Dokumentacja techniczna medirc

System telekonsultacyjny

Spis treści

Możliwości systemu

System kont użytkowników

- Automatyczna rejestracja – jeśli nie istnieje użytkownik z podanym loginem to zostanie automatycznie utworzony
- Konta użytkowników są przechowywane w bazie danych z odpowiednio zabezpieczonym hasłem

Aktywne sesje

- Każdy użytkownik może utworzyć dowolną ilość sesji
- Przechowywane w bazie danych - przebieg sesji nie jest tracony w przypadku wyłączenia serwera
- System zaproszeń, możliwość przyjęcia i odrzucenia zaproszeń – powiadamianie innych użytkowników sesji o przyjęciu/odrzuceniu zaproszenia
- 3 stany sesji
    - Przygotowania – właściciel może przygotować sesje zanim inni otrzymają do niej dostęp
    - W trakcie – dostęp dla wszystkich uczestników
    - Zakończona – zostaje automatycznie przeniesiona do archiwum, nie można już

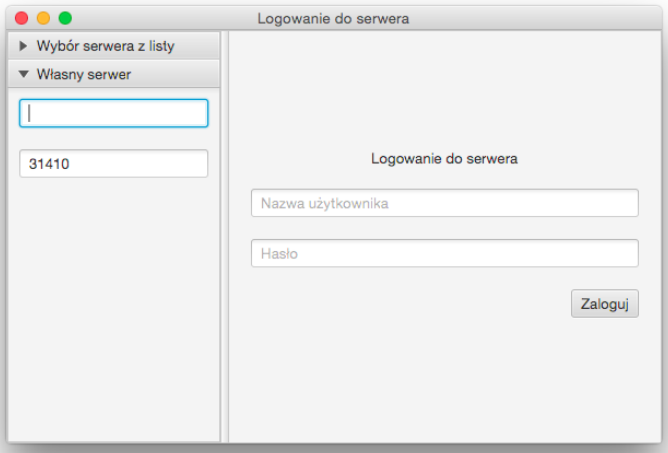
wykonywać żadnych akcji

- Grupowy czat tekstowy
- Właściciel ma możliwość wyrzucenia dowolnego uczestnika
- Dostępna lista aktywnych oraz zaproszonych użytkowników sesji
- Wrzucanie oraz usuwanie plików graficznych przez właściciela sesji
    - Obsługa zoom'owania obrazków (SHIFT + Scroll)
    - Przesuwanie (suwaki oraz prawy/środkowy przycisk myszy)
    - Zaznaczanie fragmentów obrazka – zaznaczenia są widoczne dla wszystkich
        - Wybór koloru przez użytkownika (domyślnie jest ustawiana odwrotność koloru obrazka)
    - Usuwanie własnych zaznaczeń fragmentów – właściciel może usunąć wszystkie
    - Opcja zwrócenia uwagi – wszystkim uczestnikom otwiera się ten sam obrazek z tym samym widocznym fragmentem
    - Tylko właściciel oraz użytkownicy z prawem głosu mogą wykonywać manipulacje
- Opcja ręcznego nadawania/odbierania głosu przez właściciela
- Opcja automatycznego nadawania głosu – opcjonalna
    - System kolejkowania – kto pierwszy ten lepszy
    - Głos jest nadawany w momencie oddania głosu przez poprzedniego użytkownika

Archiwum sesji

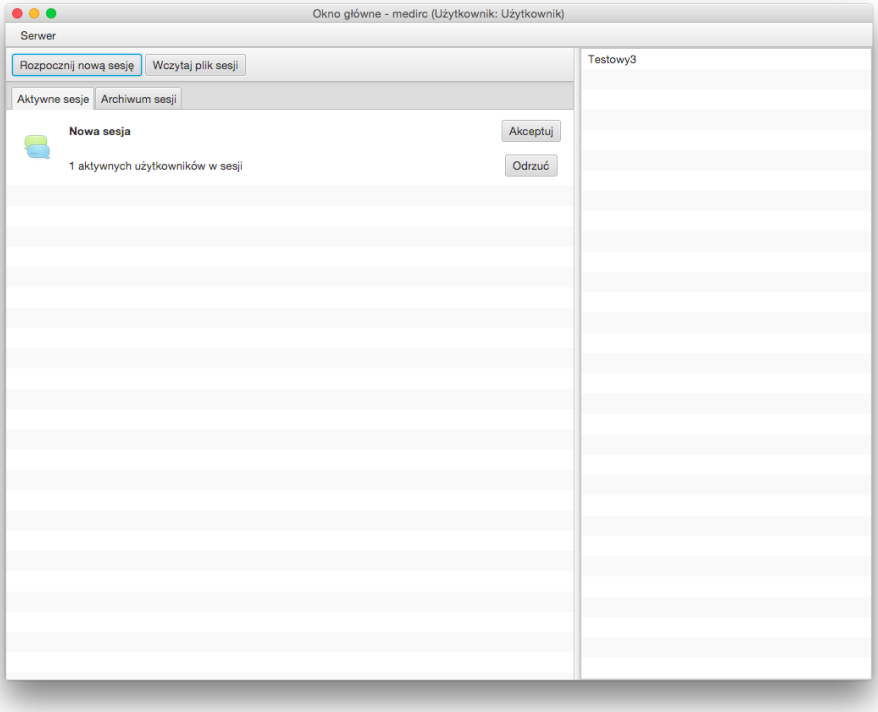
- Wszystkie wydarzenia zaszłe w sesji są zapamiętywane w bazie danych
- W momencie zmiany statusu sesji na zakończony wszyscy uczestnicy (w momencie zakończenia) mają dostęp do pobrania pełnego przebiegu sesji z poziomu listy archiwalnych sesji w głównym oknie programu
- Pobrany plik sesji umożliwia nam odtworzenie pełnego jej przebiegu
- W trakcie odtwarzania przebiegu można w dowolnym momencie spauzować odtwarzanie
- Dzięki linii czasu można przenieść się do dowolnego momentu sesji – wstecz oraz naprzód

Interfejs graficzny



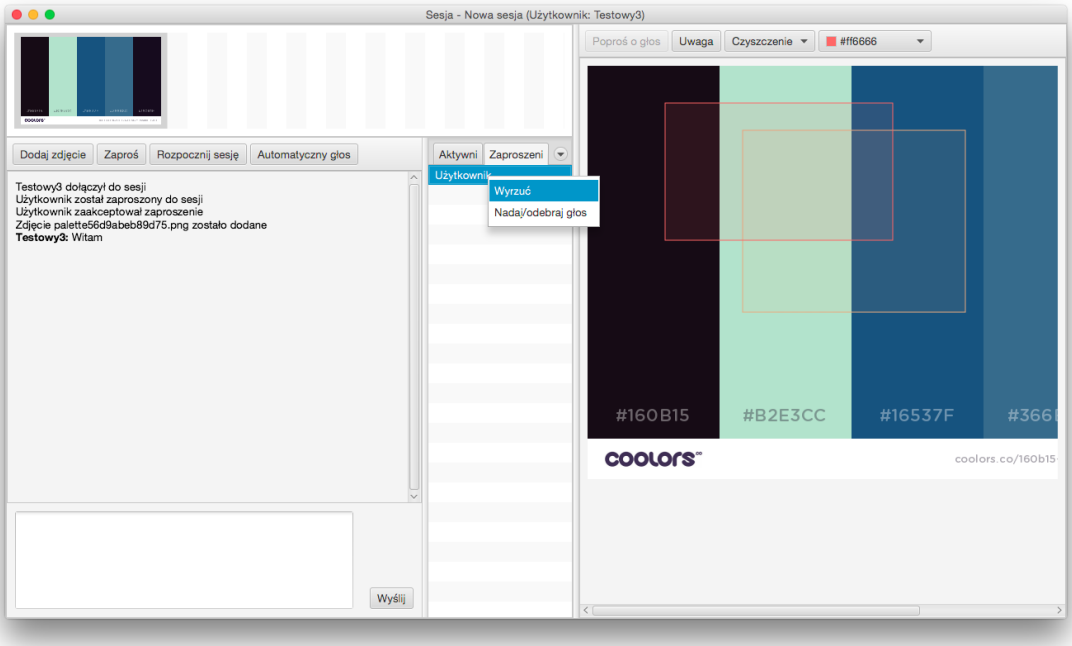
1: Logowanie do serwera

Umożliwia nam zalogowanie się do dowolnego serwera używając podanego loginu oraz hasła



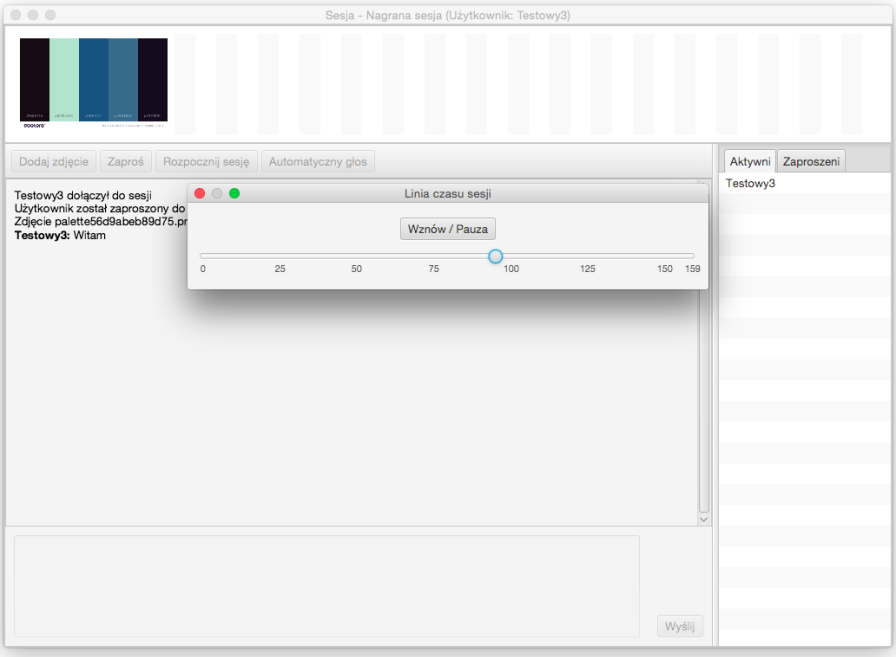
2: Główne okno aplikacji

Główne okno umożliwia nam podejrzenie listy wszystkich sesji do których mamy dostęp, utworzenie nowej sesji oraz zarządzanie naszymi zaproszeniami. Jest również możliwość pobrania archiwalnych sesji oraz podejrzenia listy użytkowników obecnie zalogowanych.



3: Aktywna sesja

Najważniejsze okno programu reprezentujące aktywną sesję. Dostępny jest wygodny czat tekstowy, oraz opcje zarządzania sesją – w tym dodawanie zdjęć czy aktywacja automatycznego systemu nadawania głosu. W lewym górnym rogu jest lista miniaturek zdjęć. Kliknięcie na dowolne zdjęcie powoduje otworzenie prawego panelu zawierającego jego pełną wersję wraz z opcjami manipulacji.



4: Odtwarzanie archiwalnej sesji

Możliwość obejrzenia całego przebiegu sesji. W dowolnym momencie możemy użyć opcji pauzy oraz przenieść się do wybranego punktu na linii czasu sesji.

Architektura systemu

System jest oparty na tradycyjnej architekturze z centralnym serwerem i wieloma klientami podłączonych do niego. Zdecydowana większość pracy jest wykonywana przez serwer, klient jedynie wykonuje polecenia serwera i wysyła zapytania użytkownika.

Komunikacja klienta z serwerem

- TCP/IP
- Pakiety w formie [Długość] [ID typu] [Ładunek]
- Ładunek jest wiadomością Google Protocol Buffers
- ID typu służy nam do określenia z jaką wiadomością mamy do czynienia – Google Protocol Buffers nie ma wbudowanego mechanizmu do tego celu
- Wspólny moduł protokołu współdzielony przez klienta oraz serwer. Zawiera wiadomości zdefiniowane w formacie Google Protocol Buffers, które są automatyczne kompilowane do plików Javy dzięki odpowiedniej konfiguracji systemu budowania. Moduł ten zawiera również mapowanie ID typu → Wiadomość oraz komponent dekodera/kodera dla frameworka Netty
- Rodzaje pakietów:
  - Zapytania klienta do serwera
  - Odpowiedź na zapytania klienta – w większości jest to po prostu informacja o powodzeniu lub błędzie (parę zapytań jest bez odpowiedzi ze względu na brak potrzeby – np. operacja zawsze się powiedzie)
  - Powiadomienia klienta o wydarzeniach (np. zmiany na liście sesji, wszystkie operacje w sesji)
- Przykładowa sekwencja wiadomości 1 klient ↔ serwer (K: wiadomość wysyłana przez klienta, S: przez serwer)
  - K: Handshake – dane do logowania
  - S: HandshakeAck – serwer potwierdza poprawność danych, wysyła ID i nick
  - K: SyncRequest – zapytanie o podesłanie aktualnej listy sesji aktywnych i archiwalnych oraz listy aktywnych użytkowników w celu synchronizacji
  - S: ActiveSessions, ArchivedSessions, UserList – dane, o które było zapytanie
  - K: CreateNewSession – zapytanie o utworzenie nowej sesji o danej nazwie
  - S: NewSessionResponse – odpowiedź serwera informująca o sukcesie

- S: SessionInvite – informacja o pojawieniu się nowej sesji na liście
- K: JoinRequest – zapytanie o dołączenie do nowo utworzonej sesji
- S: JoinResponse – odpowiedź z informacją o sukcesie i danych sesji
- S: Joined – wydarzenie dołączenia użytkownika
- K: UploadImage – dodanie obrazka do sesji przez użytkownika
- S: UploadImageResponse – informacja o sukcesie zapytania
- S: ImageAdded – informacja o nowym obrazku (bez samej zawartości obrazka)
- K: RequestImage – prośba o podesłanie zawartości obrazka
- S: RequestImageResponse – podesłanie zawartości obrazka
- […] - sekwencja zapytań, odpowiedzi i wydarzeń danej sesji
- S: SettingsChanged – wydarzenie informujące o zmianach ustawienia sesji (w tym wypadku zmiana stanu sesji na zakończoną)
- S: SessionUpdated – wydarzenie aktualizacji sesji na liście (w tym wypadku archiwizacja)
- K: DownloadSession – zapytanie pobrania pliku sesji z serwera
- S: DownloadSessionAcknowledge – serwer potwierdza, wysyła rozmiar bloku, ilość bloków, które podeśle
- S: DownloadSessionBlock – serwer wyśle wiele bloków z danymi danej sesji, które klient musi połączyć w jedną całość

Klient

- Interfejs zadeklarowany w formacie FXML
- Podział paczek:
  - components – zawiera niestandardowe “kontrolki”, w tym
    - element listy sesji
    - element listy obrazków
    - edytor obrazków wspierający zaznaczanie oraz zoom
  - data – różne opakowania na dane z serwera
  - image – klasy reprezentujące zaznaczenia na obrazkach

- instance - „sesja” połączenia z serwerem, różne elementy komunikacji sieciowej
  - używany framework Netty do połączenia
- stage – okienka aplikacji, najważniejsze dot. sesji:
  - AbstractSessionStage – okienko sesji, cała interakcja z użytkownikiem
  - ActiveSessionStage – okienko aktywnej sesji
  - ArchiveSessionStage – okienko sesji odtwarzanej z pliku

Serwer

- Do logów używamy biblioteki Log4J2, skonfigurowany do wysyłania logów do standardowego wyjścia
- Do rozwiązywania zależności w grafie obiektów wykorzystujemy bibliotekę Google Guice
  - Główna konfiguracja kontenera zależności jest w klasie module.ServerGuideModule
- System konfiguracji
  - Format konfiguracji to standardowe pliki properties Java
  - Zmiana portu, lokalizacji możliwa w zasobie default.properties
  - W chwili obecnej brak możliwości zmiany konfiguracji bez rekompilacji paczki (potrzeba dopisać odpowiednią implementację ConfigurationProvider)
- Baza danych
  - SQLite
  - Struktura tabel jest tworzona automatycznie, domyślnie baza danych jest zapisywana do pliku medirc.db
  - Do łączenia używamy JDBC z connector'em do SQLite
  - Struktura:
    - users – dane użytkowników, hasła przechowywane w postaci hash'a SHA512
    - sessions – podstawowe dane sesji (nazwa, właściciel, flagi właściciela, stan, ustawienia)
    - session_users – uczestnicy sesji wraz z flagami
    - images – zdjęcia sesji (również te usunięte)
    - image_fragments – obecne zaznaczenia na zdjęciach sesji
    - session_events – wszystkie zaszłe wydarzenia w sesjach

- System zdarzeń
  - Pętla zdarzeń z kolejką obsługująca sekwencyjnie wszystkie zdarzenia (generowane przez nas oraz przez Netty – czyli odbierane pakiety oraz połączenia/rozłączenia)
  - Konsumenci zdarzeń (obsługa specyficznych wydarzeń, wiadomości, wiadomości z autentykacją)
- Moduły:
  - auth – Obsługa logowania użytkowników do systemu
  - autovoice – System automatycznego nadawania głosu
  - home – Obsługa list sesji i użytkowników (czyli głównego okna w kliencie)
  - sessioninput – Waliduje zapytania klientów, wysyła odpowiedź i ewentualnie dodaje odpowiednie wydarzenie do pętli (jeśli przeszło walidacje). Obsługuje również wydarzenie rozłączenia użytkownika z serwerem.
  - sessionuserhandler – Obsługa aktywnych sesji, wysyła wydarzenia sesji do aktywnych uczestników
  - sessionrecorder – Archiwizuje wszystkie wydarzenia sesji do bazy danych
- Obsługa połączeń
  - Wykorzystanie framework'a Netty
  - Połączenia są obsługiwane asynchroniczne w osobnych wątkach, nie blokują naszej pętli wydarzeń
- Serializacja sesji
  - Zakończone sesje muszą być w postaci możliwej do przesłania użytkownikowi
  - Tworzona jest nowa baza danych SQLite
  - Kopiowane są tam wszystkie wydarzenia z danej sesji wraz z obrazkami oraz zredukowaną bazą danych uczestników (tylko nicki)
  - Struktura bazy danych dla plików archiwum:
    - users – tabela z mapowaniem ID użytkowników na nazwę
    - images – tabela z danymi wszystkich obrazków które wystąpiły w danej sesji
    - events – tabela z wszystkimi wydarzeniami które wystąpiły w danej sesji, z odpowiednim znacznikiem czasowym (czas serwera wystąpienia wydarzenia)

Budowanie, uruchamianie

Wymagania

- Java 1.8 (+ JavaFX) lub nowsza (do zbudowania będzie potrzebne JDK)
- Maven (lub IDE ze wsparciem Maven'a – IntelliJ, NetBeans itd.)

Budowanie z poziomu linii poleceń

W katalogu głównym projektu uruchomić polecenie:

```
mvn package
```

System budowania Maven pobierze wszystkie zależności oraz zbuduje aplikacje serwera oraz klienta w katalogach odpowiednio client/target oraz server/target. Pliki potrzebne do funkcjonowania aplikacji to client.jar / server.jar oraz cały katalog lib (zawierający skopiowane biblioteki).

Uruchamianie

W zależności od systemu operacyjnego zbudowaną aplikację (client.jar lub server.jar) można uruchomić przez “dwuklik” lub wydanie odpowiedniego polecenia w konsoli:

```
java -jar client/target/client.jar
java -jar server/target/server.jar
```

Serwer musi być włączony zanim spróbujemy się z nim połączyć z poziomu klienta

Realizacja projektu

Podział prac

Podział systemu na klient i serwer umożliwiał dość naturalny podział prac tym bardziej z protokołem określonym na początku prac. Niestety dość często była potrzeba zmiany w protokole, co wymuszało często jednoczesną pracę nad klientem jak i serwerem.

Wrażenia po pracy z wykorzystanymi technologiami

- IntelliJ to najlepsze IDE z jakim mieliśmy przyjemność pracować. Doskonała inspekcja kodu wykrywająca potencjonalne błędy, możliwość generowania powtarzalnych fragmentów kodu oraz świetny refactoring na pewno pozwoliły zaoszczędzić czas.
- Framework Netty umożliwił szybką i co najważniejsze niezawodną implementacje spraw związanych z komunikacją. Gdybyśmy zdecydowali się na pracę na niższym poziomie za pewne niepotrzebnie “wymyślilibyśmy koło od nowa” tracąc mnóstwo czasu.
- Rzuciło się dużo niedociągnięć w Javie, takich jak brak możliwości automatycznego określania typu przez kompilator (auto w C++, var w C#) często było uciążliwe. Okazało się również, że wbudowane biblioteka do wczytywania obrazków w Javie ma różne problemy z zapisem obrazków w formacie JPEG co zmusiło nas do zapisywania w formacie PNG (które zajmują więcej miejsca).

Napotkane trudności

- Implementacja manipulacji obrazków

- Początkowo były próby użycia transformacji JavyFX na wbudowanym komponencie ImageView. Było z tym dużo problemów, które zapewne wynikały z braku doświadczenia. Ostatecznie napisaliśmy własny komponent, wykorzystujący Canvas.
- Niestety nadal były problemy z rotacją, a dokładniej problem z dopasowywaniem się rozmiaru obszaru obrazka po rotacjach. Dodatkowo znacznie skomplikowałaby nasz kod dotyczący zaznaczania fragmentów przez użytkowników. Zdecydowaliśmy się z niej zrezygnować.