Kamiński Mateusz
Mościcki Adam

Dokumentacja aplikacji *SPOP-calc*

1. Realizacja projektu, wybór technologii.

Projekt arkusza kalkulacyjnego na przedmiot SPOP nie został całkowicie zrealizowany w języku Haskell. Podjęliśmy się próby stworzenia wygodnego i nowoczesnego interfejsu użytkownika. Zdecydowaliśmy się więc na użycie platformy programistycznej Yesod do tworzenia aplikacji internetowych. Wielkim plusem szkieletu Yesod jest bycie zgodnym ze współczesnymi wymaganiami dla tego typu aplikacji jak i wspomaganie programisty w jego trudzie programowania. Ponadto, aby uprościć proces wytwarzania oprogramowania, użyliśmy systemu budowania stworzonego dla języka Haskell jak i w języku Haskell - stack. Pozwala on na zarządzanie w prosty sposób na zależnościami, testowanie i benchmarkowanie projektu - będąc pełnoprawnym odpowiednikiem narzędzia maven. Dodatkowo wykorzystane zostały takie biblioteki jak: Parsec - parsowanie komórek, Array - modelowanie arkusza, Aeson - konwersja z/do JSON, UUID - generowanie unikalnych identyfikatorów. Aplikacja tworzona była przy użyciu narzędzia Visual Studio Code z odpowiednimi rozszerzeniami dla Haskella.

Projekt został podzielony na dwa główne podprojekty:

- spop-calc - zawiera wszystkie funkcje niezbędne do reprezentowania arkusza w pamięci i operowania nim (rezultat - biblioteka dołączona do drugiego podprojektu)
- spop-calc-web - pełnoprawna aplikacja internetowa, główny interfejs użytkownika pozwalający na wczytanie / zapis / operowanie arkuszem, dodatkowo udostępnia webserwis typu REST

Aplikacja została uruchomiona na darmowym serwerze w usłudze “platform as a service” Openshift. Dostępna jest pod adresem: http://www.spop-calc.tk/.

2. Interfejs użytkownika.

Zgodnie z oceną języka Haskell: https://github.com/Gabriel439/post-rfc/blob/master/sotu.md, tworzenie aplikacji z interfejsem użytkownika jest na niskim poziomie rozwoju (biblioteki, tutoriale, etc.). Natomiast rozwiązania do tworzenia backendów dla serwisów WWW są oceniane wysoko. Stąd uznaliśmy, że interfejs naszej aplikacji zrealizujemy jako stronę WWW, gdzie za logikę będzie odpowiadać serwer napisany w Haskellu, a za wygląd i logikę interakcji HTML (Hamlet) / CSS-LESS (Lucius) / JS (Julius).

Za pomocą przeglądarki użytkownik aplikacji może utworzyć nowy, pusty arkusz, wgrać istniejący lub pobrać aktualny stan prac nad arkuszem. Dla edycji arkusza stworzony został wygodny interfejs w formie tabelki, którą można obsługiwać za pomocą klawiatury jak i myszki. Pod prawym przyciskiem myszki znajduje się menu kontekstowe dla dowolnej komórki arkusza, co pozwala na korzystanie z szablonów dostępnych funkcji (SUM, AVG i MUL) oraz usuwanie wierszy lub kolumn. Dwukrotne kliknięcie w nagłówek kolumny lub wiersza powoduje dodanie nowej kolumn lub wiersza w dowolnym miejscu.

3. Szczegóły implementacyjne.

Wewnętrzna biblioteka **spop-calc** pozwala na reprezentowanie arkusza jako typu *Sheet*. Dla wydajnego przetwarzania dużych arkuszy zastosowana została biblioteka Array, której instancja jest nośnikiem dla zinterpretowanej zawartości pliku CSV (*CellContent*, *CellCord*). Typ Sheet jest immutable w sensie rozmiaru arkusza kalkulacyjnego, więc każda zmiana w wymiarach wymaga ponownego utworzenia instancji tego obiektu. Funkcja readSheet umożliwia utworzenie i zinterpretowanie arkusza reprezentowanego jako typ *[[String]]*, a *writeSheet* na zapis do tego typu. Do zmiany aktualnej zawartości komórki należy użyć funkcji *alterCell*, do usunięcia wybranego wiersza *removeRow* i odpowiednio *removeCol* dla usunięcia kolumny. Wszystkie powyższe funkcje zostały przetestowane jednostkowo.

Aplikacja internetowa **spop-calc-web** udostępnia możliwość wgrania dowolnego pliku CSV, który powinien posiadać odstępy tabulacyjne pomiędzy kolejnymi komórkami. Platforma yesod reprezentuje wgrany plik jako instancję pola formularza (*Input Form*), która jest reprezentowana jako typ źródła (Source). Proces interpretacji wgranego pliku wygląda następująco: *Source -> Lazy.ByteString -> [[String]] -> Sheet*.

Po wgraniu pliku na serwer i poprawnej walidacji, plik umieszczany jest w dostępnej pamięci aplikacji przy pomocy typu TVarIO, gdzie dla każdej sesji generowany jest ciąg znaków przy pomocy biblioteki UUID. Zapamiętanie cookie w przeglądarce umożliwia w dowolnym momencie na dalszą edycję wgranego arkusza.

Aplikacja udostępnia API typu REST, które umożliwia na dodawanie / usuwanie kolumn i wierszy a także edycję aktualnej zawartości komórek. Użyte zostało rozszerzenie Haskella *deriveJSON* do prostej zamiany reprezentacji typów.