Skrypty składowe systemu AdePt

Skrypty obsługujące pobieranie JSONów z Sonara
Skrypty obsługujące analizę i zapis przestrzeni nazw w wynikach predykcji
Skrypty obsługujące źródła danych
Skrypty rozbijające JSONy na pojedyncze metryki

L.p.	Nazwa pliku	Realizowana funkcja	Używane tabele	Używane inne skrypty
1	**AnalyzeNamespacesDepth.py**	Skrypt bada jak głęboko sięgają nazwy – tj. ile jest maksymalnie przestrzeni nazw dla każdej kompletnej nazwy pliku źródłowego zapisanej w tabelach R_*, osobno dla każdej daty. Wynik działania umieszcza w tabeli FunctDictSummary Skrypt został napisany tak, że nie powinien w żaden sposób niszczyć/modyfikować już istniejących danych	Tabele rozpoczynające się od **R_*** (odczyt – nazwy plików i daty) **FunctDictSummary** (odczyt – sprawdzanie czy już są w nim wyniki z określonego dnia[1]) **FunctDictSummary** (zapis – para tabela R_, Data oraz maksymalna uzyskana liczba przestrzeni nazw w nazwie plików)	Brak

[1] Tutaj najprawdopodobniej jest bug do poprawki – wszystko wskazuje na to, że sprawdzanie jest realizowane tylko na datach: jeśli w tabeli FunctDictSummary znajduje się określona data, bez względu na to, z której tabeli R_ pochodzi, wyniki z daną datą ze wszystkich tabel R_ są ignorowane, bez względu czy znajdują się w FunctDictSummary czy też nie.

2	**ApiTools.py**	Skrypt definiujący funkcję get_json_from_api	Brak	loggerTools.py
3	**CountNames_in_Namespaces.py**	Skrypt leci po nazwach plików źródłowych i liczy unikalne nazwy w nich występujące na każdym z poziomów Skrypt został napisany tak, że nie powinien w żaden sposób niszczyć/modyfikować już istniejących danych	**FunctDictSummary** (odczyt maksymalnej głębokości przestrzeni nazw, dat wyników predykcji oraz identyfikator pliku dla dnia (ID)) **FunctNameComplexity** (odczyt – sprawdzenie, czy dane ID już istnieje w tabeli; zapis - liczba unikalnych nazw dla każdego poziomu głębokości przestrzeni nazw + identyfikator pliku dla dnia) **FunctionDictionary** (zapis – identyfikator pliku dla dnia (ID), identyfikator poziomu namespace (NamespaceDepth), nazwa namespace (Namespace))	AnalyzeNamespacesDepth.py musi być uruchomiony wcześniej, żeby skrypt wiedział, jak głęboko ma szukać ORAZ czy są jakieś nowe DATY (tzn – nowe wyniki predykcji)
4	**DatabaseTools.py**	Skrypt definiuje funkcje: update_database(metrics_tuple_list) create_tables(Base) project_exists(project_obejctt, ses)	Brak	loggerTools.py

		save_in_database_metric_list(metrics, ses)		
5	**MetricsKindCollection.py**	Skrypt przeszukuje JSONy przechowywane w tabeli **Metrics** pod kątem nowych typów metryk. „Nowy typ metryk" to taki, którego jeszcze nie ma w tabeli MetricTypes – jak nowy typ jest znaleziony, to tabelka MetricTypes jest uaktualniana	**Metrics** – odczyt JSONów **MetricTypes** – odczyt zapisanych tam typów metryk; zapis nowy typów metryk	Brak
6	**MetricsParser.py**	Skrypt parsuje każdy z pobranych z SonarQube JSONów zapisanych wcześniej w tabeli **Metrics**. Sprawdza, czy id z Metrics nie jest już obecny w MetricsParsed, jeśli nie, parsuje JSONY z nowymi id i wstawia wartosci metryk z odpowiednimi identyfikatorami liczbowymi MetricsType metryk (wziete z MetricTypes) do tabeli MetricsParsed.	**MetricTypes** – odczyt zapisanych typów metryk **Metrics** – odczyt JSONów **MetricsParsed** – odczyt wcześniej zparsowanych metryk (aby nie dublować); zapis – metryk dopiero co sparsowanych	Dobrze jest wcześniej uruchomić MetricsKindCollection.py aby sprawdzić, czy nie pojawiły się w SonarQube nowe metryki i mieć pewność, że każda z metryk dostarczanych przez SonarQube będzie miała swój identyfikator liczbowy w tabeli MetricTypes.
7	**SonarProjectsListUpdate**	Skrypt przeszukuje tabele **SonarDictionary** i dodaje do niej klucz sonarowy i sonarową nazwę projektu, jeśli nie znajdzie w niej wcześniej takiego klucza sonarowego, a pojawi się nowy w tabeli **Projects**	**SonarDictionary** – zapis MetricGeneratorKey (klucz sonarowy) oraz SonarName (nazwa projektu w Sonarze; - odczyt już istniejących kluczy sonarowych (zeby nie dublować)	Brak

			Projects – odczyt kluczy sonarowych i nazw projektu	
8	**SonarTools.py**	Skrypt definiuje metody: **get_all_metric_keys()** metoda zwraca wszystkie możliwe metryki w wersji sonara 4.5 **get_compleate_metrics_tuple_list()** metoda zwraca wszystkie dane z sonara w postaci listy krotek (Project, [Metrics]) **get_projects_APIlist()** metoda zwraca nam listę słowników zapytania API z danymi o projektach **get_project_objects_list(projects_APIlist)** metoda zwraca name listę obiektów Project TBD!!! Do dokończenia		
9	**SourceQuery.py**	Program przepytuje źródła – JIRA i SVN – i dla każdego klucza sonarowego obecnego w **SonarDictionary** dopisuje nazwy projektów z JIRA i SVN + SimilarRatio oraz status. Potem przepisuje dla każdego klucza sonarowego z **SonarDictionary** przepisuje dane zapisane w **ProjectsData** Jak klucza sonarowego obecnego w	Sources_SVN **ProjectsData** – manualnie wprowadzone dane o źródłach danych STATYCZNE, pisane „z ręki” (bądź przez interfejs www) NIENADPISYWALNE PRZEZ SKRYPTY	Warto wcześniej odpalić skrypt **SonarProjectsListUpdate**, żeby ogarnąć klucze sonarowe, które pojawiły się w sonarze a nie ma ich w tabeli SonarDictionary Jeśli tabela SonarDictionary jest pusta, to trzeba odpalić ww. skrypt.

		ProjectsData nie ma w **SonarDictionary** to skrypt tworzy nowy rekord w SonarDictionary z takim nowym kluczem (wziętym z ProjectsData) i danymi o źródłach dla niego		Warto wcześniej odpalić skrypt **SourceToDatabase.py** aby uaktualnić tabele Sources_SVN oraz Sources_JIRA
10	**SourceToDatabase.py**	Skrypt dopisuje do tabel Sources_JIRA oraz Sources_SVN źródła danych z tych dwóch systemów **TODO: Ma wyłączoną weryfikację SSL**	Sources_JIRA Sources_SVN – zapis do obu, jeśli nazwa tam jeszcze nie istnieje	Ponieważ tabele nie są uaktualniane[2], warto przed uruchomieniem skryptu JE WYCZYŚCIĆ - **ALTER TABLE Sources_JIRA;** **ALTER TABLE Sources_SVN;**
11	**SourcesRefresh**	Odświeża informacje o źródłach danych: 1. Czyści tablicę Sources_SVN 2. Czyści tablicę SonarDictionary 3. Realizuje polecenie ze skryptu 7 4. Uruchamia skrypt 10 5. Uruchamia skrypt 9		
12	**collect_sonar_data.py**	Zbiera JSONY z Sonara. Niezmordowanie.		SonarTools.py, DatabaseTools.py loggerTools.py
13	**loggerTools.py**	Definiuje: get_exception_logger() get_normal_logger() initialize_loggers() setup_logger(logger_name, log_file, level=loging.INFO)		
14	**models.py**	Definiuje klasy: Project(Base)		

[2] To trzeba poprawić na update, bo teraz program jak napotyka nazwę projektu w tabeli Sources_* to już potem z tym nic nie robi, a powinien robić update wiersza

		Metric(Base)		

Tabele

Sources_SVN – używane przez skrypty I frontend

Sources_JIRA – używane przez skrypty I frontend