Programowanie współbieżne i rozproszone

Symulacja dziekanatu

Prowadzący: dr inż. Jacek Piwowarczyk
Autorzy: Marcin Skowron i Michał Kałduś
Wydział: Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej
Kierunek: Informatyka
Grupa: 3
Język wykonania: Erlang

Spis treści

1 Cel programu

Celem wykonanego przez nas programu była symulacja instytucji publicznej jaką jest dziekanat. Wykonany przez nas system odzwierciedla zachowania dziekanatu na Akademiku Górniczo Hutniczej w Krakowie na wydziale Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej. Najważniejszym punktem wykonania projektu było zastosowanie mechanizmów współbieżności w celu zrealizowania poszczególnych zachowań symulowanego obiektu.

2 Opis systemu

Pogram obejmuje symulację pracy 5 osób odpowiedzialnych za przyjmowanie studentów (panie z dziekanatu), pana dziekana, wyświetlacz przydzielający kolejne numery w kolejce dla petentów oraz samych studentów. Każdy z wyżej wymienionych elementów jest osobnym procesem w celu urealistycznienia zachowania. Dodatkowo napisany przez nas program jest bardzo mocno parametryzowalny. Możemy ustawić jak szybko wykonuje się dana symulacja (przeskalowanie odpowiednio jednostki czasu) oraz z jaką częstotliwością przychodzą studenci. Ilość petentów jest też uzależniona od aktualnej godziny w systemie. Przebywająca osoba w dziekanacie może zostać na różne sposoby obsłużona między innymi:

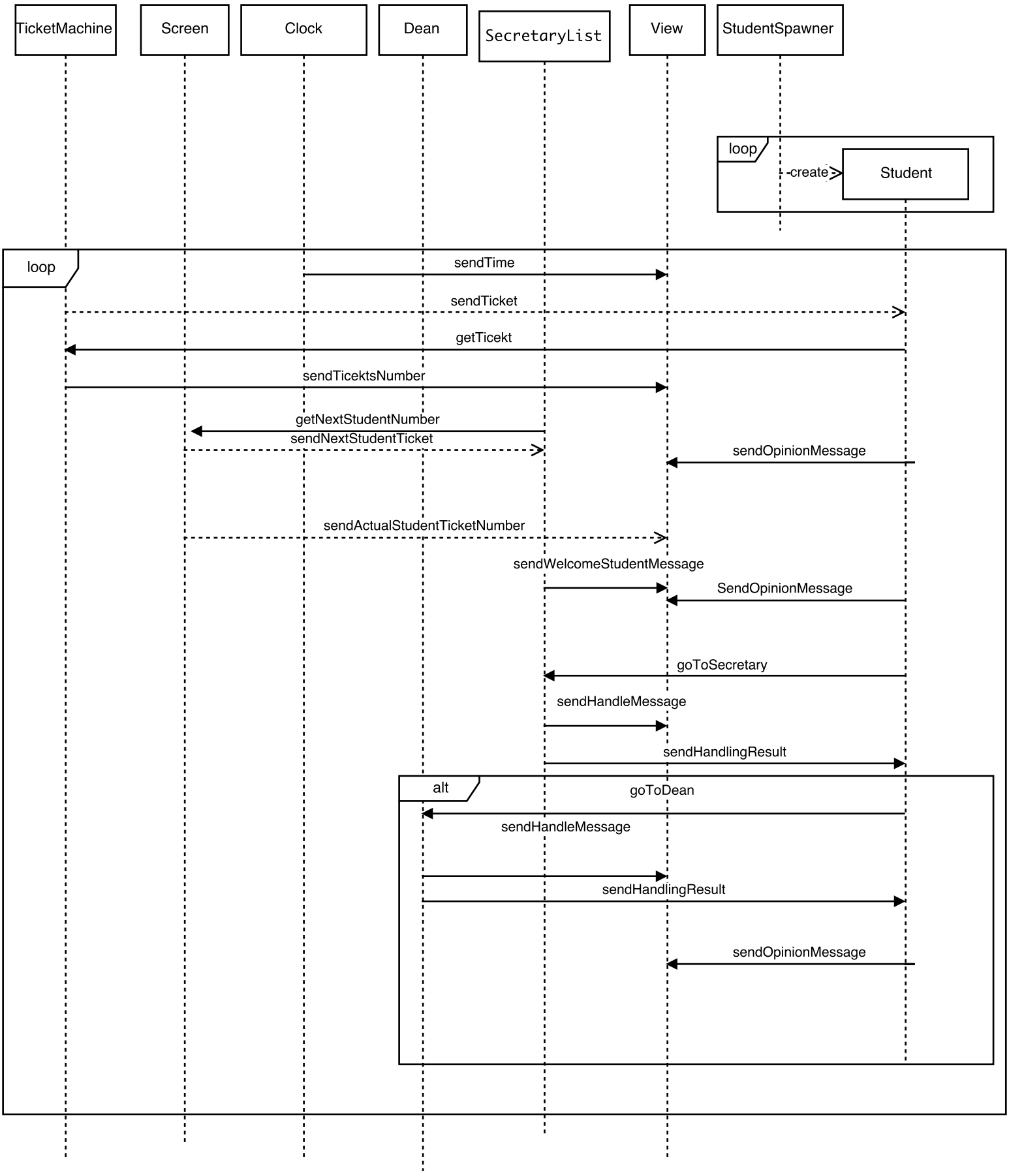
- poprawnie obsłużona
- odesłana do dziekana
- poproszona o przyjście innego dnia

Kolejnym elementem programu jest to że każdy student po obsłużeni oraz w trakcie czekania w kolejce wyraża swoją opinie, którą może być:

- radość z poprawnego obsłużenia w dziekanacie
- narzekanie na przerwę w pracy obsługi dziekanatu
- poinformowanie o przymusie udania się do dziekana
- smutek z powodu nie załatwienia swojej sprawy

3 Struktura logiczna

Poniżej przedstawiamy uproszczony schemat działania aplikacji. Każda z linii życia symbolizuje proces lub jak w przypadku *SecretaryList* czy *Student* grupę procesów, które funkcjonują na takich samych bądź podobnych zasadach.


TicketMachine
Screen
Clock
Dean
SecretaryList
View
StudentSpawner
loop
create
Student
loop
sendTime
sendTicket
getTicekt
sendTicektsNumber
getNextStudentNumber
sendNextStudentTicket
sendOpinionMessage
sendActualStudentTicketNumber
sendWelcomeStudentMessage
SendOpinionMessage
goToSecretary
sendHandleMessage
sendHandlingResult
alt
goToDean
sendHandleMessage
sendHandlingResult
sendOpinionMessage

4 Specyfikacja projektu

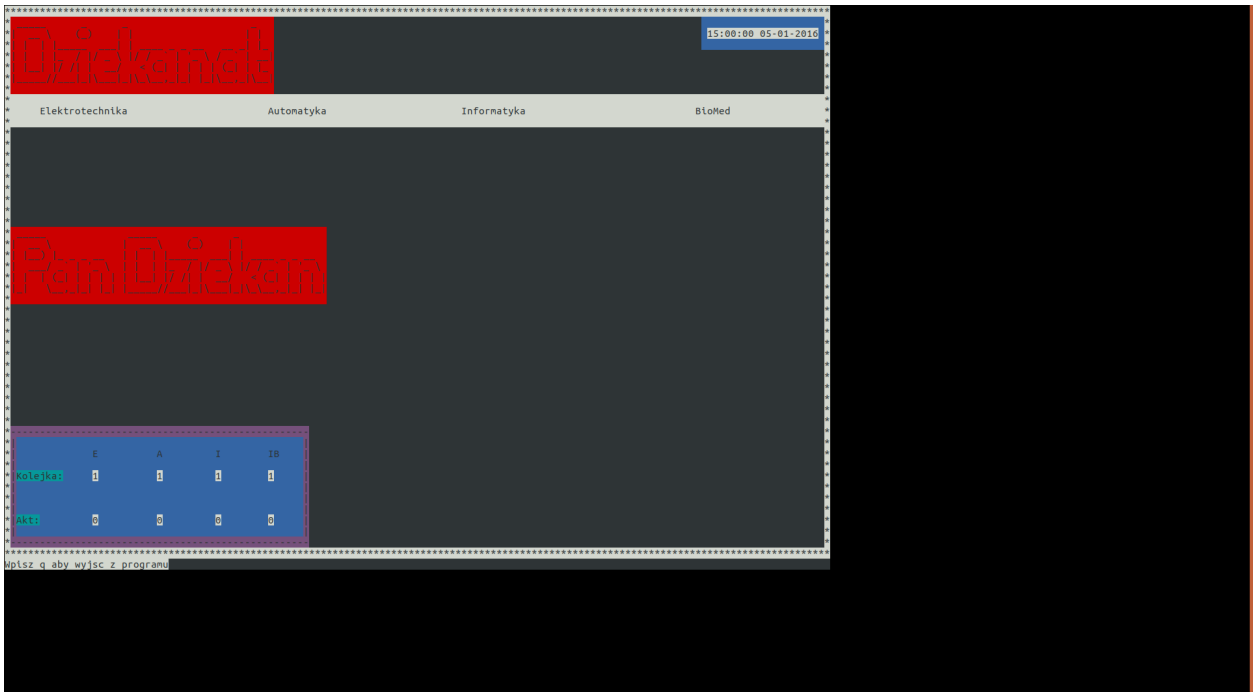
W celu zrealizowania warstwy widoku naszego projektu posłużyliśmy się biblioteką **cecho** ze strony `https://github.com/mazenharake/cecho`, która adaptuje bibliotekę ncurses z języka C. Aby ułatwić kompilacje i budowanie projektu użyliśmy narzędzia o nazwie: *rebar*. W celu łatwego i szybkiego uruchamiania projektu stworzyliśmy skrypt uruchomieniowy *run.escript*.

W celu poprawnego uruchomienia aplikacji należy w terminalu wpisać dwie poniższe komendy.

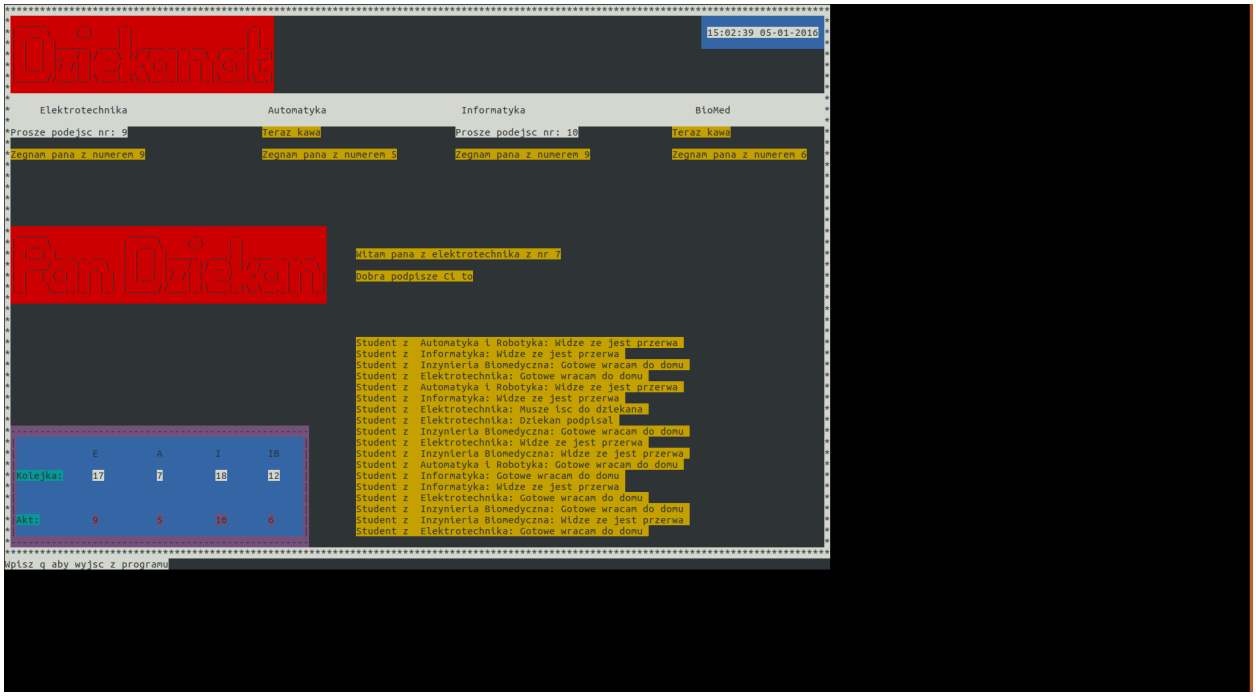
UWAGA! Należy pamiętać o nadaniu odpowiednich praw tym dwóm plikom.

```
./rebar compile
./run.escript
```

5 Przykłady



Rysunek 1: Zrzut ekranu



Rysunek 2: Zrzut ekranu

6 Rozszerzenia

Do programu można dodać możliwość wprowadzania nowych studentów ze standardowego wejścia w czasie trwania programu.

Kolejną możliwością rozbudowania jest dodanie mechanizmu odesłania studenta, który ma przyjść do dziekanatu za określoną ilość czasu.

Ciekawym rozszerzeniem programu mogłoby być dodanie mechanizmu odciążania bardziej obciążonych sekcji, przez sekcji, które nie mają aktualnie studentów w kolejce lub maja ich znacznie mniejszą ilość.

Innym nieco bardziej skompilowanym sposobem ulepszenia projektu mogłoby być zmiana sposób prezentacji danych np. zmienić aktualny terminal na okno przeglądarki internetowej