MCTimeMeter

Mateusz Chodkowski nr indeksu 161127

Założenia projektu

Program ma umożliwiać pracownikowi rejestrację czasu pracy nad konkretnymi zadaniami wprowadzonymi z automatycznie aktualizowanego pliku excela, perzystować wyniki w oddzielnym pliku a także zapamiętywać obecny stan programu w przypadku jego zamknięcia. Interfejs ma być prosty i czytelny a także zawierać możliwość modyfikacji zarejestrowanych wyników czasowych oraz dopisania dodatkowych zadań. Aplikacja ma też udostępniać widget który jest zawsze obecny na ekranie ponieważ pracownicy mają mieć poczucie czasu jaki spędzają nad zadaniami i czuć presję aby wyniki były dokładne.

Dane uruchomieniowe

Wszystkie dane uruchomieniowe zostały wpisane w pliku ProductProperties.cs w projekcie Data. Omówienie znajduje się poniżej:

InputFileName – plik wejściowy zawierający kolumny :

- CRK_NumerPelny - pełen numer zadania
- Ope_Kod - kod pracownika nadzorującego
- Knt_Kod - kod klienta
- Kat_KodOgolny - kod handlowca
- CRK_DataDok - data utworzenia zadania

InputFileSheetName – nazwa arkusza w pliku wejściowym

OutputFileName – nazwa pliku wyjściowego (z raportem)

OutputFileSheetName – nazwa arkusza w pliku wyjściowym

LastSessionFileName – nazwa pliku xml z danymi z ostatniej sesji programu

LastLoggedUserFile – nazwa pliku xml z danymi ostatnio zalogowanego użytkownika

Podział solucji na projekty:

- **Data –** Zawiera klasę TimeRow oraz Klasę opisującą podstawowe dane uruchomieniowe
- **Helpers –** Zawiera klasy odpowiedzialne za komunikację z plikami xml oraz xls
- **MCTimeMeter –** Zawiera implementację kontrolera oraz widoku głównego okna aplikacji
- **MCTMWidget –** Zawiera Kontrolkę i widok widgetu który jest uproszczonym podglądem czasu pracy nad aktywnym zadaniem
- **TimePanelControl –** Zawiera klasę DayRegister odpowiedzialną za kontrolę nad panelami oraz kontrolkę i o nazwie takiej samej jak nazwa projektu, która jest pełnym modelem pojedynczego zadania.

Komentarz: Obecna struktura nie jest satysfakcjonująca i jest w fazie wstępnej refaktoryzacji. Klasy powinny zostać odpowiednio rozbite w taki sposób aby nie napotkać na problem wzajemnych referencji.

Model

TimeRow – model pojedynczego zarejestrowanego wpisu pracy nad zadaniem:

- Start – Data rozpoczęcia pracy
- End – Data zakończenia pracy
- TaskName – Nazwa zadania
- IsAddedManually – Czy wpis został wygenerowany na podstawie manualnego dodania czasu
- Duration – Łączny zas pracy w formie TimeSpan
- DurationAddedManually – Czas pracy dodany manualnie w formie TimeSpan
- DesignerCode – Kod pracownika pracującego nad zadaniem
- MainDesignerCode – Kod pracownika nadzorującego zadanie
- ClientCode – Kod klienta dla którego wykonywane jest zadanie
- TraderCode – Kod handlowca który odpowiada za to zadanie

Powyższa encja jest perzystowana do pliku excela w postaci wiersza o kolumnach, który posłuży do raportu generowanego przez odpowiednie makro:

- Projekt
- Projektantka
- Kontrahent
- Handlowiec
- ProjektantOpiekun
- CzasWMinutach
- Start
- Stop
- KorektaWMinutach

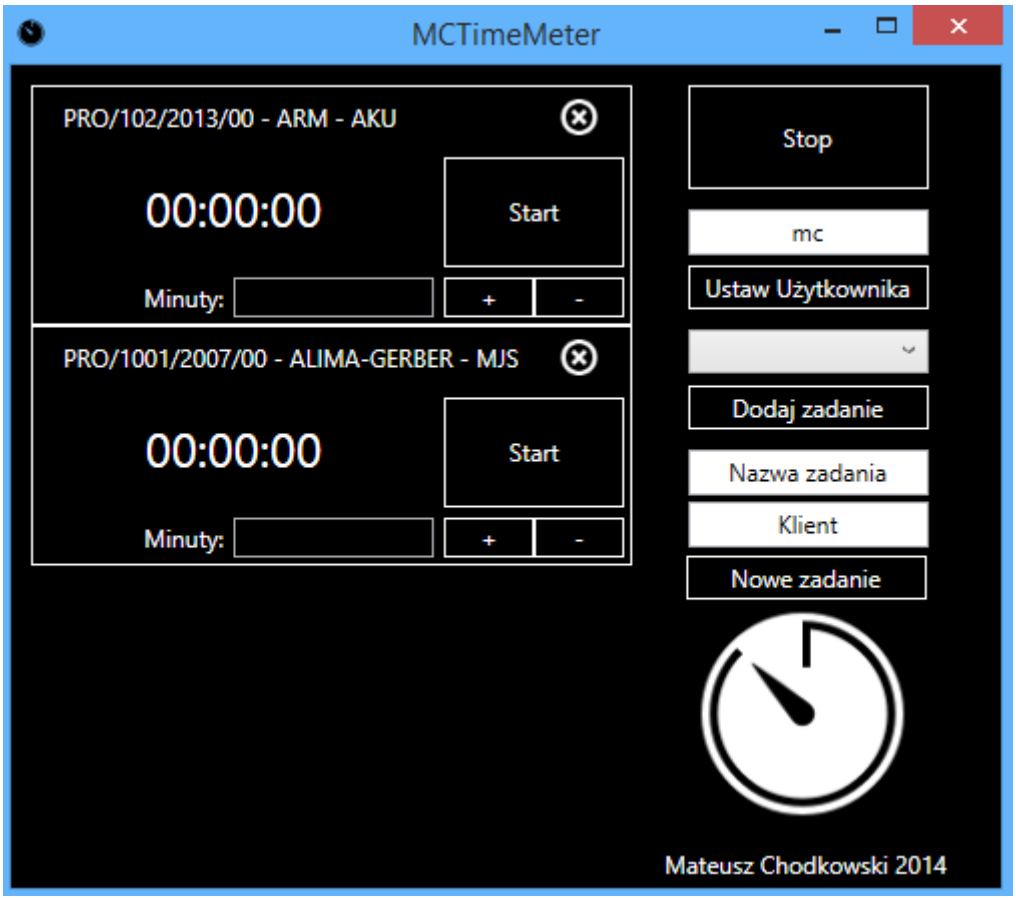
DayRegister – Ta encja perzystowana jest na potrzeby ponownego uruchomienia aplikacji do odpowiedniego pliku xml. Odpowiada ona za kontrolę nad zadaniami, zapamiętuje użytkownika i jego wybrane w poprzedniej sesji zadania. Dane pobrane z pliku wejściowego służą bezpośrednio do utworzenia kontrolek TimeControl (o których za chwilę) gdzie są przekazywane do DayRegister i w ten sposób perzystowane. Poza metodami których nazwy są autodokumentujące warto zapamiętać trzy podstawowe listy:

- List<TimeRow> Rows - Dotychczas zachowane wpisy
- ObservableCollection<TimeControl> Controls - Dodane na główny widok kontrolki zadań
- ObservableCollection<TimeControl> InactiveControls - Nie dodane jeszcze na główny widok kontrolki zadań

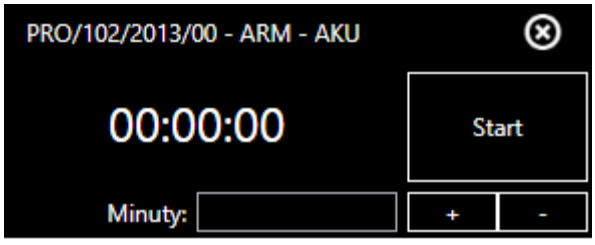
Widoki

Widoki w aplikacji zostały zaprogramowane w xaml'u. Każdy z nich korzysta z oddzielnego kontrolera i jest w pełni oprogramowany. Główne trzy widoki odnoszą się do głównego okna aplikacji (MCTimeMeter.MainWindow), kontrolki pojedynczego zadania (TimePanelControl .TimePanelControl) oraz do widgetu (MCTMWidget.MainWindow). W kolejności:

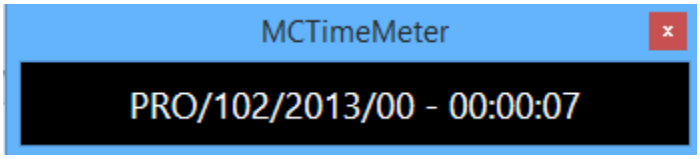
MCTimeMeter.MainWindow – główne okno aplikacji, pozwala na zmianę użytkownika i dodawanie zadań z listy. Minimalizacja tego okna uruchamia widget. W przypadku braku zadań na liście w poprzedniej sesji główne okno zostaje zminimalizowane tak aby pokazywać tylko prawy panel.



TimePanelControl .TimePanelControl – Kontrolka składowa powyższego widoku. Pozwala na wystartowanie zegara oraz ręczne dodawanie i odejmowanie minut. W widoku kontrolki znajduje się również przycisk służący do usunięcia jej z listy obserwowanych zadań



MCTMWidget.MainWindow – Widget wyświetlający aktywne zadanie oraz czas pracy nad nim. Zamknięcie tego okna wywołuje powrót do głównego okna.



Kontrolery:

Z racji moich podstawowych umiejętności dot. xaml'a postanowiłem odpuścić używanie bindingu oraz praktyk przybliżonych do MVVM w związku z czym dużo ciężaru i funkcjonalności zostało przełożone na kontrolery widoków.

MCTimeMeter.MainWindow - Kontroler uruchamiany jest na starcie aplikacji w związku z czym jego konstruktor wykorzystuje projekt Helpers do pobrania danych pliku wejściowego oraz poprzedniej sesji. Poza tym znajdziemy tu metody odpowiedzialne za stopowanie, dodawanie i zdejmowanie zadań. Dodatkowo użyto tu ciekawego rozwiązania z selektywnym ignorowaniem kliknięć myszki dzięki któremu dochodzi do zaznaczania całego tekstu w opdpowiednich polach tak by można go było łatwo edytować. Znajdziemy tu również mechanizm powodujący pojawienie się widgetu gdy zminimalizujemy okno oraz zapis danych w przypadku gdy je zamykamy.

TimePanelControl .TimePanelControl – dosyć skomplikowany kontroler, poza przekazywanymi do niego parametrami posiada również zegar DispatcherTimer do którego podpięte zostało kilka metod oraz podpięcie do instancji widgetu dzięki czemu łatwo odbywa się komunikacja między kontrolką a uruchomionym widgetem. Znajdziemy tu również metodę zatrzymującą i startującą zegar. Konstruktory, metodę elapsacji czasu zegara, metody obsługujące dodawanie i odejmowanie czasu pracy, obsługę przycisków i inne. Z najbardziej interesujących części warto zwrócić tu uwagę na:

- implementację IComparable dzięki której możemy sprawdzać czy zadania pobrane z pliku nie duplikują się z zadaniami pobranymi z ostatniej sesji programu.
- metodę *StopTimer()* która zpisuje dane do pliku wynikowego
- metodę *StartButton_Click* która uruchamia zegar oraz zatrzymuje wszystkie pozostałe przy pomocy DayRegister

MCTMWidget.MainWindow – Kontroler ma dosyć prostą implementację, tak naprawdę jest to tylko prosta obsługa zegara oraz metoda służąca do powrotu na główny ekran aplikacji gdy okno zostaje zminimalizowane.

Podsumowanie

Projekt w nietypowy sposób wykorzystuje MVC, pomimo wszystko zachowuje standardowy podział zadań i realizuje zaplanowaną funkcjonalność. Oderwanie widoku od kontrolera pozwala w każdej chwili na szeroką modyfikację. Użycie tutaj tej technologii jest celne choć zwykle w tego typu aplikacjach stosuje się MVVM który również sprawdziłby się, ale napewno nie powstał w tak szybkim tempie. W projekcie udało się również uniknąć wprowadzania dodatkowych zdarzeń, dzięki czemu mamy centralizację danych w odpowiednich statycznych klasach jak np. DayRegister. Projekt jest kontynuowany na zlecenie pewnej firmy i obecnie jest już używany od około miesiąca, wciąż jednak uważam że jego architektura wewnętrzna mogłaby być znacznie lepsza a wiele funkcjonalności rozbitych na mniejsze funkcje i eksportowane do dodatkowych klas pomocniczych. Jeśli chodzi o perzystencję danych to klient nalegał aby „zewnętrzna" aplikacja nie miała bezpośredniego dostępu do bazy danych więc rozwiązanie pośrednie (plik xls na dysku sieciowym aktualizowany przez odpowiednie makro przy każdym otwarciu) jest niezbyt eleganckie ale powstało w oparciu o kompromis między logiką biznesową a wymagania klienta.