Pytania typu „gap”

# Wyświetlanie

Dopuszczone są dwa typy wyświetlania:

1. Z danymi poszczególnymi literami lub bez

2. Z zaznaczoną ilością liter lub bez

W efekcie daje to 4 możliwości których wygląd wygląda następująco (przykładowa fraza „Ala ma kota”:

|  |  |  |
| --- | --- | --- |
|  | Z literami | Bez liter |
| Z ilością liter | \_ L \_ \_ \_ \_ \_ \_ A | \_ \_ \_ \_ \_ \_ \_ \_ \_ |
| Bez ilości liter\* | \_L\_ \_\_\_ \_\_\_\_\_A | \_\_\_ \_\_\_ \_\_\_\_\_\_\_ |

Tabela - tryby wyświetlania miejsc na odpowiedzi

\*- jest jedno podkreślenie dla całego słowa

Pojęcie „z ilością liter” oznacza że jednej literze odpowiada **dokładnie** jedno podkreślenie.

W przypadku podania liter, jeśli mamy podaną ilość liter, litera musi się znajdować idealnie w tym samym miejscu. Gdy ilość liter nie jest podana, rozróżniamy tylko 3 opcje – litera rozpoczyna wyraz, litera kończy wyraz, litera jest wewnątrz wyrazu, i prezentujemy to jako literę (odpowiednio) na początku, na końcu lub w środku kreski na wyraz (spacje gdy odpowiedzi są wielowyrazowe - tak czy inaczej są pokazywane)

# Struktura pytania w bazie:

W tabeli Question w polu „content” pisana jest treść pytania. W miejscu na lukę wstawiany jest Tag

**{numer}**, np.:

„W systemie linux aby przeniesc plik script.sh do folderu {1} wywoływana jest komenda {2} scripts.sh /root/scripts.sh”

Poprawne odpowiedzi:

luka 1 - /root/

luka 1 – root

luka 1 - /root

luka 2 – mv

odpowiedzi w tabeli answerGap w polu „questionId” mają id pytania na które są odpowiedzią, w polu „content” zawierają treść odpowiedzi, a w polu „gapNumber” zawierają numer luki na którą odpowiadają (integer, numer luki zawarty w Tagu). Ponieważ na jedną lukę może przypadać kilka poprawnych odpowiedzi, kluczem jest: id odpowiedzi + id pytania + numer luki do której pasuje.

# Realizacja w Javie

*Ten podpunkt głównie dla Patryka i Sokoła, ale w dokumentacji też się może przydać*

Obsługa pytań na poziomie Javy działa analogicznie do obsługi typów pytań. Metody obsługujące umieszczone są w **beans.imports.TypeCalculator**

Sprawdzanie pytań:

* *TypeCalculator.hasLetters(int type)* – Sprawdza czy przy wyświetlaniu odpowiedzi mają być wyświetlane litery czy nie. Parametrem jest typ odpowiedzi (pobrany z bazy danych). Jeśli zwróconą wartością jest **true** to odpowiedź ma mieć litery (na razie niech posiada 25% liter, ilość liter będzie jako bonus na później. Wyglądać to ma jak w pierwszym wierszu tabeli 1), jeśli zwrócona wartość to **false** to nie wyświetlamy liter (odpowiedź wygląda jak 2 wiersz z tabeli 1)
* *TypeCalculator.hasUnderlines(int type)* – Sprawdza czy przy wyświetlaniu odpowiedzi każdej literze ma odpowiadać oddzielne podkreślenie, czy jednemu słowu jedna litera. Parametrem jest typ odpowiedzi (pobrany z bazy danych). Jeśli zwróconą wartością jest **true** to odpowiedź ma mieć dla każdej litery odpowiedzi oddzielną linię, jeśli zwrócone jest **false** to jedna linia dla każdego wyrazu odpowiedzi.

Ustawianie typu odpowiedzi:

* *TypeCalculator.setLetters()* – ustawia odpowiedź z wyświetlanymi literami pomocniczymi
* *TypeCalculator.setUnderlines()* – ustawia odpowiedź w której jedna litera to jedno podkreślenie

Trzeba pamiętać że te typy odpowiedzi zapisywane są bitowo, więc odpowiedź z literami **i** podkreśleniami ustawia się tak:

*int type = (TypeCalculator.setLetters() | TypeCalculator.setUnderlines());*