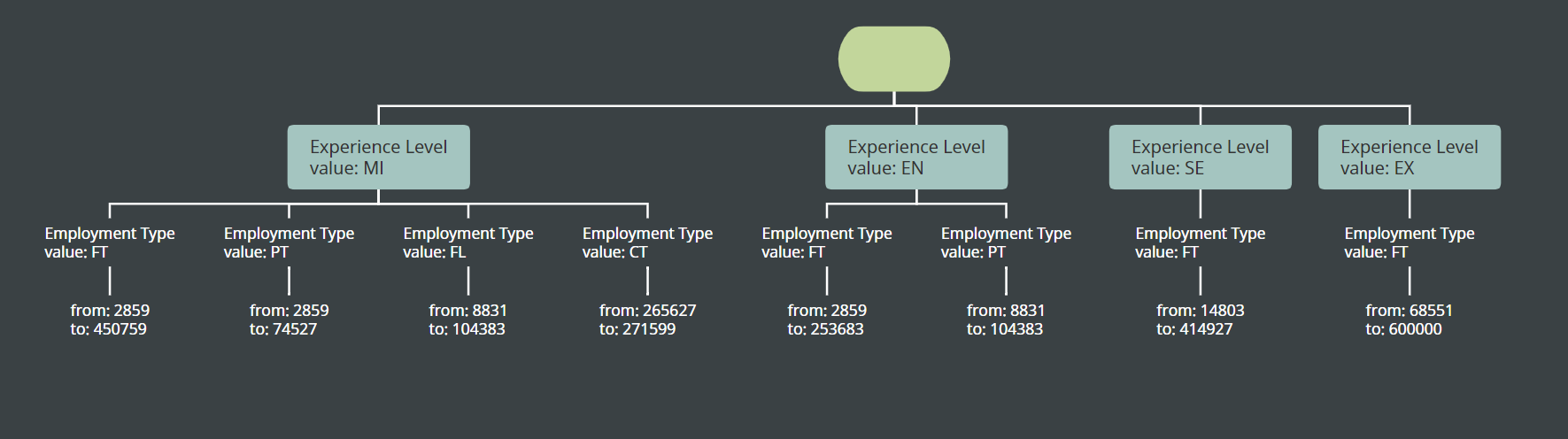
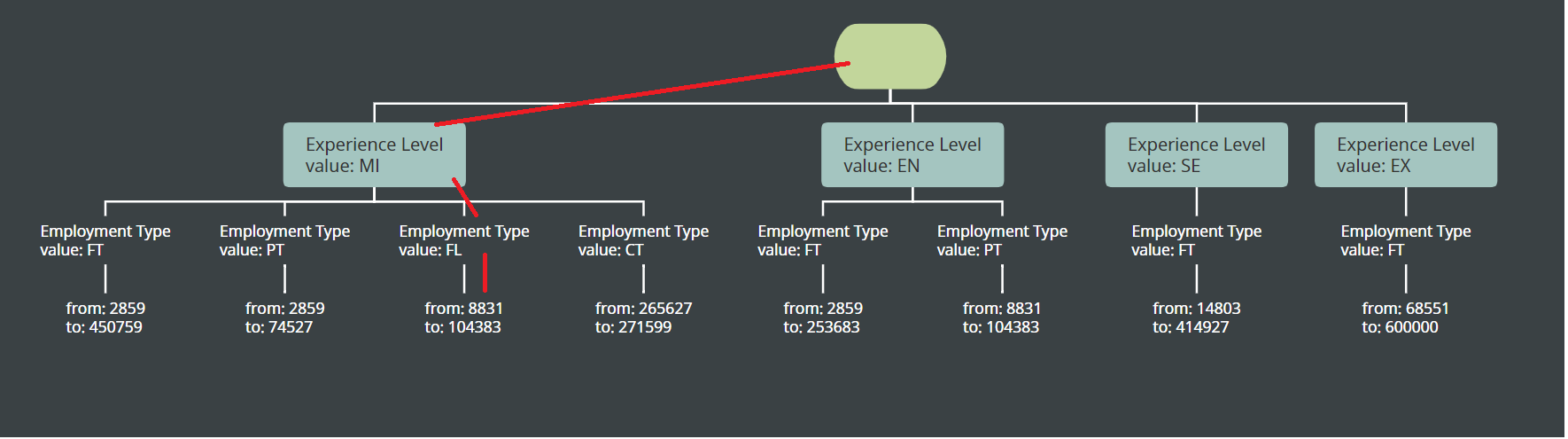
1. Objaśnienie struktury drzewa decyzyjnego  
   Drzewo decyzyjne jest obiektem javascriptu. Jego orginalna struktura została przedstawiona w pliku doc/original-tree.json (zostało przekonwertowane na format JSON-a).   
     
   Każda gałąź bazy danych jest obiektem o strukturze:  
   - propName – nazwa przechowywanej właściwości  
   - value - wartość  
   - children – tablica przechowująca następne parametry o przedstawionej tutaj strukturze  
     
   Rozwiązywanie danych przekazanych do programu odbywa się na zasadzie przechodzenia przez kolejne gałęzie drzewa aż do dojścia do gałęzi, której propName odpowiada poszukiwanej wartości. Rozwiązaniem jest właściwość ‘value’ tej gałęzi.  
     
   Weźmy sobie przykładowe drzewo stworzonego w tym powyższym standardzie.  
     
     
     
   Dla przykładowych danych:  
   {  
    experience\_level: MI,  
    employement\_type: FL  
   }  
     
     
     
   Musimy przejść drogą, która została przedstawiona powyżej na czerwono. W wyniku przeszukiwań dojdziemy do rozwiązania, gdzie wynagrodzenie jest od 8831$ do 104383$ dla podanych danych.
2. Opisanie działania programu
   1. właściwości uwzględniane w obecnym drzewie decyzyjnym (opis wszystkich dostępnych wartości można znaleźć w pliku README w projekcie):
      1. Work\_year – definiuje rok, w którym była wykonywana praca
      2. Experience\_level – definiuje doświadczenie pracownika
      3. Job\_title – definiuje kategorię wykonywanej pracy
      4. Company\_size – definiuje rozmiar firmy
      5. Employent\_type – definiuje etat wykonywanej pracy
      6. Salary\_in\_usd – definiuje zarobki pracownika, jest to poszukiwana wartość
   2. program wczytuje dane z przykładowego pliku, który znajduje się w data/employees/ds\_salaries.csv. Na ich podstawie tworzone jest drzewo decyzyjne.
      1. Program pobiera wszystkie rekordy zbierając wszystkie możliwe wartości poszczególnych właściwości (również tej poszukiwanej czyli salary\_in\_usd)
      2. Następnie salary\_in\_usd jest dzielone na przedziały (ich ilość jest zdefiniowana w src/models/DecisionTree jako COUNT\_RANGE. Definiuje to dokładność, z jaką będą wyznaczane będą wyznaczane pensje.
      3. Następnie wszystkie rekordy w bazie danych są przypisywane do określonych przedziałów wypłat na zasadzie – dla określonych właściwości pasuje dany przedział. Jeśli po przejściu całego drzewa okaże się, że dla danego zestawu właściwości jest kilka przedziałów wypłat zostaną one zsumowane.
      4. Następnie struktura listy jest przerabiana na strukturę drzewa. Polega to kolejno na:
         1. Przejdź do pierwszego elementu listy. Dla każdego kolejnego parametru odbij od nową gałąź z wartością się tam znajdującą, aż dojdziesz do wartości ‘salary\_in\_usd’, czyli poszukiwanej
         2. Następnie dla każdego następnego elementu listy powtarzaj określone czynności: zacznij od pierwszego parametru, sprawdź czy jest on zdefiniowany w dotychczasowym drzewie, jeśli tak, przekieruj drzewo na tę gałąź, jeśli nie, odbij nową gałąź i przekieruj się w tamtą stronę
         3. Powtarzaj to dla każdego elementu listy, aż dostaniesz pełne drzewo
      5. W następnym kroku skorzystaj z dotychczas zdefiniowanych test-data, lub utwórz nowe poprzez utworzenie JSON-a z parametrami wejściowymi i podłączenie go w pliku data/employees/DataProvider. Program w ten sposób pozwoli ci wyznaczyć wynagrodzenie dla określonych właściwości.
      6. Jeśli okaże się, że dla określonych właściwości nasze drzewo nie otrzymało żadnych danych, rzuci ono błąd w tym momencie i poinformuje użytkownika o właściwości powodującej problem, oraz pokaże wszystkie możliwe