Warszawa,  
16.04.2016

**System zarządzania zadaniami i rejestracji czasu pracy “Karbowy”**

*projekt wstępny*

***Autorzy*:**  
 **Wojciech Wiśniewski**

**Mikhail Lukashevich**

**Tatsiana Lukashevich**

**Yauheni Barodzich**

# Treść zadania

Proszę założyć istnienie 1 stacji szefa i N stacji pracowników. Szef definiuje zbiory zadań i przydziela te zbiory pracownikom. Zadania mają status: nieprzydzielone, realizowane, zrealizowane częściowo (X na Y pracowników zakończyło jego realizację), zrealizowane całkowicie (podana liczba pracowników i łączny czas ich pracy). W każdej chwili dowolna stacja może być włączona lub wyłączona. Pracownik widzi przydzielone zadania i oznacza: rozpoczęcie pracy nad wybranym zadaniem, przerwanie pracy, zakończenie realizacji danego zadania. Stacja szefa wykonuje okresowe backupy danych ze stacji pracowników. Stacja pracownika ma możliwość odtworzenia danych z backupu. Należy zaprojektować i wykonać system komunikacji pomiędzy stacjami. System komunikacji powinien móc pracować w przestrzeni adresów IPv4 i IPv6. Ponadto należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów.

# Wymagania funkcjonalne

## Założenia podstawowe

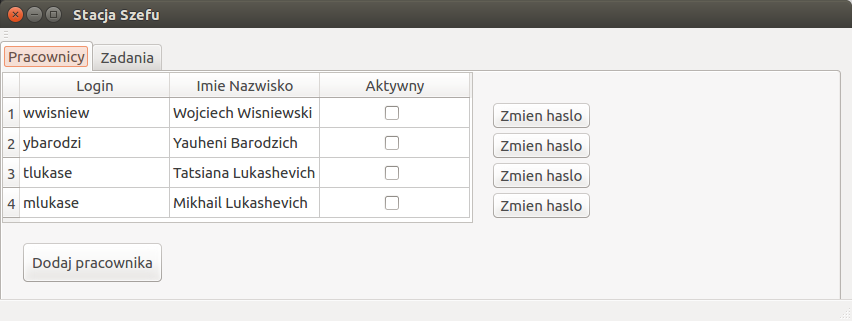
Użytkownicy systemu przyjmują jedną z dwóch ról: *szefa* lub *pracownika*. Szef jest jeden, pracowników może być dowolna liczba. Szef pracuje na stacji szefa, pracownicy - na stacjach pracowników. Szef jest przypisany do swojej stacji, natomiast pracownik może skorzystać z dowolnej stacji pracownika. Pracownik identyfikuje się przez identyfikator (login) i hasło. Stacja szefa pełni rolę *serwera*, a stacje pracowników *klientów*: wszelka komunikacja jest inicjowana przez stacje pracowników, a stacja szefa tylko odpowiada na ich żądania. W dalszej części dokumentu pojęcia serwer i stacja szefa oraz klient, stacja kliencka i stacja pracownika będą używane zamiennie. Zarówno stacja szefa, jak i stacje pracowników mają przydzielone globalnie unikalne identyfikatory (GUID). Stacja pracownika zna identyfikator swojego serwera, natomiast serwer nie przechowuje listy uprawnionych do komunikacji z nim klientów; każda ze stron uznaje drugą za wiarygodną, gdy wykaże się ona znajomością identyfikatora stacji szefa. Stacja kliencka wymaga łączności z serwerem tylko w momencie logowania i pobierania listy zadań, w pozostałym czasie pracuje autonomicznie.

## Przypadki użyciaTIN-przypadki-uzycia.png

### Stacja szefa (aktor: szef)

#### Edycja listy pracowników

Na swojej stacji szef ma możliwość zarządzania listą pracowników. Każdy pracownik posiada unikalny identyfikator (login), hasło, dane opisowe (imię, nazwisko) oraz status: aktywny lub nieaktywny. Na swojej stacji szef ma możliwość dodawania nowych pracowników, edycji danych opisowych pracowników oraz zmian ich statusów i haseł. Ze względu na spójność danych nie ma możliwości usuwania pracowników ani zmiany ich identyfikatorów.

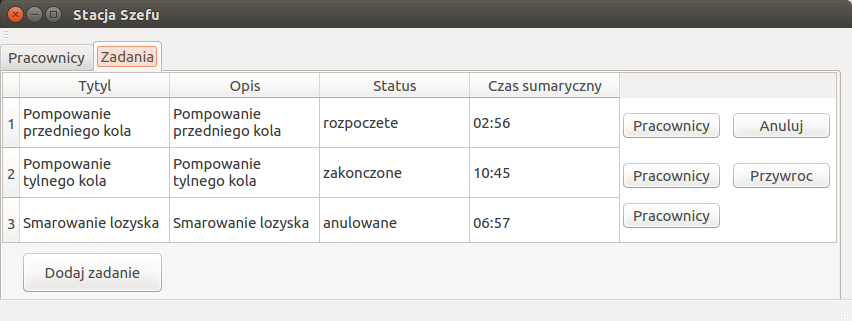


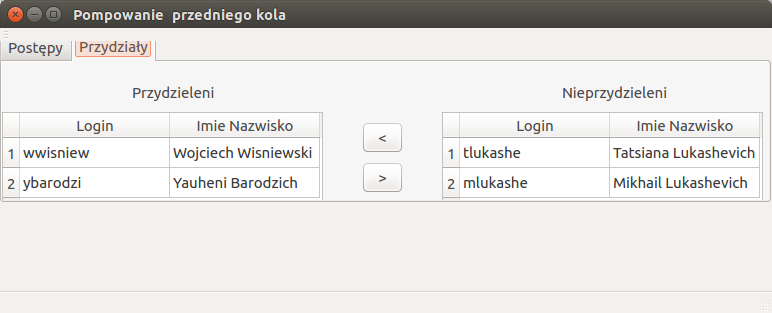
#### Edycja listy zadań

Stacja szefa umożliwia definiowanie listy zadań. Każde zadanie ma unikalny identyfikator, tytuł, opis tekstowy, status oraz listę pracowników przydzielonych do jego wykonania. Prezentowany przez system *status* zadania może przybierać wartości:

* *nierozpoczęte* (żaden z pracowników nie rozpoczął pracy nad danym zadaniem),
* *rozpoczęte* (przynajmniej jeden pracownik zaczął pracować nad zadaniem, ale żaden jeszcze go nie zakończył),
* *częściowo zakończone* (przynajmniej jeden pracownik zakończył pracę nad zadaniem),
* *zakończone* (wszyscy pracownicy zakończyli pracę)
* *anulowane* (szef anulował zadanie)

Dodatkowo przy każdym zadaniu prezentowany jest sumaryczny czas, który został spędzony na jego wykonywanie przez wszystkich pracowników, którzy kiedykolwiek nad nim pracowali. Szef może dodawać nowe zadania, a dla zadań już istniejących: edytować opisy, dodawać i usuwać pracowników przydzielonych do zadania. Jeśli zadanie nie znajduje się w stanie *zakończone,* może zostać anulowane. Anulowanie może zostać cofnięte, wtedy zadanie wraca do stanu wynikającego z postępów prac przydzielonych do niego pracowników. Zmiany w liście przydzielonych pracowników powodują ponowne obliczenie statusu zadania, np. usunięcie jedynego pracownika, który nie zakończył zadania przenosi zadanie ze stanu *częściowo zakończone* do stanu *zakończone*, nie wpływa natomiast na prezentację sumarycznego czasu. Ze względu na spójność danych nie jest możliwe usuwanie zadań.



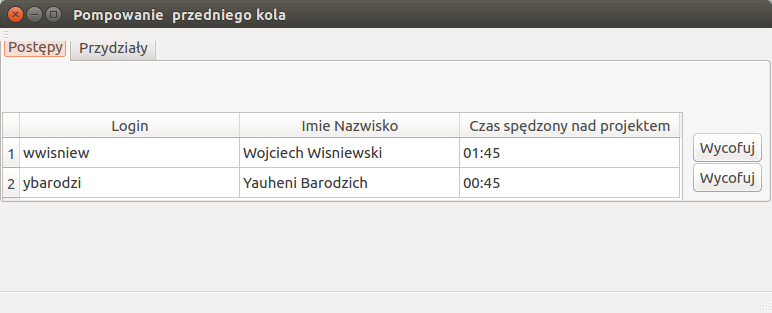


##### Prezentacja postępów prac.

Po kliknięciu na zadanie, zostanie wyświetlona lista pracowników, którzy spędzili czas pracując nad wybranym zadaniem. Dla każdego pracownika prezentowany jest: jego identyfikator, imię i nazwisko, sumaryczny czas spędzony nad zadaniem i status wykonania:

* nierozpoczęte
* rozpoczęte
* zakończone

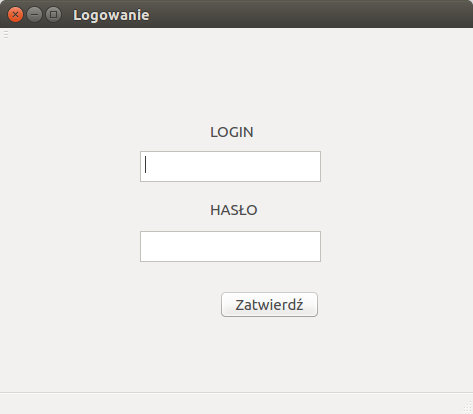
Usuwanie pracowników z listy przydzielonych do zadania nie powoduje usunięcia ich z widoku postępu prac; tacy pracownicy są tylko oznaczeni flagą.



### Stacja pracownika (aktor: pracownik)

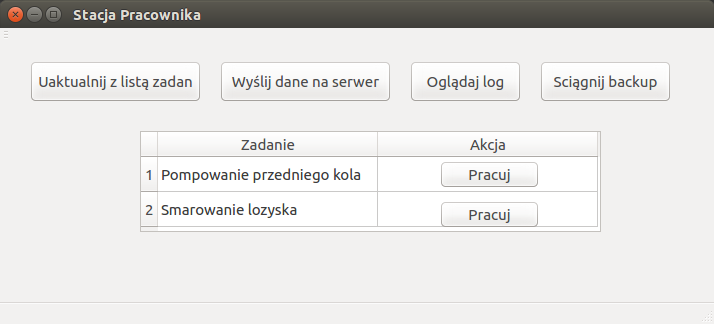
#### Logowanie i wylogowywanie

Przed przystąpieniem do pracy na stacji klienckiej pracownik musi się zalogować przy pomocy identyfikatora (loginu) i hasła. Po niepoprawnym logowaniu jest możliwość następnej próby logowania. Lista pracowników przechowywana jest na serwerze i w celu sprawdzenia poprawności pary login-hasło, stacja kliencka musi skonsultować się z serwerem. Dany pracownik może być zalogowany tylko na jednej stacji jednocześnie. Ponieważ stacja szefa nie musi działać cały czas, nie ma możliwości zapobiegania podwójnemu logowaniu (scenariusz: pracownik A loguje się na stacji X, restartujemy stację szefa, pracownik A loguje się na stacji Y), sytuacja taka powinna być jednak wykrywana i raportowana.



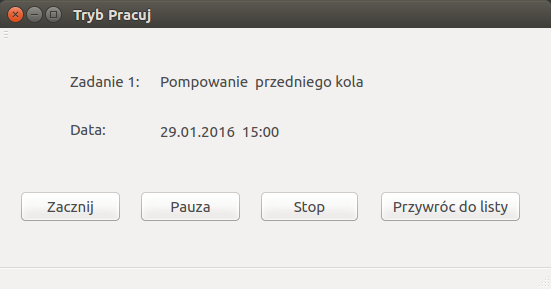
#### Pobieranie listy zadań

Po poprawnym zalogowaniu stacja kliencka samoczynnie pobiera aktualną listę zadań dostępnych dla pracownika. Dodatkowo, pracownik może manualnie zażądać uaktualnienia listy zadań w każdym momencie.



#### Rejestracja postępów pracy

Pracownikowi prezentowana jest lista zadań, nad którymi może pracować (nieanulowane zadania, do których jest przydzielony, a których jeszcze nie zakończył). Pracownik wybiera zadanie do wykonania. Po wybraniu prezentowany jest licznik sumarycznego czasu spędzonego nad zadaniem. Na początku licznik jest zatrzymany, można go uruchomić (opcja “start”), tymczasowo zatrzymać (opcja ”pauza”’), zdeklarować zakończenie zadania (opcja “stop”), lub powrócić do listy zadań. Powrót do listy zadań automatycznie zatrzymuje licznik.

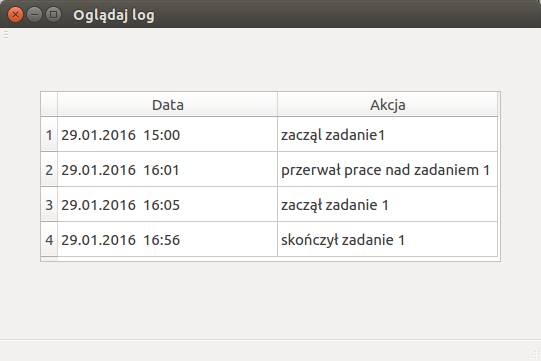


#### Przegląd historii zdarzeń

Stacja pracownika umożliwia przegląd historii operacji wykonywanych na stacji. Rejestrowane zdarzenia to:

* zalogowanie pracownika
* rozpoczęcie pracy nad zadaniem
* przerwanie pracy nad zadaniem
* zakończenie pracy nad zadaniem
* wylogowanie pracownika

Każde zdarzenie jest opatrzone punktem czasowym i identyfikatorem użytkownika. Zdarzenia dotyczące zadań opatrzone są dodatkowo tytułem zadania.



#### Odtwarzanie kopii zapasowej historii zdarzeń

Pracownik może zażądać odtworzenia historii operacji z serwera. Stacja kliencka pyta pracownika o chwilę czasową, od której historia powinna zostać pobrana. Przy odtwarzaniu kopii zapasowej historia zdarzeń zarejestrowana na stacji jest zastępowana historią pobraną z serwera.

# Wymagania niefunkcjonalne

System powinien być odporny na przerwanie komunikacji w dowolnym momencie. Przerwanie połączenia sieciowego nie powinno powodować awarii stacji klienta ani stacji serwera. Awaria stacji klienta nie powinna powodować utraty danych, w szczególności awaria nie powinna powodować “zapominania” długich okresów pracy nad zadaniem (dopuszczalna utrata najwyżej kilku minut) .Serwer powinien mieć możliwość obsługi wielu połączeń klienckich jednocześnie.

# Środowisko sprzętowo-programowe

Zarówno stacja klienta, jak i stacja serwera będą działającymi pod kontrolą systemu operacyjnego Linux i udostępniającymi GUI w systemie X Window. Obie aplikacje będą napisane w języku C++14. GUI będzie zaimplementowane przy pomocy biblioteki Qt5. Oba programy dane nieulotne przechowują w relacyjnej bazie danych. Jako silnik bazy danych została wybrana biblioteka SQLite.

# Architektura

## Podział na moduły

### Stacja szefa

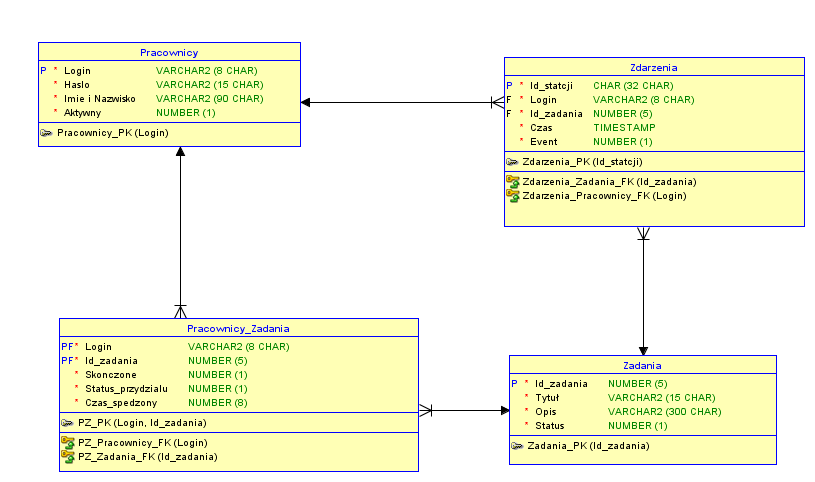
Program stacji szefa składa się z modułu GUI, zaimplementowanego przy pomocy biblioteki Qt5 i modułu komunikacyjnego (serwerowego). Moduł GUI współpracuje z bazą danych przy pomocy modułów wbudowanych w bibliotekę Qt (moduł QtSql). Moduł komunikacyjny działa w tle (w osobnych wątkach) i komunikuje się z modułem GUI tylko za pomocą bazy danych.

### Stacja pracownika

Architektura stacji pracownika jest podobna, jak w przypadku stacji szefa: składa się z modułu GUI i modułu komunikacyjnego (serwerowego). Podstawową różnicą w stosunku do stacji szefa jest konieczność ściślejszej komunikacji między modułem GUI a modułem komunikacyjnym: w przypadku stacji szefa GUI może zlecać zadania modułowi komunikacyjnemu i odbierać ich wyniki. W tym celu moduł komunikacyjny będzie posiadał kolejkę komunikatów w celu przyjmowania zleceń. Przekazywanie wyników do modułu GUI będzie zaimplementowane przy pomocy systemu sygnałów i slotów biblioteki Qt.

## Model danych

### Stacja szefa



**Tabela Pracownicy** - opisuje pracownika, któremu możemy przydzielić zadanie

* *Login -* rozróżniamy każdego pracownika za pomocą loginu przydzielonego przez szefa
* *Haslo -* każdy pracownik ma swoje hasło, za pomocą którego loguje się
* *Imie i Nazwisko* - informacja o pracowniku: jego imię i nazwisko
* *Aktywny -* pracownik może mieć dwa stany, które określa szef:1- aktywny (ma pozwolenie na wykonywanie zadań), 0 - nieaktywny(nie posiada pozwolenia)

**Tabela Zadania** - opisuje zadanie, które przedzielamy pracownikowi

* *Id\_zadania* - numer zadania służy do rozróżniania zadań między sobą
* *Tytul* - nazwa zdania
* *Opis -* opis zdania dla lepszego zrozumienia
* *Status* - 0-aktywne, 1-zakończone, 2-anulowane

**Tabela Pracownicy\_Zadania** - tabela pomocnicza, opisująca poszczególne zadanie przydzielone do pracownika

* *Login* - dany pracownik, który jest przydzielony do określonego zadania
* *Id\_zadania -* określone zadanie, do którego jest przydzielony dany pracownik
* *Skonczone* - zadanie może być skończone przez pracownika: 1, lub nieskończone: 0
* *Status\_przydzialu -* powiązanie między pracownikiem a zadaniem może być: 1 - aktywne (pracownik może dalej pracować nad zadaniem, o ile go nie ukończył) i 0 - nieaktywne (pracownik nie może już pracować nad zadaniem, a status zakończenia przez niego zadania nie wpływa już na sumaryczny status zadania)
* *Czas\_spedzony* - łączny czas spędzony nad określonym zadaniem przez pracownika

**Tabela Zdarzenia** - opisuje zdarzenia na stacji szefa

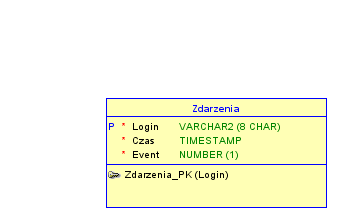
* *Id\_stacji* - numer stacji pracownika
* *Login* - określa konkretnego pracownika
* *Id\_zadania* - numer zadania, którego dotyczy zdarzenie
* *Czas* - czas rozpoczęcia zdarzenia
* *Event* - może przyjmować następujące wartości: 1 - zalogowanie, 2- rozpoczęcie pracy nad zadaniem, 3- przerwanie pracy nad zadaniem, 4- zakończenie pracy nad zadaniem, 5- wylogowywanie, 6 - punkt kontrolny (kontynuacja pracy nad zadaniem): dodawane automatycznie co jakiś czas, zapobiega utracie czasu spędzonego przez pracownika w przypadku awarii stacji

**Mapowanie stanu bazy danych na stan zadania prezentowany w widoku zadań**. Na status zadania mają wpływ tylko pracownicy aktywni, dla których istnieje aktywny przydział do tego zadania w tabeli Pracownicy\_Zadania. Pracowników tych nazywamy pracownikami istotnymi. Rozpatrujemy w kolejności następujące warunki:

* Jeśli zadanie ma status “anulowane” lub “zakończone”, prezentujemy taki właśnie status
* Jeśli zadanie ma status “aktywne”:
  + Jeśli zbiór pracowników istotnych jest pusty, prezentujemy status “nierozpoczęte”
  + Jeśli wszyscy istotni pracownicy zakończyli zadanie, prezentujemy status “zakończone”, a w tabeli zadań zmieniamy status na zakończone
  + Jeśli przynajmniej jeden istotny pracownik zakończył zadanie, prezentujemy status “częściowo zakończone”.
  + Jeśli przynajmniej jeden pracownik zarejestrował niezerowy czas pracy nad zadaniem, prezentujemy status “rozpoczęte”
  + W pozostałym przypadku prezentujemy status “nierozpoczęte”.

Zmiany statusów pracowników i przydziałów zadań powodują ponowne przeliczenie stanów zadań.

### Stacja pracownika



**Tabela zdarzenia** - opisuje zdarzenia na stacji klienta

* *Login* - za pomocą loginu, rozróżniamy pracowników
* *Czas* - czas rozpoczęcia zadania
* *Event* - ma następujące stany: 1 - zalogowanie, 2- rozpoczęcie pracy nad zadaniem, 3- przerwanie pracy nad zadaniem, 4- zakończenie pracy nad zadaniem, 5- wylogowywanie

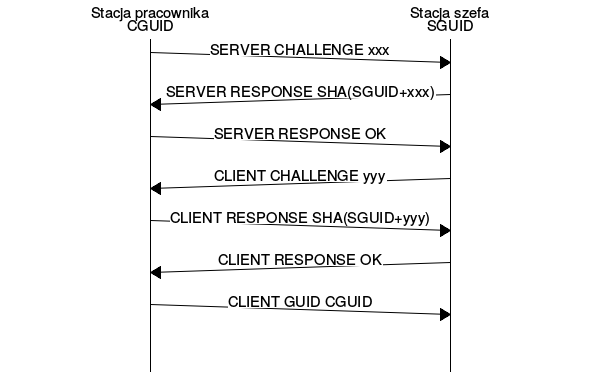
## Protokół komunikacyjny

### Założenia ogólne

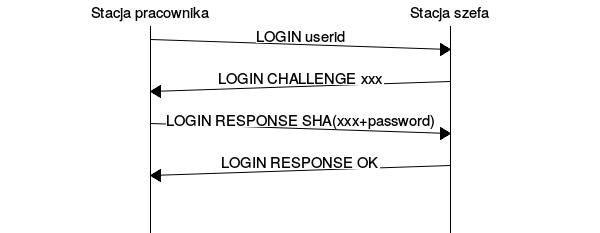
Stacje współpracują w schemacie klient-serwer. Stacja szefa jest serwerem, a stacje pracowników - klientami. Jako protokół warstwy transportowej będzie użyty TCP. Wszelka komunikacja jest inicjowana przez klienta: klient nawiązuje połączenie i wysyła żądania do serwera, serwer oczekuje na połączenia od klientów i ich zlecenia. Dla łatwości debugowania przyjmujemy, że wszystkie komunikaty będą przesyłane w postaci tekstowej.

### Autentykacja serwera i klienta

Zanim klient i serwer wejdą w interakcję, powinny się zautentykować się wzajemnie. Autentykacja składa się z dwóch etapów: potwierdzenie tożsamości serwera względem klienta oraz potwierdzenie uprawnień stacji klienckiej do komunikacji z serwerem. Stacja kliencka i serwer potwierdzają swoją tożsamość na podstawie wspólnego sekretu: GUIDu serwera. Autentykacja następuje na zasadzie challenge-response: najpierw klient stawia wyzwanie serwerowi w postaci losowego ciągu znaków, serwer odsyła skrót SHA konkatenacji tego ciągu z GUIDem serwera. Następnie w analogiczny serwer stawia wyzwanie klientowi. Po wzajemnej autentykacji klienta i serwera, klient przedstawia się serwerowi przy pomocy swojego GUID.

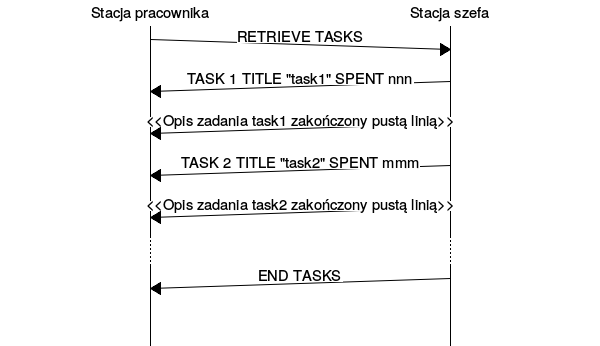


### Logowanie pracownika

Po wzajemnej autentykacji stacji następuje weryfikacja tożsamości pracownika przy pomocy jego identyfikatora i hasła. Podobnie jak w przypadku wzajemnej autentykacji stacji, nie chcemy przesyłać tajnego hasła poprzez sieć, więc autentykacja następuje w schemacie challenge-response.

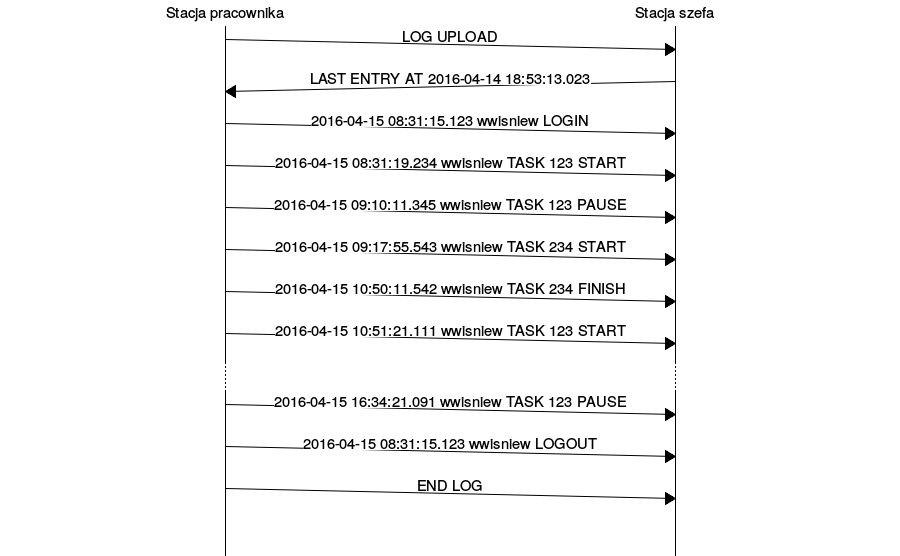
### Pobieranie listy zadań

Stacja kliencka, tuż po poprawnym zalogowaniu (automatycznie) lub później na żądanie pracownika wysyła do serwera żądanie pobrania listy zadań dla aktualnie zalogowanego pracownika. Serwer odpowiada, wysyłając listę zadań, nad którymi może pracować pracownik (tych, do których jest przydzielony, które nie są anulowane i pracownik ich nie zakończył). Dla każdego zadania przesyłany jest: identyfikator, tytuł, dotąd spędzony nad nim przez pracownika czas i opis. Niepowodzenie (kolejnego) pobrania listy zadań z serwera nie powinno wpływać na pracę stacji klienckiej: pracownik w dalszym ciągu powinien mieć możliwość pracy nad pobranymi wcześniej zadaniami.

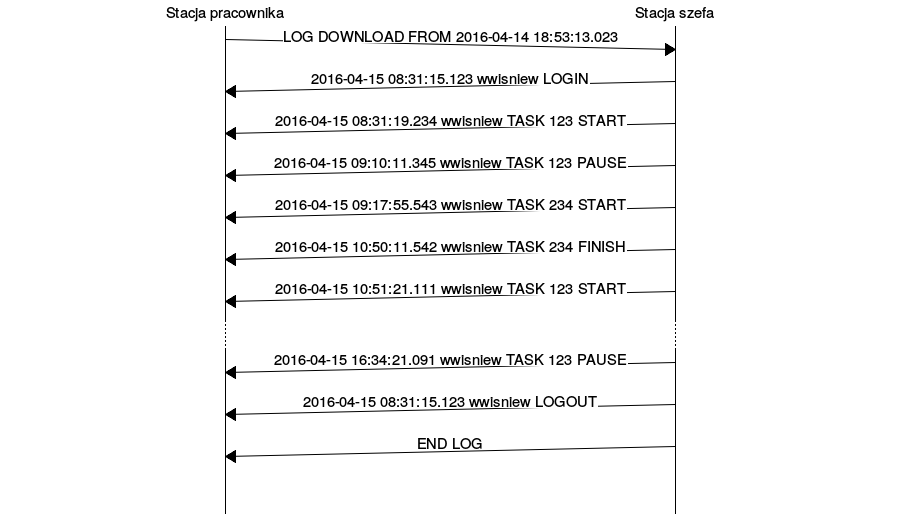


### Wysyłanie historii zdarzeń

Stacja pracownika periodycznie próbuje wysłać na serwer zarejestrowaną historię zdarzeń. Wysyłanie to odbywa się w tle, bez interakcji z użytkownikiem. Dodatkowe wysyłki mogą być wymuszane przez pracownika poprzez GUI. Stacja pracownika inicjuje wysyłanie, następnie serwer odpowiada podając czas najnowszego zarejestrowanego przez siebie zdarzenia ze stacji klienckiej, a stacja kontynuuje dialog, wysyłając kolejno opisy zdarzeń późniejszych niż podana chwila. Niepowodzenie wysłania historii nie powinno wpływać na pozostałą funkcjonalność stacji klienckiej.



### Pobieranie backupu historii zdarzeń

W sposób analogiczny do wysyłania logu zdarzeń na serwer, stacja kliencka może zażądać pobrania logu zdarzeń z serwera (odtworzenia backupu). Tym razem stacja kliencka podaje czas ostatnio zarejestrowanego zdarzenia, a serwer odpowiada listą logów w formacie identycznym jak w przypadku wysyłania logów na serwer.

# Sposób testowania

Niektóre części projektu są bardziej, niektóre mniej podatne na testowanie jednostkowe. Planujemy testy jednostkowe dla logiki aplikacji (operacje na bazie danych) oraz testy jednostkowe dla interface między modułem komunikacyjnym a GUI w stacji pracownika. Częściowo podatny na testowanie jednostkowe jest też protokół komunikacyjny. Dodatkowo planujemy oddawanie poszczególnych części protokołu komunikacyjnego jednocześnie dla klienta i serwera, aby ułatwić testy części komunikacyjnej. Testy dla GUI pozostają tylko manualne. Ostateczne testowanie aplikacji nastąpi przy pomocy manualnych testów integracyjnych, których scenariusze znajdują się poniżej. Prawdopodobnie opracujemy dodatkowe scenariusze w fazie testowania.

## Scenariusze testów akceptacyjnych

**Scenariusz 1 (praca normalna)**: stacja szefa i dwie stacje klienckie: stacja X komunikuje się przy pomocy protokołu IPv4, a stacja Y IPv6. Stacja szefa ma dostęp do obu protokołów.

* 1. Tworzymy dwóch pracowników A i B. Tworzymy zadana 1 i 2. Pracownicy powinni pojawić się na liście pracowników, a zadania - na liście zadań.
  2. Przydzielamy pracownika A do zadania 1 i 2, a pracownika B do zadania 2. Dla obu zadań powinno być: status: “nierozpoczęty” a czasy: sumaryczny i spędzony przez każdego pracownika: 0.
  3. Na stacji X logujemy się jako pracownik A a na stacji Y jako pracownik B. Na każdej stacji podejmujemy dwie próby logowania: pierwsza z błędnym hasłem, druga z prawidłowym. Po poprawnym zalogowaniu pracownik A powinien widzieć dostępne zadania 1 i 2, a pracownik B zadanie 2.
  4. Pracownik A rozpoczyna pracę nad zadaniem 1, pracuje przez chwilę i wymusza wysłanie logu na serwer. Na serwerze status zadania powinien zmienić się na “rozpoczęte”, a sumaryczny czas odpowiadać czasowi naliczonemu pracownikowi A.
  5. Pracownik A przełącza się na zadanie 2, pracuje przez chwilę i wymusza wysłanie logu na serwer. Na serwerze status zadania 2 powinien zmienić się na “rozpoczęte”, a sumaryczny czas obu zadań powinien odpowiadać czasowi naliczonemu dla nich pracownikowi A.
  6. Pracownik A kończy zadanie 2 i wymusza wysłanie logu na serwer. Na serwerze zadanie 2 przechodzi w stan “częściowo zakończone”, a sumaryczny czas zostaje uaktualniony do czasu naliczonego dla zadania 2 pracownikowi A.
  7. Pracownik A wylogowuje się.
  8. Pracownik B zaczyna pracę nad zadaniem 2, pracuje przez chwilę, następnie przerywa pracę i wymusza wysłanie logu na serwer. Na serwerze sumaryczny czas dla zadania 2 powiększa się o czas spędzony przez pracownika B.
  9. Na serwerze ustawiamy zadanie 2 na anulowane. Pracownik B odświeża listę zadań. Nie powinno być dostępne żadne zadanie.
  10. Na serwerze ustawiamy przywracamy zadanie 2. Jego status powinien zmienić się na “częściowo zakończone”. Pracownik B odświeża listę zadań. Do wyboru ma tylko zadanie 2. Pracuje nad nim przez chwilę, następnie przerywa pracę i wymusza wysłanie logu na serwer. Sumaryczny czas spędzony nad zadaniem 2 powiększa się o nowy odcinek czasu spędzony przepracowany przez pracownika B.
  11. Na serwerze usuwamy pracownika B z listy pracowników przydzielonych do zadania 2. Status powinien zmienić się na “zakończone”. Pracownik B odświeża listę zadań. Nie powinien mieć żadnych zadań do wyboru.
  12. Dodajemy z powrotem pracownika B do listy pracowników przydzielonych do zadania 2. Zadanie powinno zmienić status na “częściowo zakończone”. Pracownik B odświeża listę zadań. Powinno pojawić się zadanie 2. Pracownik B wybiera to zadanie, pracuje przez chwilę, kończy zadanie i wylogowuje się ze stacji Y. Zadanie 2 powinno zmienić status na “zakończone”, a jego czas sumaryczny powinien powiększyć się o odcinek czasu przepracowany przez pracownika B.
  13. Pracownik A loguje się na stacji Y. Powinien mieć do wyboru tylko zadanie 1. Zaczyna pracę nad zadaniem. Naliczanie czasu powinno kontynuować od wartości zarejestrowanej na stacji X. Pracownik A kończy zadanie 1 i wylogowuje się. Na stacji szefa zadanie 1 powinno przejść w stan “zakończone”, a czas sumaryczny powinien zostać uaktualniony.
  14. Dodajemy pracownika B do zadania 1. Zadanie powinno zmienić status na “częściowo zakończone”. Pracownik B loguje się na stacji X. Powinien mieć dostępne zadanie 1. Wybiera je, pracuje przez chwilę, następnie kończy zadanie i wylogowuje się. Na stacji szefa sumaryczny czas zadania 1 uaktualnia się i przechodzi ono w stan “zakończone”.

**Scenariusz 2 (praca autonomiczna stacji pracownika)**- użytkownik na stacji klienta loguje się pobiera zadania, zaczyna wykonywać. Stacja szefa zostaje wyłączona

* 1. Dodajemy pracowników i zadania, jak w scenariuszu 1
  2. Pracownik A loguje się na stację i pobiera swoje zadania i rozpoczyna pracę nad jednym z nich.
  3. Stacja szefa została wyłączona
  4. Pracownik przerywa pracę nad zadaniem i próbuje pobrać nowe zadania. Operacja pobrania nie udaje się, ale pracownik ma możliwość kontynuacji pracy nad pobranymi wcześniej zadaniami.
  5. Pracownik kończy pracę nad zadaniem i próbuje wysłać log na stację szefa. Operacja wysłania logu nie udaje się, ale pracownik ma możliwość kontynuacji pracy.
  6. Włączamy stację szefa i definiujemy nowe zadanie i przydzielamy pracownika A do tego zadania
  7. Pracownik odświeża listę zadań i wymusza wysłanie logu. Pracownik widzi nowe zadania. Statusy i czasy zadań na serwerze uaktualniają się.

**Scenariusz 3 (pobieranie backupu)**

* 1. Dodajemy pracowników i zadania,logujemy się jak w scenariuszu 1.
  2. Pracownik pobiera listę zadań i wykonuje je.
  3. Pracownik przegląda listę zdarzeń. Czynności wykonywane w punkcie b Scenariusza 3 powinny być odzwierciedlone na liście.
  4. Pracownik wylogowuje się i wyłącza stację.
  5. Kasujemy plik z historią zdarzeń.
  6. Włączamy stację i logujemy się.
  7. Pobieramy backup.
  8. Oglądając historię zdarzeń, powinna wyglądać jak w punkcie c z dodatkowym zdarzeniem ostatniego logowania.

# Podział pracy

|  |  |
| --- | --- |
| Tatsiana Lukashevich | dokumentacja, GUI stacji pracownika, |
| Yauheni Barodzich | moduł komunikacyjny stacji klienta |
| Mikhail Lukashevich | moduł komunikacyjny stacji szefa |
| Wojciech Wiśniewski | *Lider projektu* GUI stacji szefa, integracja modułów, plugin do Wiresharka |

# Harmonogram

|  |  |
| --- | --- |
| Do 18 kwietnia | wysłanie projektów wstępnych |
| Do 25 kwietnia | publiczne omówienie projektów wstępnych i zaawansowania implementacji |
| Do 25 kwietnia | komunikacja: autentykacja stacji (klient+serwer) |
| Do 28 kwietnia | komunikacja:logowanie użytkownika (klient+serwer) |
| Do 5 maja | GUI stacji szefa i GUI stacji pracownika  komunikacja: pobieranie listy zadań (klient+serwer) |
| Do 12 maja | komunikacja: wysyłanie i pobieranie logu (klient+serwer) |
| Do 16 maja | integracja GUI z komunikacją |
| Do 29 maja | testowanie i optymalizacja programu |
| 30 maja | wysłanie projektu końcowego |
| Do 6 czerwca | publiczna prezentacja projektu |

# Adres projektu na GitHub

https://github.com/Meehanick/-TIN-Karbowy