell,
   Parameter, Column, SmellType, File, User -o
   schema.png
   ```

Analisi dei Requisiti

Requisiti funzionali

- **RF_01 - Registrazione al sistema:** L'utente ospite deve poter registrarsi al sistema inserendo le proprie informazioni;
- **RF_02 - Autenticazione al sistema:** L'utente ospite deve poter autenticarsi con le proprie credenziali;
- **RF_03 - Modifica informazioni utente:** L'utente registrato deve poter modificare le proprie informazioni;
- **RF_04 - Logout dal sistema:** L'utente registrato deve poter effettuare il logout dal sistema;
- **RF_05 - Caricamento del dataset:** L'utente ospite e l'utente registrato devono poter caricare un dataset all'interno del sistema;
- **RF_06 - Cambio del dataset:** L'utente ospite e l'utente registrato devono poter cambiare il dataset caricato;
- **RF_07 - Detection di data smell:** L'utente ospite e l'utente registrato devono poter effettuare una detection personalizzata dei data smell all'interno di un dataset appena caricato;
- **RF_08 - Visualizzazione risultati detection:** L'utente ospite e l'utente registrato devono poter visualizzare i risultati della detection appena effettuata;
- **RF_09 - Salvataggio risultati detection:** L'utente registrato deve poter salvare i risultati ottenuti dalla detection effettuata per visualizzarli in un secondo momento;
- **RF_10 - Eliminazione risultati detection:** L'utente registrato deve poter eliminare i risultati ottenuti da una detection, che sono stati precedentemente salvati;
- **RF_11 - Visualizzazione documentazione smell:** L'utente ospite e l'utente registrato devono poter visualizzare la documentazione dei vari tipi di data smell;

Requisiti non funzionali

Il sistema esistente ha uno scopo molto specifico e un'utenza ristretta che non corrisponde a quella di un tradizionale cliente, ma a un'utenza interna ad un team di sviluppo. Non conoscendo le specifiche esigenze richieste, non sappiamo quali sono i requisiti non funzionali che devono essere prioritizzati. A seguito della nostra analisi abbiamo comunque individuato alcuni aspetti che potrebbero essere oggetto di migliorie future:

- **Usabilità**: Il sistema spesso non fornisce messaggi chiari all'utente di ciò che sta succedendo nelle fasi di detection e le label di alcuni pulsanti non sono molto chiare rispetto all'azione che si andrà ad effettuare;
- **Sicurezza**: I criteri di scelta della password risultano essere poco restrittivi, ciò può comportare una facile manomissione del sistema da attacchi esterni;

Pseudo-requisiti

L'applicativo è stato sviluppato in Python come linguaggio di programmazione e per la web application è stato utilizzato django con il database di sistema mantenuto sul DBMS relazionale SQLite.

Modello dei casi d'uso

Sono stati identificati due attori principali all'interno del sistema:

- Utente ospite
- Utente registrato

Essi hanno quasi le stesse funzionalità in comune, la differenza fondamentale è che l'utente registrato può salvare i risultati delle detection.

UC_01: Upload file

Attore: Utente ospite, Utente registrato

Entry condition: L'utente sta visualizzando la home page.

Flusso di eventi:

1. L'utente seleziona il campo "select file" dove dovrà inserire il dataset sul quale vuole effettuare l'analisi;
2. Il sistema fa visualizzare la schermata di selezione del file;
3. L'utente seleziona il file, che deve essere di tipo .csv;
4. Il sistema informa l'utente di aver ricevuto correttamente il file;
5. L'utente fa l'upload del file nel sistema;
6. Il sistema informa l'utente che l'upload è avvenuto con successo;
7. L'utente può cambiare il file oppure proseguire la detection.

Exit condition: L'utente ha caricato con successo il file.

Flussi alternativi/eccezioni:

- Se al passo 5 l'utente tenta di fare l'upload di un file con il formato errato, verrà eseguito il caso d'uso 01.01.
- Se al passo 7 l'utente decide di cambiare file, verrà eseguito il caso d'uso 01.02.

UC_01.01: Formato file errato

Attore: Utente ospite, Utente registrato

Entry condition: L'utente ha provato ad inserire il file sul quale verrà effettuata la detection ma non era del formato corretto.

Flusso di eventi:

1. Il sistema mostra un messaggio di errore "unsupported file type";
2. Il sistema ripropone la schermata di selezione del file.

Exit condition: L'utente carica il file del formato corretto.

UC_01.02: Cambio file

Attore: Utente ospite, Utente registrato

Entry condition: L'utente accede alla funzionalità di cambio file.

Flusso di eventi:

1. L'utente richiede di cambiare file;
2. Il sistema rimuove il file precedentemente caricato e ripropone la schermata di caricamento del file.

Exit condition: L'utente carica il nuovo file.

UC_02: Data smell detection

Attore: Utente ospite, Utente registrato

Entry condition: L'utente sta visualizzando la pagina con il dataset caricato con successo.

Flusso di eventi:

1. L'utente procede con la fase successiva di personalizzazione;
2. Il sistema fornisce una serie di parametri e scelte che permettono di personalizzare la detection;
3. L'utente seleziona i parametri più adatti al tipo di detection che vuole effettuare e invia le informazioni al sistema;
4. Il sistema salva le informazioni e informa l'utente di aver salvato con successo le informazioni;
5. L'utente procede con la fase successiva;
6. Il sistema mostra i risultati della detection sul dataset, divisi per colonne differenti, accessibili nuovamente dall'utente nella sezione "saved results".

Exit condition: L'utente ha completato la detection sul dataset e ha ricevuto i risultati.

Flussi alternativi/eccezioni:

- Se al passo 3 l'utente decide di non selezionare alcun parametro, verrà eseguito il caso d'uso 02.01.
- Se al passo 6 l'utente registrato decide di eliminare il risultato, verrà eseguito il caso d'uso 02.02.

UC_02.01: Parametri non selezionati

Attore: Utente ospite, Utente registrato

Entry condition: L'utente non seleziona alcun data smell da individuare oppure nessuna colonna del dataset sulla quale effettuare la detection.

Flusso di eventi:

1. Il sistema fornisce un messaggio di errore "Select smells and columns";
2. Il sistema ripropone la schermata di personalizzazione della detection.

Exit condition: L'utente seleziona almeno uno smell e almeno una colonna.

UC_02.02: Cancellazione risultati

Attore: Utente registrato

Entry condition: L'utente registrato decide di eliminare i risultati.

Flusso di eventi:

1. Il sistema chiede all'utente la conferma per eliminare i risultati;
2. L'utente da il suo consenso;
3. Il sistema elimina i risultati, fornendo a schermo un messaggio di conferma.

Exit condition: I risultati vengono eliminati dall'account dell'utente registrato.

UC_03: Visualizzazione risultati salvati

Attore: Utente registrato

Entry condition: L'utente visualizza l'home page.

Flusso di eventi:

1. L'utente seleziona il campo "saved results" per accedere ai risultati;
2. Il sistema fornisce la schermata di visualizzazione dei risultati.

Exit condition: L'utente visualizza la pagina con i risultati salvati.

Flussi alternativi/eccezioni:

- Se al passo 2 il sistema non riscontra alcun risultato, verrà eseguito il caso d'uso 03.01.

UC_03.01: Nessun risultato salvato

Attore: Utente registrato

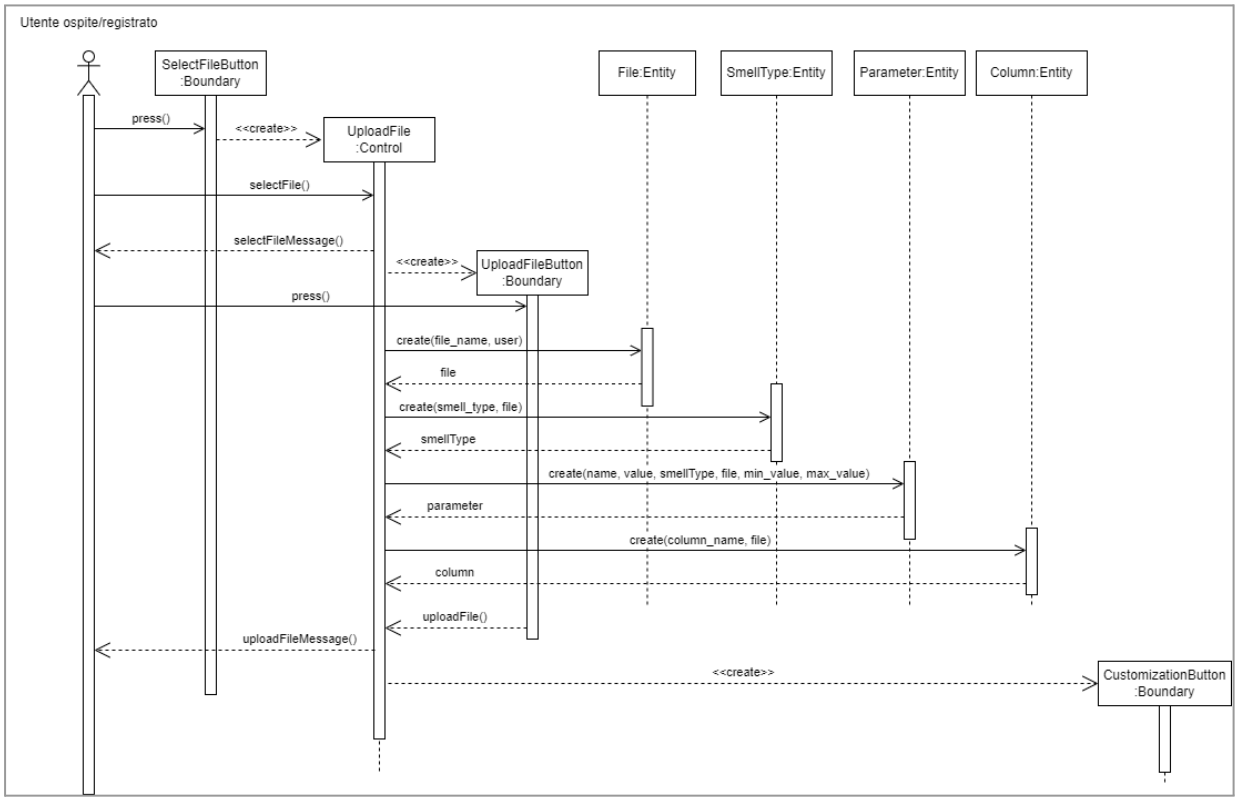
Entry condition: Il sistema non riscontra alcun risultato salvato.

Flusso di eventi: Il sistema restituisce una schermata con il messaggio "There aren't any saved detection results".

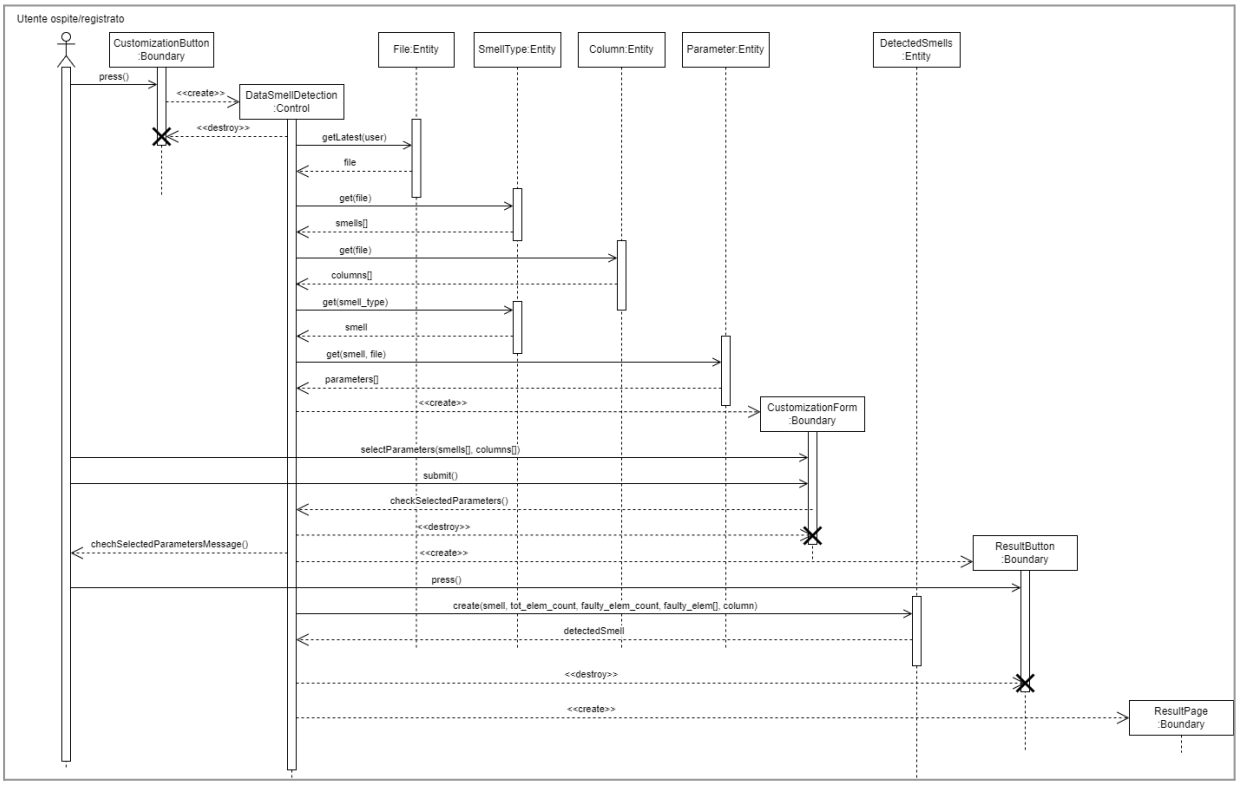
Exit condition: L'utente visualizza il messaggio.

Sequence diagram

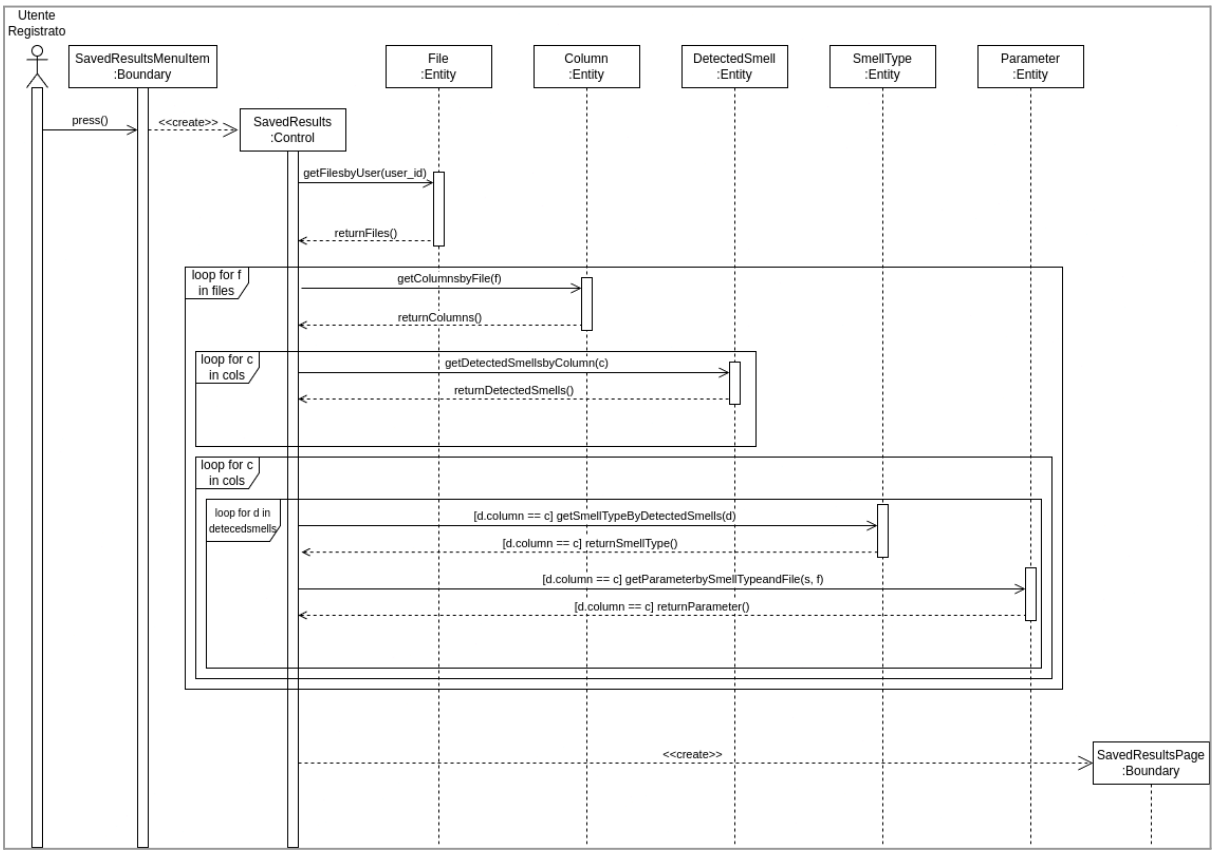
SD_01: Upload file



SD_02: Data smell detection



SD_03: Visualizzazione risultati salvati



Change Requests

Change request 1

Change request id: CR_01

Titolo: Estensione della suite di detection di data smell.

Descrizione: La seguente change request prevede la realizzazione di almeno una nuova detection di data smell, in modo tale da estendere la suite di detection che mette a disposizione il tool.

Tipologia di manutenzione: additiva/perfettiva

Requisiti coinvolti: RF_07

Stato: presentata/sottomessa

Change request 2

Change request id: CR_02

Titolo: Funzionalità di calcolo delle data quality dimensions metrics.

Descrizione: La seguente change request prevede la realizzazione di un meccanismo di calcolo, rispetto ai risultati ottenuti dall'analisi del tool, di un sottoinsieme di data quality dimensions (metriche che consentono di riuscire a definire la qualità dei dataset che analizza il tool DSD):

- **Completeness**: La percentuale di dati memorizzati rispetto al potenziale di "100% completo";
- **Uniqueness**: Nulla verrà registrato più di una volta in base a come viene identificato;
- **Timeliness**: Il grado di rappresentazione della realtà da parte dei dati nel momento richiesto;
- **Validity**: I dati sono validi se sono conformi alla sintassi (formato, tipo, intervallo) della loro definizione;
- **Accuracy**: Il grado in cui i dati descrivono correttamente l'oggetto o l'evento del "mondo reale" che viene descritto;
- **Consistency**: L'assenza di differenze, quando si confrontano due o più rappresentazioni di un elemento con una definizione.

Tipologia di manutenzione: additiva/perfettiva

Requisiti coinvolti: //

Stato: presentata/sottomessa

Change request 3

Change request id: CR_03

Titolo: Estensione del meccanismo di reporting.

Descrizione: La seguente change request prevede l’estensione dell’attuale meccanismo di reporting, aggiungendo la possibilità di visualizzare a schermo i risultati ottenuti dall’analisi di un dataset e mostrare nel tempo le differenze di risultati, tra differenti versioni di uno stesso dataset, dei relativi smell individuati e metriche/misure calcolate.

Tipologia di manutenzione: additiva/perfettiva

Requisiti coinvolti: RF_09

Stato: presentata/sottomessa

Impact Analysis

Attraverso le CR precedentemente formulate, è possibile individuare l'insieme iniziale dei moduli relativi all'implementazione del sistema che potrebbero essere oggetto di impatto, a favore delle modifiche richieste al sistema. Nella sezione *Selezione Change Request*, è stata effettuata anche un'analisi di fattibilità sulla realizzazione delle tre CR. In questa sezione saranno invece individuati tutti i moduli coinvolti per le modifiche da effettuare. Non avendo alcuna documentazione per le fasi di raccolta e analisi requisiti, saranno effettuate delle considerazioni sull'impatto a partire dal class diagram del Package detector.

Impatto di CR_01

Per questa change request sarà necessario aggiungere nuove istanze di detection, in particolare bisogna aggiungere dei nuovi moduli nel package ***datasmelldetection/detectors/great_expectations/expectations***. Per una corretta visualizzazione della scelta dei nuovi smell nella Customization Page (***customization.html***), sarà necessario modificare i file ***presettings.json*** e ***smells.json*** presenti in ***web_application/argon-dashboard-django/app,*** in questo modo l'utente avrà la possibilità di scegliere i nuovi smell e di decidere al meglio i parametri da utilizzare. Le nuove detection saranno utilizzate dal detector. All'interno del modulo ***datasmelldetection/detectors/great_expectations/detector.py*** è stata implementata la classe ***GreatExpectationsDetector*** che rappresenta la funzionalità principale del progetto. Per far si che una detection vada a buon fine, il detector ha bisogno di istanziare le seguenti classi, alcune delle quali sono direttamente parte di questo progetto:

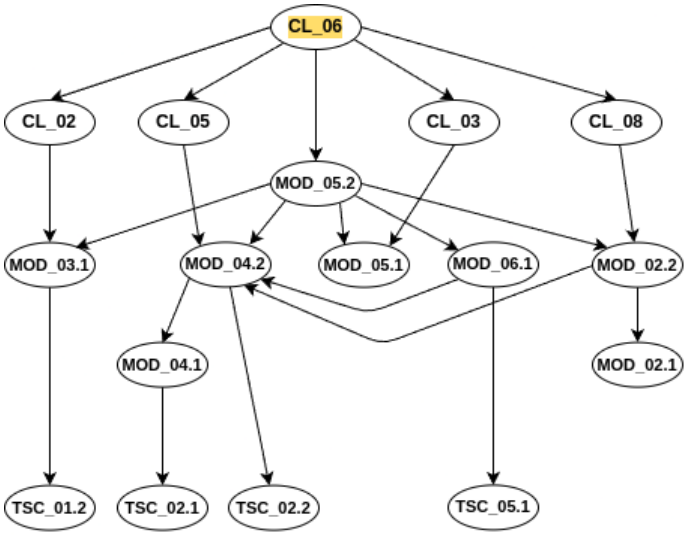
- ***DataContext:*** Classe che imposta il contesto necessario per il corretto funzionamento della libreria di terze parti GreatExpectations;
- ***DatasetWrapper:*** Classe che "wrappa" il dataset da analizzare;
- ***DataSmellRegistry:*** Classe che contiene il registro delle detection di smell utilizzabili;
- ***DatasetProfiler:*** Classe che associa le colonne del dataset con la relativa detection contenuta nel DataSmellRegistry;
- ***DetectionResultConverter:*** Classe che si occupa di formattare i risultati della detection in un formato conveniente per la webapp;
- ***DataSmellAwareConfiguration:*** Classe che memorizza la configurazione personalizzata da utilizzare per effettuare le detection.

Per l'introduzione delle nuove detection di smell, potrebbe quindi essere necessario effettuare delle modifiche alle classi descritte in precedenza.

Sulla base di queste informazioni, a partire dal class diagram che abbiamo a disposizione, per una maggiore tracciabilità sono stati realizzati il grafo diretto e derivate le matrici, evidenziando la classe ***GreatExpectationsDetector***. Inoltre è stato realizzato il Glossario acronimi per la tracciabilità degli artefatti. Dunque, utilizzando la terminologia definita dal glossario, avremo il SIS:

Starting Impact Set = {CL_02, CL_03, CL_05, CL_06, CL_08}.

Grafo diretto per CR_01



Matrice di connettività per CR_01

	CL_06	CL_02	MOD_03.1	TSC_01.2	CL_05	MOD_04.2	MOD_04.1	TSC_02.1	TSC_02.2	CL_03	MOD_05.1	CL_08	MOD_02.2	MOD_02.1	MOD_05.2	MOD_06.1	TSC_05.1
CL_06		X			X					X		X			X		
CL_02			X														
MOD_03.1				X													
TSC_01.2																	
CL_05						X											
MOD_04.2							X		X								
MOD_04.1								X									
TSC_02.1																	
TSC_02.2																	
CL_03											X						
MOD_05.1																	
CL_08													X				
MOD_02.2						X								X			
MOD_02.1																	
MOD_05.2			X			X					X		X			X	
MOD_06.1																	X
TSC_05.1																	

Matrice di raggiungibilità per CR_01

	CL_06	CL_02	MOD_03.1	TSC_01.2	CL_05	MOD_04.2	MOD_04.1	TSC_02.1	TSC_02.2	CL_03	MOD_05.1	CL_08	MOD_02.2	MOD_02.1	MOD_05.2	MOD_06.1	TSC_05.1
CL_06		1	2	3	1	2	3	4	3	1	2	1	2	3	1	2	3
CL_02			1	2													
MOD_03.1				1													
TSC_01.2																	
CL_05						1	2	3	2								
MOD_04.2							1	2	1								
MOD_04.1								1									
TSC_02.1																	
TSC_02.2																	
CL_03											1						
MOD_05.1																	
CL_08													1	2			
MOD_02.2						1	2	3	2					1			
MOD_02.1																	
MOD_05.2			1	2		1	2	3	2		1		1	2		1	2
MOD_06.1																	1
TSC_05.1																	

Attraverso un approccio Distance based, si selezionano tutti gli artefatti entro tre archi di distanza a partire dalla classe principale CL_06, ottenendo il CIS:

- **Candidate Impact Set** = {CL_02, MOD_03.1, TSC_01.2, CL_05, MOD_04.2, MOD_04.1, TSC_02.2, CL_03, MOD_05.1, CL_08, MOD_02.2, MOD_02.1, MOD_05.2, MOD_06.1, TSC_05.1}.

Impatto di CR_02

La funzionalità descritta da questa change request non è presente nel sistema, non è possibile mappare nessun modulo esistente che possa essere modificato. In questo caso bisogna aggiungere un nuovo modulo che consenta di calcolare le data quality dimensions, i cui risultati saranno mostrati all'interno del nuovo meccanismo di reporting, che sarà anch'esso oggetto di modifica nella successiva change request. Questo modulo sarà posizionato nella cartella ***app/*** all'interno del package ***web_application***.

Impatto di CR_03

Per questa change request sarà necessario migliorare la presentazione dei risultati, in particolare bisogna modificare ciò che sarà visualizzato nella Saved results page *(**saved.html**)* ed eventualmente introdurre delle nuove pagine. Tutte le pagine dell'applicazione sono contenute nel package ***web_application/argon-dashboard-django/core***, alla subfolder ***templates/***. Ogni volta che si aggiunge una nuova pagina al progetto, è necessario modificare i seguenti moduli:

- ***web_application/argon-dashboard-django/app/views.py:*** Questo modulo si occupa di elaborare le richieste inviate alla webapp per fornire una risposta all'utente.
- ***web_application/argon-dashboard-django/app/urls.py:*** Questo modulo si occupa di recuperare le richieste HTTP e di individuare un match tra l'URL richiesto e gli URL memorizzati nella webapp. Se c'è un match, viene invocata la funzione associata che è presente in *views.py,* la quale si occuperà di eseguire tutta la logica necessaria, che permetterà di fornire una risposta per mostrare la corretta visualizzazione della pagina richiesta e del relativo contesto.

Nel caso sia necessario introdurre nuove entità nel database, è necessario modificare il seguente modulo:

- ***web_application/argond-dashboard-django/app/models.py:*** Questo modulo si occupa di creare le tabelle del database in SQLite tramite tecniche di ORM.

Glossario acronimi per la tracciabilità degli artefatti

Design

- **CL_01 - Classe *CSVDatasetManager* del class diagram**
- **CL_02 - Classe *DatasetWrapper del class diagram***
- **CL_03 - Classe *DataSmellAwareConfiguration* del class diagram**
- **CL_04 - Classe *DataSmellMetadata* del class diagram**
- **CL_05 - Classe *DataSmellRegistry* del class diagram**
- **CL_06 - Classe *GreatExpectationsDetector* del class diagram**
- **CL_07 - Classe *DataSmell* del class diagram**
- **CL_08 - Classe *DetectionResultConverter* del class diagram**
- **CL_09 - Classe *ExtendedDetectionResult* del class diagram**
- **CL_10 - Classe DataSmellAwareProfiler del class diagram**

Codice

PKG_01 - Package *datasmelldetection/detectors/great_expectations*

- **MOD_01 - modulo *context.py*:**
    - **MOD_01.1 - class *GreatExpectationsContextBuilder*,** implementazione del pattern Builder per facilitare la creazione di istanze di *DataContext*;
- **MOD_02 - modulo *converter.py*:**
    - **MOD_02.1 - class *ExtendedDetectionResult*,** implementazione della classe *CL_09 - ExtendedDetectionResult*;
    - **MOD_02.2 - class *StandardResultConverter*,** implementazione della classe *CL_08 - DetectionResultConverter* per codificare i risultati restituiti dalla libreria GreatExpectations, nel formato definito nell' implementazione della classe *ExtendedDetectionResult*;
- **MOD_03 - modulo *dataset.py*:**
    - **MOD_03.1 - class *DatasetWrapper*,** implementazione della classe *CL_02 - DatasetWrapper*;
    - **MOD_03.2 - class *FileBasedDataSetManager*,** implementazione della classe *CL_01 - CsvDatasetManager*;

- **MOD_04 - modulo *datasmell.py*:**
  - **MOD_04.1 - class *DataSmellMetadata*,** implementazione della classe *CL_04 - DataSmellMetadata*;
  - **MOD_04.2 - class *DataSmellRegistry*,** implementazione della classe *CL_05 - DataSmellRegistry*;
  - **MOD_04.3 - class *DataSmell*,** implementazione della classe *CL_07 - DataSmell*;
- **MOD_05 - modulo *detector.py*:**
  - **MOD_05.1 - class *DataSmellAwareConfiguration*,** implementazione della classe

    CL_03 - DataSmellAwareConfiguration
  - **MOD_05.2 - class *GreatExpectationsDetector*,** implementazione della classe *CL_06 - GreatExpectationsDetector*
  - **MOD_05.3 - class *DetectorBuilder*,** implementazione del pattern builder per facilitare la creazione di istanze di *GreatExpectationsDetector*
- **MOD_06 - modulo *profiler.py*:**
  - **MOD_06.1 - class *DataSmellAwareProfiler*,** implementazione della classe *CL_10 - DataSmellAwareProfiler*

Testing

(per TSC si intende Test Suites Collection)

- **TSC_01 - modulo *test_dataset.py*:**
  - **TSC_01.1 - class *TestFileBasedDatasetManager*,** test suite che implementa i test case per *MOD_03.2 - class FileBasedManager*
  - **TSC_01.2 - class *TestDatasetWrapper*,** test suite che implementa i test case per *MOD_03.1 - DatasetWrapper*
- **TSC_02 - modulo *test_datasmell.py*:**
  - **TSC_02.1 - class *TestDataSmellMetadata*,** test suite che implementa i test case per la classe *MOD_04.1 - class DataSmellMetadata*
  - **TSC_02.2 - class *TestDataSmellRegistry*,** test suite che implementa i test case per la classe *MOD_04.2 - class DataSmellRegistry*
  - **TSC_02.3 - class *TestDataSmell*,** test suite che implementa i test case per la classe *MOD_04.3 - DataSmell*
- **TSC_03 - modulo *test_detector.py*:**
  - **TSC_03.1 - class *TestDetectorBuilder*,** test suite che implementa i test case per la classe *MOD_05.3 - DetectorBuilder*
- **TSC_04 - modulo *test_expectations.py*:**
  - **TSC_04.1 - class *TestExpectations*,** test suite che implementa i test per *PKG_02 - datasmelldetection/detectors/great_expectations/expectations*
- **TSC_05 - modulo *test_profiler.py*:**
  - **TSC_05.1 - class *TestDataSmellAwareProfile*,** test suite che implementa i test per *MOD_06.1 - class DataSmellAwareProfiler*
- **TSC_06 - modulo *test_registration.py*:**
  - **TSC_06.1 - class *TestExpectationRegistration*,** test suite che implementa i test per la registrazione delle *Expectations* del *PKG_02* in una istanza della classe *DataSmellRegistry*

Selezione Change Request

Le change request sono state prioritizzate in base all'effort che è stato determinato nel realizzare le determinate modifiche e all'importanza che hanno.
Ecco di seguito la lista che mostra l'ordinamento delle CR:

1. **CR_02 - Funzionalità di calcolo delle data quality dimensions metrics**: Questa CR è quella che tocca meno moduli all'interno del sistema, poiché si tratta semplicemente di inserire delle funzionalità di calcolo matematico sulle colonne dei dataset. L'effort richiesto dunque è molto basso e, inoltre, senza la CR_02 non è possibile avere la realizzazione completa della CR_03, poiché tra le informazioni nuove che saranno aggiunte all'interno del meccanismo di reporting, ci saranno anche queste metriche. Le metriche che si intendono calcolare sono:
    - **Completeness**
    - **Uniqueness**
    - **Validity**
2. **CR_03 - Estensione del meccanismo di reporting**: Questa CR consente di inserire all'interno del sistema un meccanismo di reporting aggiornato, che permette di avere informazioni dettagliate sul perché quel dataset soffre di quei data smell e avere una tracciabilità nel tempo di come le modifiche al dataset possano migliorarlo o peggiorarlo, in base alle metriche calcolate in precedenza. Risulta essere la funzionalità principale che si vuole introdurre e dunque non avrebbe senso posticiparla.
3. **CR_01 - Estensione della suite di detection di data smell**: Questa CR consente di estendere la suite di detection di data smell, andando ad inserire due nuove detection:
    - **Spacing**
    - **Special Character**

    Nel dettaglio, questa modifica è stata lasciata come ultima poiché, richiede una approfondita comprensione del funzionamento della libreria del detector che può richiedere un maggiore sforzo, il quale potrebbe rallentare di molto il processo.

Forward Engineering

Prima di vedere nel dettaglio il processo specifico per la realizzazione delle CR, viene fornita una breve panoramica di alcune correzioni effettuate al sistema per poter implementare le CR individuate:

- **Rimozione Utente guest**: L'utente guest una volta effettuate le analisi e visualizzati i risultati, non aveva alcun modo per interagire e cancellare i risultati ottenuti e, inoltre, per il tipo di sistema che si aveva a disposizione, l'accesso come guest non era coerente con quello che è lo scopo del tool. L'utente guest è stato rimosso, dando accesso al tool solo ad utenti registrati, che devono effettuare il login per utilizzarlo.
- **Correzione upload del file:** L'applicazione permetteva di effettuare l'upload di file .csv vuoti o che presentavano solamente l'intestazione delle colonne. Questa problematica è stata risolta.
- **Cancellazione fisica del file del dataset analizzato:** Dopo che l'utente registrato effettua un'analisi, la cancellazione dei risultati ottenuti rimuove esclusivamente le informazioni memorizzate sul database e il "riferimento logico" del file del dataset analizzato, senza però andarlo a rimuovere fisicamente dal sistema. Questa problematica è stata risolta.
- **Aggiunta gruppi per il nuovo reporting system**: Sono stati aggiunti i gruppi. Il sistema consente di inserire, al momento del caricamento del file, il nome del gruppo e di conseguenza aggiungere il file a quel gruppo specifico per effettuare analisi più dettagliate di dataset specifici. I motivi per cui si è deciso di aggiungere il concetto di gruppo sono principalmente due:
    - Confronto tra più dataset di ambito simile, per stabilire rispetto agli smell individuati e le Data Quality Dimensions calcolate, quale sia il dataset più adeguato da utilizzare per un task specifico.
    - Confronto temporale tra più dataset per verificare se una modifica effettuata al dataset ha migliorato o peggiorato il medesimo, rispetto agli smell individuati e le Data Quality Dimensions calcolate.

Analisi dei requisiti

In questa sezione, evidenzieremo i casi d'uso individuati precedentemente nel sistema, i quali hanno avuto un cambiamento dovuto dalla realizzazione delle CR.

Modello dei casi d'uso

Come detto in precedenza, ora l'attore del sistema è unico ed è l'Utente registrato.

UC_CR_01: Upload File

Attore: Utente registrato

Entry condition: L'utente sta visualizzando la home page.

Flusso di eventi:

1. L'utente seleziona il campo "select file" dove dovrà inserire il dataset sul quale vuole effettuare l'analisi;
2. Il sistema fa visualizzare la schermata di selezione del file;
3. L'utente seleziona il file, che deve essere di tipo .csv;
4. Il sistema informa l'utente di aver ricevuto correttamente il file e mostra la parte del form dove deve essere inserito il nome del gruppo di cui il dataset farà parte;
5. L'utente inserisce il nome del gruppo di cui il dataset farà parte e fa l'upload del file nel sistema;
6. Il sistema informa l'utente che l'upload è avvenuto con successo;
7. L'utente può cambiare il file oppure proseguire la detection.

Exit condition: L'utente ha caricato con successo il file.

Flussi alternativi/eccezioni:

- Se al passo 5 l'utente tenta di fare l'upload di un file con il formato errato, verrà eseguito il caso d'uso 01.01.
- Se al passo 5 l'utente tenta di fare l'upload di un file con il campo del form vuoto, verrà eseguito il caso d'uso 01.03.
- Se al passo 7 l'utente decide di cambiare file, verrà eseguito il caso d'uso 01.02.

UC_CR_01.01: Formato file errato

Attore: Utente registrato

Entry condition: L'utente ha provato ad inserire il file sul quale verrà effettuata la detection ma non era del formato corretto.

Flusso di eventi:

1. Il sistema mostra un messaggio di errore "unsupported file type";
2. Il sistema ripropone la schermata di selezione del file.

Exit condition: L'utente carica il file del formato corretto.

UC_CR_01.02: Cambio file

Attore: Utente registrato

Entry condition: L'utente accede alla funzionalità di cambio file.

Flusso di eventi:

1. L'utente richiede di cambiare file;
2. Il sistema rimuove il file precedentemente caricato e ripropone la schermata di caricamento del file.

Exit condition: L'utente carica il nuovo file.

UC_CR_01.03: Campo form gruppo vuoto

Attore: Utente registrato

Entry condition: L'utente ha provato ad inserire il file sul quale verrà effettuata la detection ma non ha inserito alcun testo all'interno del form.

Flusso di eventi:

1. Il sistema mostra un messaggio "Compila questo campo";
2. Il sistema ripropone la schermata di caricamento del file.

Exit condition: L'utente carica il nuovo file.

UC_CR_03: Visualizzazione risultati salvati

Attore: Utente registrato

Entry condition: L'utente visualizza l'home page.

Flusso di eventi:

1. L'utente seleziona il campo "saved results" per accedere ai risultati delle detection salvati;
2. Il sistema fornisce la lista dei gruppi di dataset creati;
3. L'utente accede ad un gruppo specifico;
4. Il sistema mostra le metriche per il gruppo di dataset e i dataset analizzati;
5. L'utente seleziona il dataset specifico contenuto nel gruppo;
6. Il sistema mostra i risultati specifici del dataset analizzato e le metriche calcolate.

Exit condition: L'utente visualizza la pagina con i risultati salvati di un gruppo specifico.

Flussi alternativi/eccezioni:

- Se al passo 2 il sistema non riscontra alcun risultato, verrà eseguito il caso d'uso 03.01.

UC_CR_03.01: Nessun risultato salvato

Attore: Utente registrato

Entry condition: Il sistema non riscontra alcun risultato salvato.

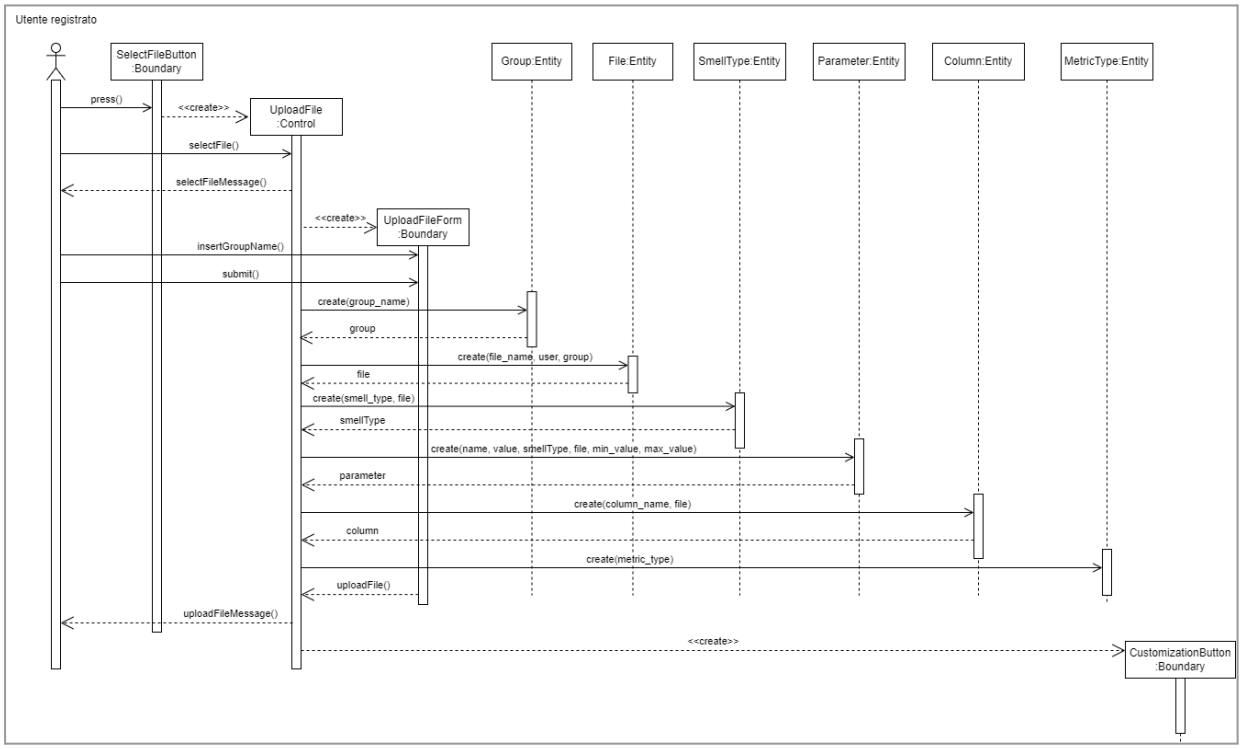
Flusso di eventi:

1. Il sistema restituisce una schermata con il messaggio "There aren't any saved detection results".

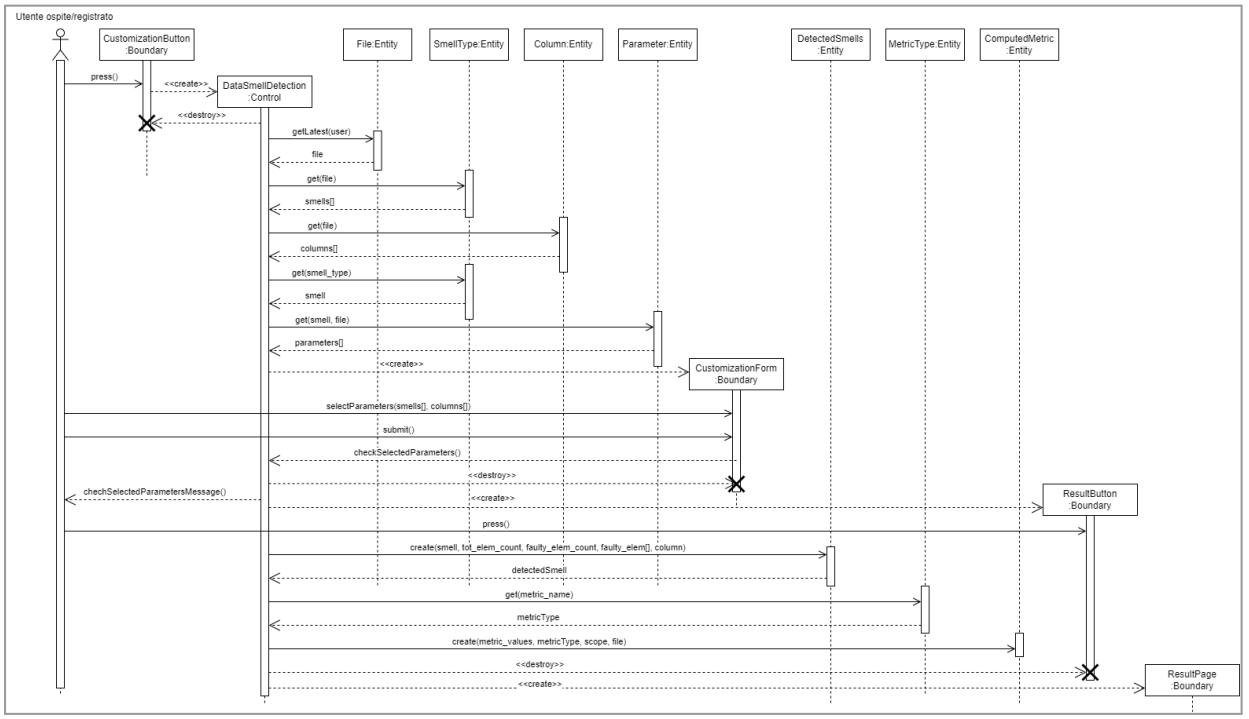
Exit condition: L'utente visualizza il messaggio.

Sequence diagram

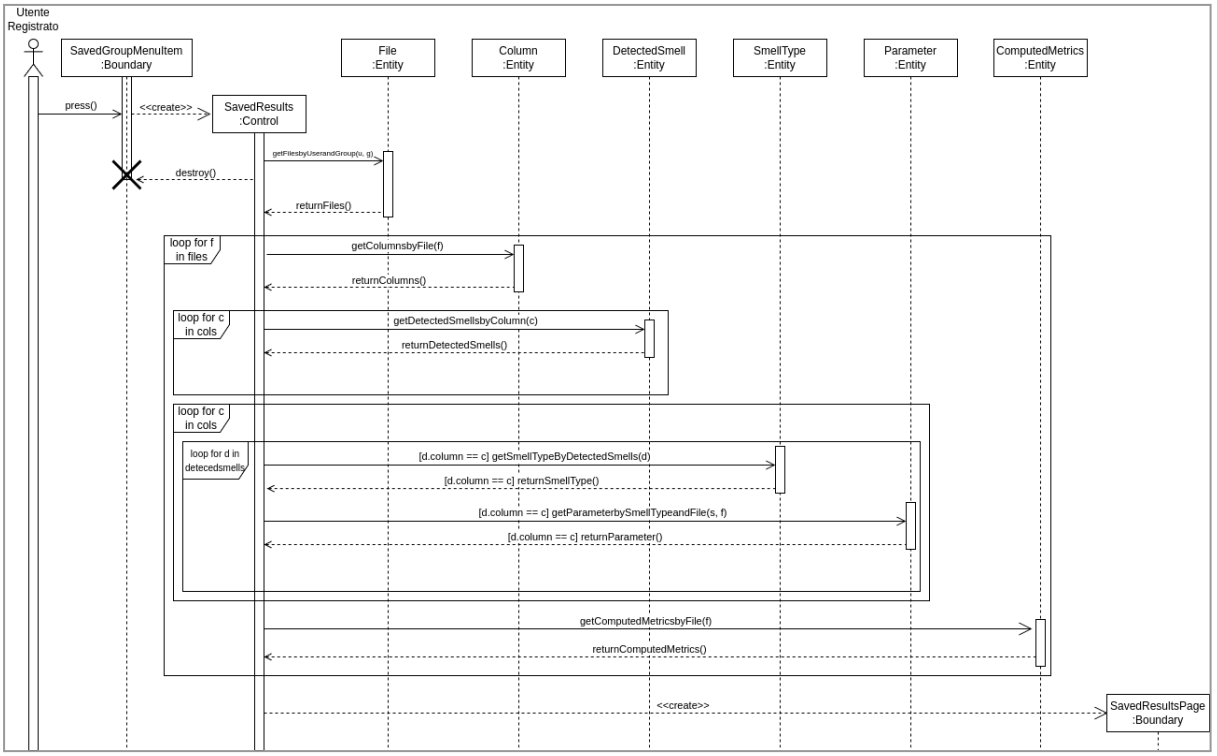
SD_CR_01: Upload file



SD_CR_02: Data Smell Detection



SD_CR_03: Visualizzazione risultati salvati



System & Object Design

Ci sono state poche modifiche effettuate a livello di classi presenti nel sistema. Infatti, la struttura è rimasta pressoché identica, le uniche aggiunte sono state effettuate nel package expectations, dove sono stati aggiunti due moduli per i due detector nuovi dei data smell che sono stati considerati, cioè Spacing e Special Character Smell.

Package expectations

Per le nuove classi, implementate per gli smell specifici, è stato utilizzato un approccio regex-based, cioè sono state utilizzate delle regex, che hanno consentito di effettuare un controllo sintattico sulle colonne del dataset e capire se rispettassero o meno la regex. Di seguito mostreremo quali sono le classi aggiunte nel package già esistente:

- **ExpectColumnValuesToNotContainSpacingSmell**: Classe che consente di implementare il detector per lo Spacing Smell. Oltre al parametro *mostly*, è stato inserito un parametro *regex*, dove è stata inserita la regex che consente di individuare dei pattern specifici nelle colonne per lo smell che deve essere rilevato. La regex utilizzata è la seguente: `'^\s|\s\s+|\s$'`.
- **ExpectColumnValuesToNotContainSpecialCharacterSmell**: Classe che consente di implementare il detector per lo Special Character Smell. Anche qui è stato aggiunto il parametro *regex*. La regex utilizzata è la seguente: `'[^a-zA-Z0-9\s]'`.

Package web_application

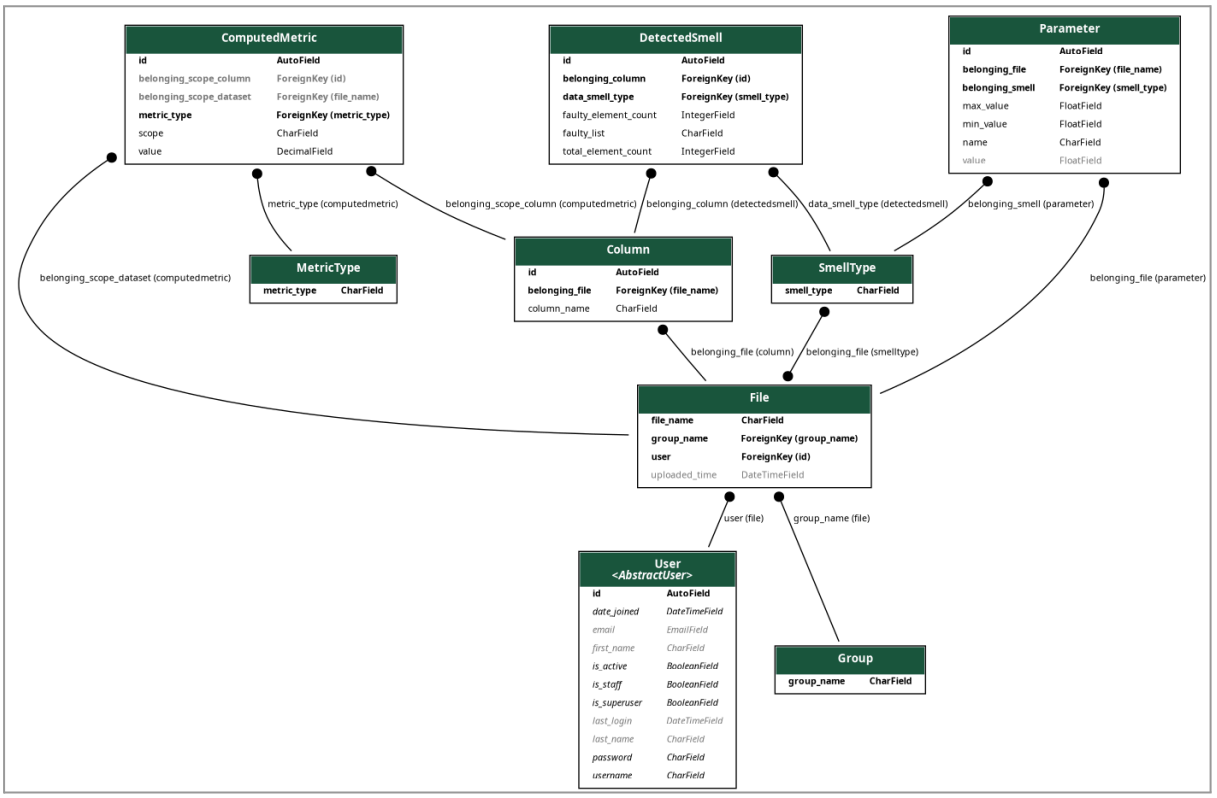
In questo package (nel dettaglio nella subfolder *app*), è stato aggiunto un nuovo modulo denominato *manage_metrics.py*, che consente di gestire le metriche. In esso, sono state inserite le seguenti funzioni:

- **compute_metric**: Funzione che consente di calcolare le metriche di completeness, uniqueness e validity di un dataset. Il seguente metodo è un metodo parametrico, quindi, a seconda del tipo di metrica che deve essere calcolato, ci sono parametri specifici che vengono settati, per consentire il corretto calcolo delle metriche.
- **save_metric**: Funzione che consente di salvare le metriche nel database, distinguendo se il tipo di metrica è calcolata a livello globale oppure a livello di colonna del dataset.
- **retrieve_metric**: Funzione che consente di prelevare le metriche calcolate precedentemente dal database, distinguendo se il tipo di metrica è calcolata a livello globale oppure a livello di colonna del dataset.

In questo modo si evita di dover (ri)calcolare le metriche ogni volta che vengono visualizzati i risultati, ma di poterle invece rapidamente recuperare dal database. Le metriche fanno parte dei risultati da mostrare all'utente rispetto al dataset analizzato. Una volta ottenute le informazioni sulla detection degli smell, le metriche saranno calcolate per una sola volta rispetto al dataset analizzato e successivamente memorizzate nel database.

Gestione dei dati persistenti

Nella figura seguente viene mostrato il database del sistema con le modifiche effettuate su di esso per rispettare i cambiamenti delle CR implementate.



I nuovi oggetti inseriti all'interno del db sono:

- **Group**: In esso è possibile salvare il nome del gruppo che viene associato ai file, per garantire la tracciabilità nel tempo di più dataset e dunque visualizzare i dati di più dataset in un singolo gruppo.
- **MetricType**: In esso vengono salvati i tipi di metriche che sono stati inseriti nel sistema (Completeness, Uniqueness, Validity). Ciò è stato fatto per garantire un salvataggio delle informazioni in maniera persistente e per garantire una modularità, che consenta a altri developer di poter inserire nuove metriche in futuro, in maniera più chiara e semplice.
- **ComputedMetric**: In esso sono salvate tutte le informazioni che riguardano il tipo di metrica calcolato su uno specifico dataset o colonna di dataset, con il suo valore.

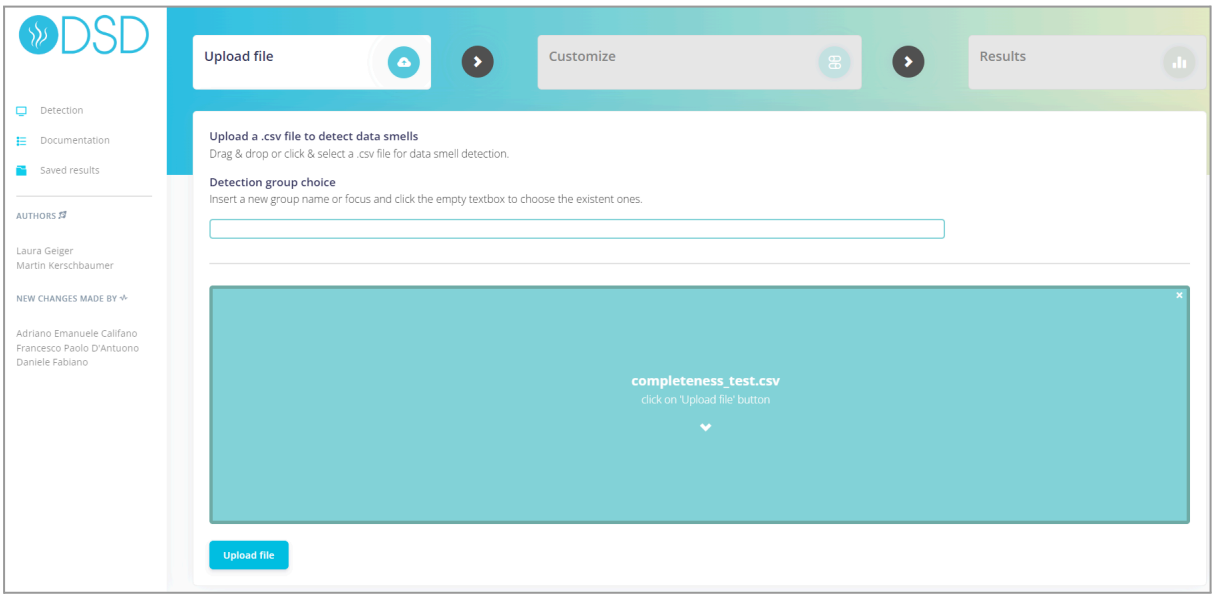
Testing

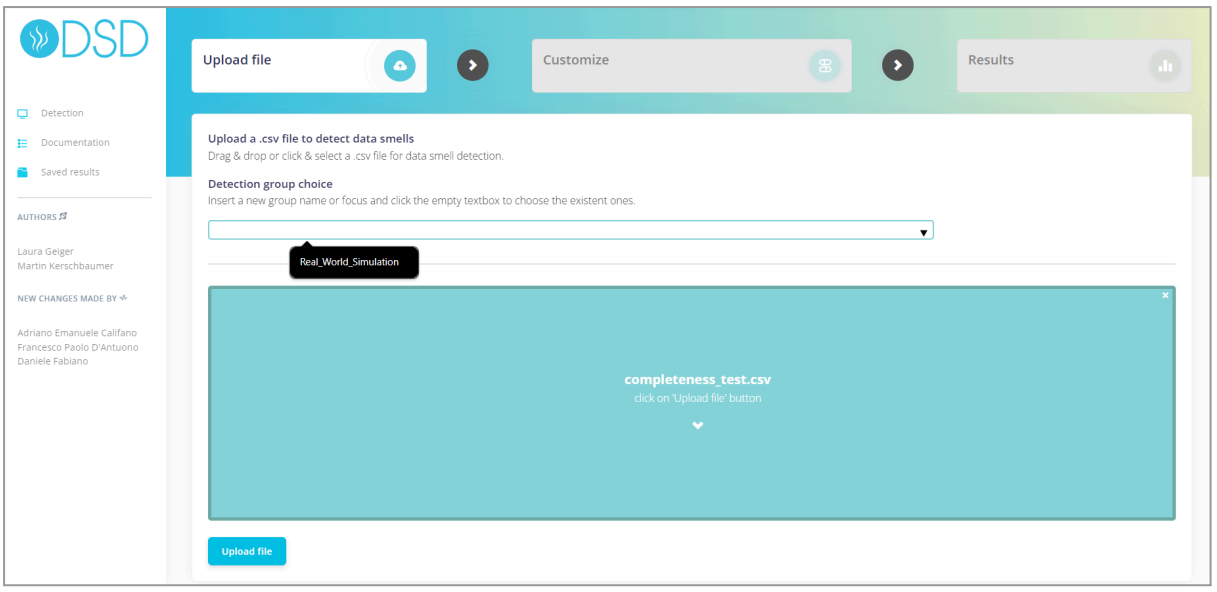
Il testing delle nuove funzionalità del sistema è descritto nei documenti che sono specificatamente dedicati al testing.

Descrizione delle nuove pagine

Home page

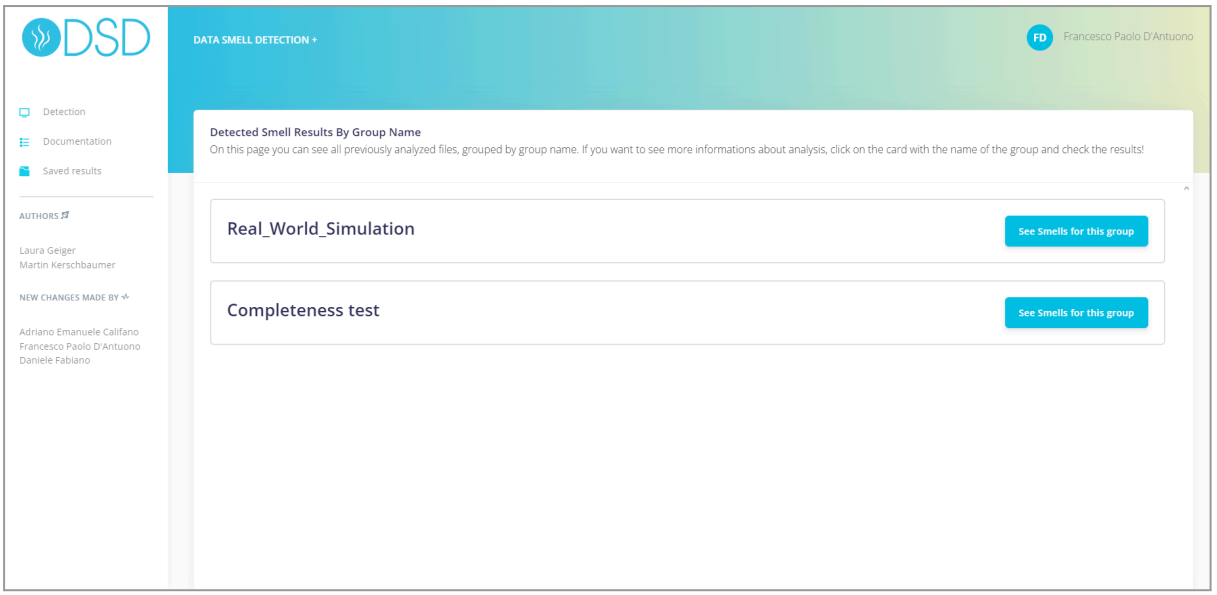
La nuova Home Page permette l’aggiunta del gruppo di risultati, dopo aver scelto il file da voler analizzare. Inoltre cliccando sulla casella di testo è possibile scegliere un gruppo già esistente dalla lista che compare.





Saved results page

La nuova Saved results page permette di avere una visualizzazione iniziale dei vari gruppi, che sono stati creati nella fase iniziale della detection, nei quali sono presenti tutti i risultati di uno stesso gruppo. Se si vogliono avere informazioni dettagliate di un gruppo, bisogna cliccare sul pulsante alla destra della lista. La pagina del gruppo permette di visualizzare le metriche di completeness, uniqueness e validity globali. Se si vogliono avere più dettagli per la detection del singolo dataset che fa parte di quel gruppo, bisogna cliccare sul pulsante con su scritto il nome del dataset del quale si vogliono visualizzare i risultati.



In questa specifica sezione, che si potrà visualizzare solo dopo aver cliccato sul pulsante, si può avere informazione sugli smell che sono stati detectati su ogni colonna e le metriche specifiche per quel dataset.