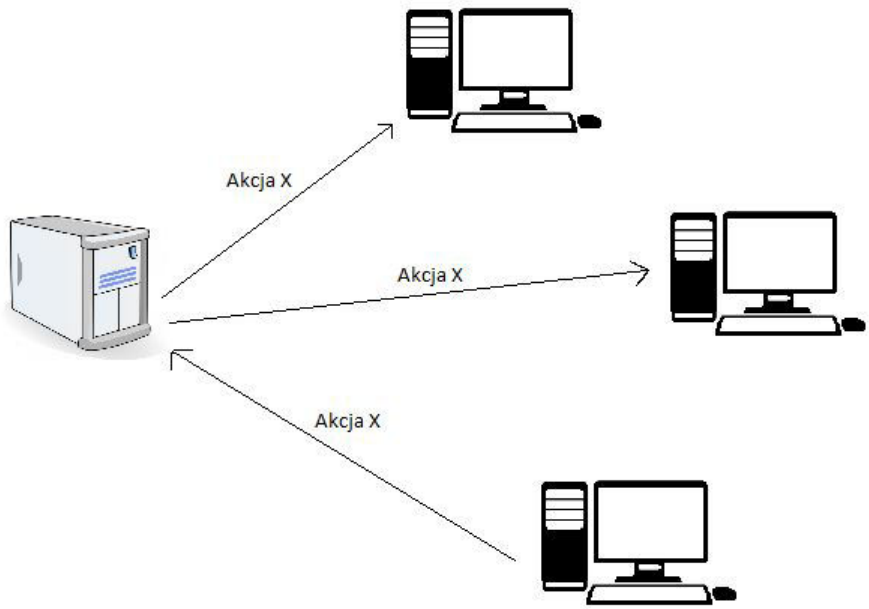
1. Opis problemu

Celem projektu było stworzenie systemu, który pozwalałby na bezproblemową konsultację pacjent-lekarz lub lekarz-lekarz. System taki powinien zapewniać podstawowe funkcje umożliwiające swobodną komunikację oraz wizualny przekaz informacji. Główną funkcjonalnością, którą miał wspierać system było zdalne przeglądanie zdjęć, manipulowanie nimi np. powiększanie oraz ich tagowanie przez wiele osób naraz.

2. Projekt systemu

System został oparty na architekturze klient-serwer. Zastosowanie takiego podziału eliminuje wiele niepożądanych zdarzeń np. ogranicza nieprzewidziane przypadki tj. manipulowanie zapytaniami sql przez klienta, rozwiązuje problem synchronizacji oraz umożliwia łatwiejszą rozbudowę aplikacji w przyszłości. Jednym z powodów, dla którego zdecydowaliśmy się ostatecznie na zastosowanie architektury klient-serwer, a nie w pełni rozproszonej, jest prędkość łącza. Taki system ma ułatwiać pracę lekarzy, konsultantów medycznych, więc nie można pozwolić na to, aby na przykład stracone zostały pewne informacje. Osoby korzystające z takiego systemu nie powinny się zastanawiać, czy dane które chciały przekazać pacjentowi, zostały na komputerze w prywatnym gabinecie, czy na komputerze w szpitalu. Ponadto nie zawsze da się zagwarantować wystarczającą prędkość łącza, aby klient mógł rozsyłać tą samą informację do wielu osób naraz. Wysłanie obrazu do 10 pozostałych członków konferencji może być kosztowne i czasochłonne. Lepiej przesłać ten obraz na serwer i przerzucić na niego operację rozesłania obrazu, ponieważ gwarantuje on szybkie łącze i zmniejsza prawdopodobieństwo zerwania połączenia podczas wysyłania danych.



Projekt realizuje następujące funkcjonalności:

- przechowanie na serwerze danych i historii ich edycji
- edycja danych w czasie rzeczywistym (obraz):
    - rysowanie linii prostych/łamanych
    - nanoszenie adnotacji na obraz
    - wskaźniki interaktywne
- lista uczestników konferencji
- komunikacja pisemna

Aby obraz mógł zostać pobrany przez członków konferencji, najpierw musi znaleźć się na serwerze. Po pobraniu obrazu, gdy przejdziemy do etapu jego edycji tzn. rysowania linii, dodawania adnotacji itp. struktura obrazu pozostaje nienaruszona. Poszczególne obiekty (np. narysowana linia) są obiektami, które znajdują się na przezroczystych warstwach, znajdujących się nad obrazem.

3. Charakterystyka wybranych technologii i uzasadnienie ich wyboru

Do realizacji projektu użyliśmy technologii Java SE 8 + MySQL + Kryonet, przy pomocy których zapewniliśmy wszystkie funkcjonalności serwerowe. Po stronie klienta aplikacja została napisana przy wykorzystaniu JavaFX 8 oraz Kryonet. Kryonet jest biblioteką która zapewnia wydajną komunikację TCP/UDP oraz serializację obiektów.

4. Opis implementacji

W implementacji zastosowano podział na serwer współpracujący z bazą danych, którego zadaniem było przechowywanie danych, routing poleceń oraz zapewnienie synchronizacji konferencji. Zadaniem klienta jest pobranie danych z serwera, przetwarzanie ich, wyświetlanie, edycja i odsyłanie ponownie na serwer.

Baza danych

Actions_history

ID	GROUP_ID	FILE_ID	TYPE_ID	USER_ID	PARAMETERS
53	16	126	4	13	#163.7176470588235...#1#126#pl@gma...
54	16	126	4	13	#162.0470588235294#1.6705882352941177#1#126#pl@gma...
55	16	126	4	13	#437.6941176470589#20.04705882352941#1#126#pl@gmai...
56	16	126	4	13	#-693.2941176470588#28.400000000000002#1#126#pl@gm...
57	16	126	4	13	#377.55294117647054#686.6117647058824#1#126#pl@gma...
58	16	126	4	13	#359.17647058823536#-349.15294117647056#1#126#pl@g...
59	16	126	4	13	#-679.9294117647058#-60.14117647058824#1#126#pl@gm...
60	16	126	4	13	#-180.0432112804185#-248.66954741869458#1#126#pl@g...
61	16	126	4	13	#276.12008187400505#12.110529906754607#1#126#pl@gm...
62	16	126	4	13	#388.5#5.5#1#126#pl@gmail.com
63	16	126	2	13	#334.5#165.0#8388863#Tutaj wpisz tekst#3#126#pl@gm...

Sposób w jaki zostało zaimplementowane przez nas przechowywanie operacji pozwala nam na odtworzenie przebiegu konferencji krok po kroku. W tabeli zaprezentowanej powyżej, dla klienta najważniejsze są informacje zawarte w kolumnie *TYPE_ID* oraz *PARAMETERS*. Ta pierwsza oznacza rodzaj akcji jaki ma zostać wykonany. W powyższej tabeli 4 oznacza przesunięcie obiektu, 2 natomiast dodanie nowej adnotacji do konferencji. W ostatniej kolumnie przechowywane są parametry obiektu. W zależności od typu obiektu są to inne dane, tak np. dla linii będzie to pozycja punktów końcowych oraz kolor. Dla powyższego przykładu (adnotacji) będzie to natomiast pozycja na obrazie, kolor oraz tekst adnotacji. Dodatkowo każdy obiekt zawiera unikalny identyfikator, zapisany w parametrach, na który składa się lokalne id obiektu, id obrazu na którym obiekt został narysowany

oraz osoba, która dodała obiekt. Wszystko po to, aby uniknąć synchronizacji obiektów po stronie serwera, jako że nie przechowujemy obiektów jawnie w bazie danych. Po stronie klienta parsowany jest ciąg string i na tej podstawie odczytywane są szczegółowe informacje, jakie operacje mają zostać wykonane. Przechowywanie parametrów jako string, a nie jako części struktury tabeli wynika z faktu, że zestaw parametrów jest inny dla adnotacji, rysowania linii prostej, linii łamanej, przez co nie da się go jednoznacznie usystematyzować, pomimo wspólnych części różnych typów. Tworzenie nowej tabeli dla każdego obiektu (inna dla linii prostej, linii łamanej itp.) mocno utrudnia rozbudowę aplikacji w przyszłości. Rozwiązanie takie sprawia, że aplikacja jest bardziej elastyczna pod kątem implementacji nowych rozwiązań. Dodawanie nowych możliwości nie wiąże się z ingerencją w bazę danych.

files

id	path	name	user_id
114	.	ElonMusk.jpg	4
115	.	borabora.jpg	4
116	.	server11213big_image.jpg	4
117	.	server110evalm_board.png	4
118	root/	*NULL*	4

Istotnym aspektem implementacji jest przechowywanie struktury plików jako rekordów w bazie danych. W tabeli *files* przechowywane są linki do plików na dysku oraz struktura katalogów jaką widzi użytkownik. Wynika to z faktu, iż przeszukiwanie bazy danych, sortowanie rekordów, parsowanie stringów jest szybsze niż operowanie na fizycznych plikach. W celu rozróżnienia pliku i folderu zdecydowaliśmy się nie tworzyć osobnej tabeli lecz umieścić obie struktury w jednej. Jeśli pole *name* jest puste, to dany obiekt jest folderem o adresie widniejącym w kolumnie *path*. Dane zawarte w kolumnie *path* nie mają odzwierciedlenia w rzeczywistej strukturze plików na dysku. Na dysku, wszystkie pliki znajdują się w jednym folderze, a w momencie odebrania przez serwer mają nadawaną tymczasową unikatową nazwę. Która ponownie zostaje podmieniona na oryginalną w chwili pobierania ich z serwera.

groups

ID	NAME	OWNER_ID
16	stargate	4
17	atlantis	6

Tutaj implementacja jest stosunkowo prosta. Grupa reprezentuje konferencja jaka została kiedykolwiek założona. Przechowujemy tylko jej nazwę oraz id osoby, która ją założyła. *OWNER_ID* przechowujemy ze względu na to iż wprowadziliśmy podział na konferencje trwające oraz zamknięte. Jeśli lekarz będzie chciał się skontaktować z pacjentką, to tworzy nową konferencję, do której dołącza pacjentka. Po zakończeniu konferencji niepożądanym jest, aby jakieś osoby postronne miały dostęp do danych które zostały udostępnione w ramach konferencji więc zostaje ona zamknięta. Ponownie otworzyć może ją tylko osoba, która ją utworzyła.

group_members

ID	USER_ID	GROUP_ID
1	4	16
7	7	17

Prosta struktura mająca służyć do zapamiętania, kto w jakiej konferencji bierze w danej chwili udział.

users

ID	NAME	SURNAME	EMAIL	PASSWORD	POSITION
1	Piotr	Majewski	xyz@gmail.com	xyz	PATIENT
2	Patryk	Lewandowski	zyx@gmail.com	yzx	DOCTOR

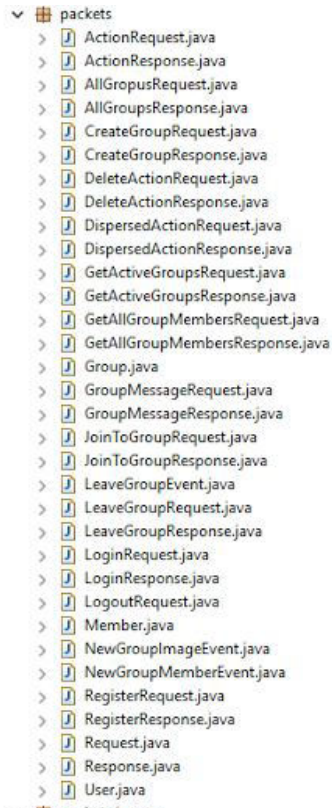
Przechowuje podstawowe informacje służące do identyfikacji użytkownika. Elementem unikatowym jest adres e-mail.

Serwer - ogólny zarys funkcjonowania

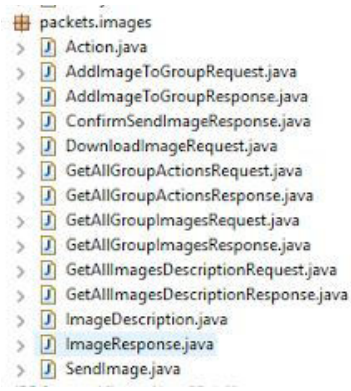
Serwer działa na dwóch osobnych portach. Jeden służy do wymiany krótkich informacji tj. danych logowania, akcji które zaszły, informacji o błędach. Natomiast słuchacz przypięty do drugiego portu odpowiada za wymianę multimediów. Wynika to głównie z dwóch powodów: po pierwsze informacje o akcjach są stosunkowo małe, więc potrzebują znacznie mniejszych paczek, lecz paczek takich jest znacznie więcej i są one znacznie częściej wysyłane. Multimedia są znacznie obszerniejsze, przez co potrzebują większych paczek, czas ich przesyłu jest znacznie dłuższy, lecz operacji związanych z pobieraniem lub wysyłaniem plików multimedialnych jest znacznie mniej. Można by obsługiwać wszystko na jednym porcie przy pomocy paczek jednakowej wielkości dla akcji oraz multimediów. Trzeba by wtedy obrazek podzielić na znacznie mniejsze fragmenty lecz byłoby to nieefektywne. Najważniejsze jest zapewnienie płynności obecnie trwających konferencji. W momencie gdy trwa kilka konferencji a ktoś chciałby wysłać obraz na serwer doprowadziłby do opóźnienia innych konferencji, ponieważ serwer byłby zajęty odbieraniem milionów malutkich paczek, które musiałby następnie połączyć w jeden obraz. Podział zastosowany przez nas eliminuje ten problem, ponieważ nawet gdy ktoś wysyła obraz na serwer, to ta operacja jest realizowana przez serwer w innym wątku na innym porcie i nie ma wpływu na obecnie trwające konferencje. Pozwala to na zastosowanie dużych paczek tylko dla multimediów, przez co obraz jest dzielony na kilkadziesiąt paczek, a nie kilka tysięcy. W ten sposób oszczędzamy czas związany z operacjami deserializacji, które są pewnym narzutem czasowym.

Serwer - rozróżnianie operacji

Serwer oraz klient posiadają zestaw takich samych obiektów służących do wymiany informacji. Gdy do serwera dociera jakaś informacja, jest ona w rzeczywistości serializowanym obiektem, przez co można z łatwością sprawdzić jakiego jest on typu i wykonać operacje dla określonego typu obiektu.



W pakiecie packets znajdują się obiekty które kierowane są do serwera na pierwszy port, o którym była mowa wcześniej, służącym do wymiany małych i częstych informacji.



W pakiecie packets.images znajdują się obiekty służące do przesyłu multimediów.

Serwer - przesyłanie multimediów

Mechanizm przesyłania multimediów jest taki sam dla serwer-klient jak i klient-serwer. Zanim w ogóle dojdzie do przesłania obrazu zostaje on podzielony na porcje o tym samym rozmiarze ~16000 bajtów. Następnie można przystąpić do przesyłania obrazu. Po nawiązaniu połączenia do serwera zostaje wysłana informacja o rozmiarze całego obrazu. Może sobie zarezerwować miejsce dla obrazu. Każda paczka zawierająca fragment o wielkości około ~16000 bajtów ma swój identyfikator, który został jej nadany podczas podziału obrazka, dzięki czemu można kontrolować jaka część obrazu została już przesłana. Aby uniknąć niespójności danych cała operacja realizowana jest przy wykorzystaniu protokołu TCP.

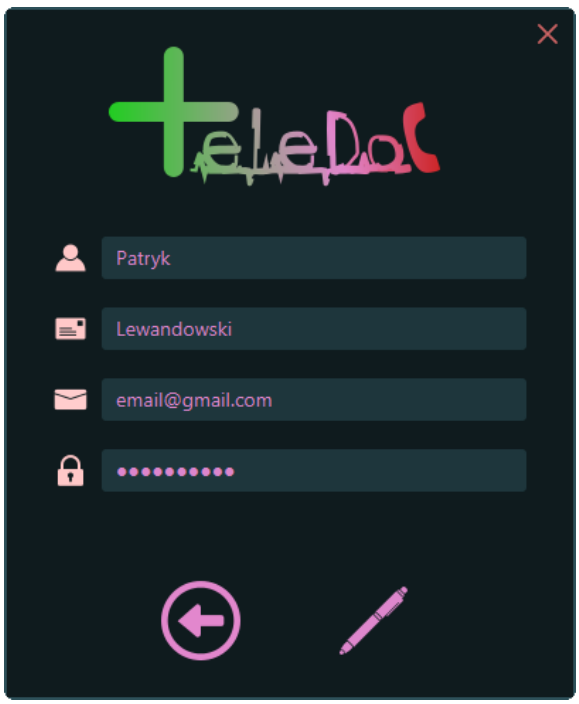
Serwer - zarządzanie akcjami

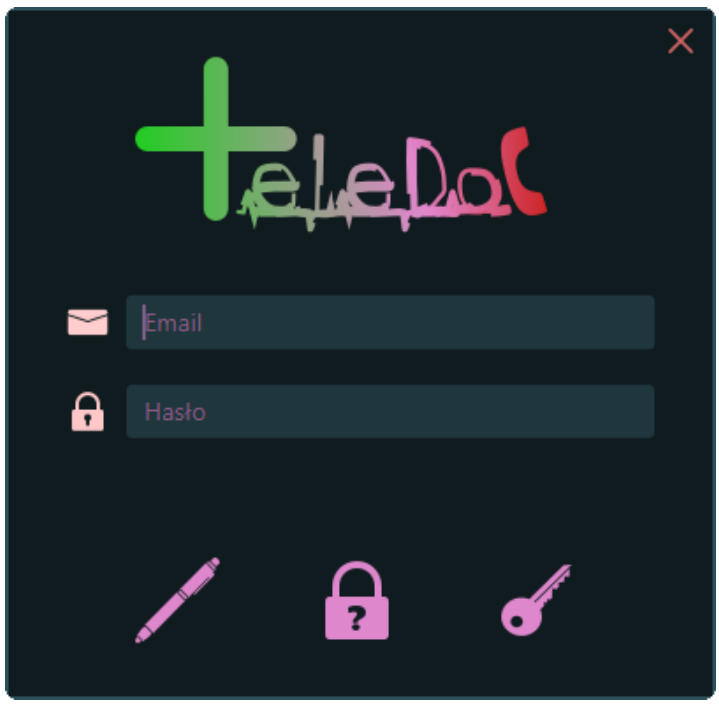
Akcje, czyli informacje o tym, jakie czynności zostały wykonane przez innych użytkowników podczas trwania konferencji, podzielone są na kilka kategorii.

- Akcje przechowywane w bazie - mają istotne znaczenie w przekazywaniu informacji pozostałym członkom konferencji. Zostają pobrane z serwera jako historia, gdy konferencja zostanie wznowiona w nowej sesji.
- Akcje rozproszone - akcje które nie muszą być przechowywane na serwerze, są dynamiczne i mają charakter pomocniczy w procesie wnioskowania. Należą do nich wskaźnik interaktywny oraz chat. Uznaliśmy że wszystkie wnioski powinny być przechowywane jako adnotacje na obrazie. Wynika to z tego, że jeśli chat byłby przechowywany w historii, to jeśli ktoś chciałby powrócić do treści konferencji i sprawdzić co wtedy ustalono, musiałby przeczytać całą treść konferencji a nawet jeśliby to zrobił, to nie ma pewności, że dobrze zinterpretowałby to, co się w niej znajduje. Uczestnicy mogli używać skrótów, niedopowiedzeń i lekarz mógłby myśleć że opisywana jest inna część obrazu niż w rzeczywistości. Dlatego lepiej jednoznacznie określić do czego konkretnie odnosi się dana treść poprzez dodanie adnotacji na obrazie. Są one też tak nazwane, ponieważ w obecnej formie, mogłyby być przesyłane całkowicie bez udziału serwera.
- Eventy - zdarzenia są to krótkie powiadomienia informujące że zaktualizowana została jakaś treść na serwerze np. dodano nowy obraz na serwer.

Klient - ogólny zarys

Aplikacja kliencka składa się z czterech różnych ekranów, które dotyczą innego aspektu logiki aplikacji. Pierwszym oknem jest okno logowania, które składa się z dwóch pól tekstowych do wprowadzenia odpowiednich danych oraz 3 przycisków służących odpowiednio do: przejścia do ekranu rejestracji, przypomnienia hasła oraz zalogowania do właściwej części programu. Okno rejestracji jest tak naprawdę tylko prostym formularzem służącym do zakładania konta, dlatego też znajdziemy tam pola tekstowe do uzupełnienia naszych danych oraz 2 przyciski: powrotu do ekranu logowania oraz wysłania na serwer danych do rejestracji.

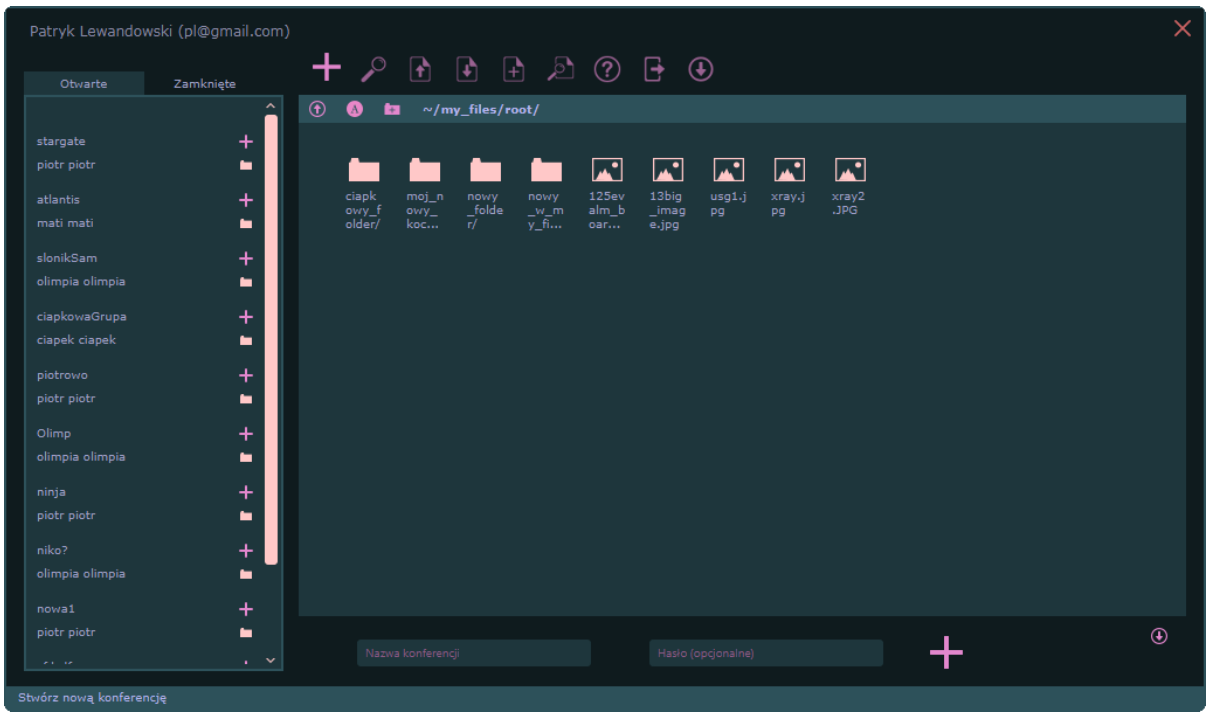




Okno rejestracji *Okno logowania*

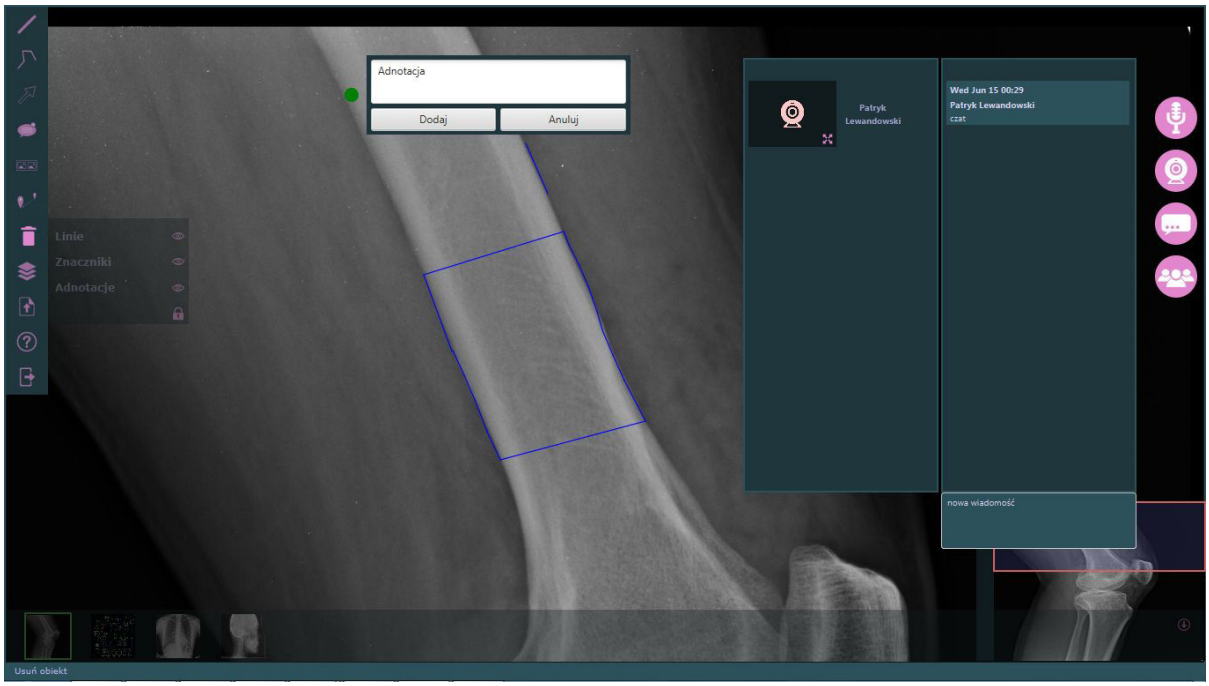
Po zalogowaniu przechodzimy do właściwego okna z aplikacją. W tym miejscu możemy sprawdzić, jakie konferencje aktualnie trwają oraz nad czym dana grupa osób pracuje. To jest też miejsce na wrzucenie własnych

plików na server, czy podgląd obrazów, które wrzucił ktoś inny. Aplikacja pozwala nam na przegląd wszystkich plików, a także tylko tych, które zostały przesłane przez nas. Możemy w bardzo prosty sposób dodać nową konferencję klikając ikonkę “plusika” nad panelem z plikami. Po stworzeniu konferencji, zostajemy automatycznie przeniesieni do kolejnego okna, gdzie odbywa się już wspólna praca nad danym obrazem. Do tego okna możemy też przejść, klikając mały plusik przy karcie z opisem konferencji, który spowoduje przyłączenie się do tej konferencji.



Okno po zalogowaniu - z lewej dostępne są konferencje. Prawą część zajmuje panel z plikami, a w dolnej części znajduje się panel, do utworzenia nowej konferencji oraz status bar opisujący dostępne akcje.

Kolejne okno pozwala nam na grupową pracę nad obrazami. Tutaj możemy komunikować się za pomocą czatu z innymi uczestnikami konsultacji. Możemy w wygodny sposób dodawać nowe pliki do konferencji z servera, a jeśli ich tam jeszcze nie ma, także najpierw je przesłać z naszego dysku. Użytkownik ma możliwość rysowania linii zwykłej i łamanej oraz dodawania adnotacji do obrazu. Jeśli użytkownik sobie życzy, może wyłączyć z widoku niektóre warstwy zawierające wcześniej dodane obiekty. Jest także możliwość podejrzenia aktualnie wczytanych obrazów do konferencji poprzez panel z obrazami. Tam także wyświetla się aktualnie otwarty obraz, a także zaznaczane są obrazy, w których inni użytkownicy dokonali jakichś zmian. Panel nawigacyjny pozwala na szybkie przemieszczanie się po obrazie, użytkownik ma dodatkowo możliwość przesuwania obrazu za pomocą myszki. Jest możliwość także podejrzenia aktualnej listy członków konsultacji, a wszystkie zmiany są sygnalizowane zmianą ikonki.



Okno “konferencyjne” - z lewej strony widać toolbar z akcjami jakie użytkownik może wykonać. Z prawej strony dostępny jest czat oraz lista uczestników. Prawy dolny róg zajmuje minimapa obrazu, która pozwala na nawigację. Dolną część wypełnia status bar, opisujący pewne akcje, a tuż nad nim znajduje się panel z wczytanymi do konferencji obrazami.

Klient - interfejs

Jak wcześniej wspomniano klient został napisany z wykorzystaniem technologii JavaFX 8, która to pozwala na tworzenie wydajnego interfejsu, który jest konfigurowalny praktycznie dla każdej możliwej kontrolki udostępnionej z tą technologią. FXML, wbudowany język oparty na XML pozwala na zbudowanie całego layoutu aplikacji oddzielnie od kodu związanego z logiką. Dodatkowo kontrolki są konfigurowalne za pomocą CSS, na którym się głównie opieraliśmy tworząc spójny interfejs. Wszystkie grafiki są wczytywane jako obrazy rastrowe, przez co przy zoom’owaniu ich mogą pojawiać się lekkie rozmycia wynikające z konieczności przeskalowania obrazu. Jakkolwiek JavaFX pozwala na wczytywanie i wyświetlanie grafik wektorowych, a biorąc pod uwagę, że wszystkie użyte ikonki były rysowane ręcznie w programie Inkscape, prawdopodobnie nie byłoby więc wielkiego problemu aby podmienić te grafiki. Całość była zaprojektowana tak, aby przede wszystkim zachować spójność aplikacji. W tym celu zastosowaliśmy coś co można nazwać “flat ui” czy “flat design”. Pojęcia te odnoszą się do tworzenia minimalistycznego interfejsu użytkownika, bez zbędnych “wodotrysków” dzięki czemu wszystko wygląda bardzo przejrzyście i można by powiedzieć nowocześnie. Tego typu interfejsy są przede wszystkim popularne na mobilnych platformach, czy różnych stronach internetowych, chociaż nawet system Windows od wersji 8 polega na tego typu rozwiązaniu. Pewnym kluczowym elementem był też wybór palety kolorów. Ważnym było aby wszystkie kolory ze sobą w pewien sposób współgrały. Do pomocy w wyborze palety wykorzystaliśmy serwis www.materialpalette.com. W oknie konferencji, głównym celem było udostępnienie jak największej powierzchni ekranu na sam obraz. Dlatego też staraliśmy się, aby jedynymi elementami go były przyciski, które wysuwają i chowają pozostałe panele, tak jak na przykład dla panelu z czatem. Nie jest potrzebne aby był on cały czas aktywny, więc użytkownik samemu może go wysunąć lub schować. Jest to bardzo wygodne tym bardziej, że ikonka czatu poprzez zmianę wyglądu powiadamia nas o nowej wiadomości. Dodatkowo część paneli staje się półprzezroczysta, kiedy nie są używane, a jest potrzeba aby były widoczne. Aplikacja posiada kilka skrótów klawiszowych wspomagających pracę, jednak w obecnej postaci nie są one wyświetlane w żaden sposób użytkownikowi. Powstały one ze względu iż początkowo był zamysł, aby umożliwić pracę także za pomocą samej klawiatury. Być może nie byłby to najwygodniejszy sposób na użytkowanie programu, jednak gdybyśmy pracowali na komputerze stacjonarnym, a nasza myszka akurat by nas zawiodła, bylibyśmy w stanie

kontynuować pracę. Poza tym wiele akcji, przy pewnej wprawie, wykonuje się szybciej za pomocą klawiatury niż myszki.

Klient - system plików

System plików użyty w aplikacji jest w całkowity sposób tylko reprezentacją logiczną. Przy logowaniu do aplikacji, klient wysyła żądania do servera, o wszystkie dostępne u niego pliki. Tutaj należy wspomnieć, że wszystkie pliki trzymane po stronie servera trzymane są fizycznie w jednym folderze i struktura fizyczna nijak ma się do tego, co możemy zobaczyć w aplikacji. Klientowi udostępniany jest wirtualny folder główny o nazwie “root”. W tym miejscu użytkownik może stworzyć dowolnie wyglądające drzewo katalogów i przesłać do nich pliki tam gdzie ich potrzebuje. Kiedy tworzony jest nowy folder, server jedyne co robi, to dodaje nowy wpis w bazie. Nie tworzy go fizycznie na dysku. Kiedy przesyłany jest plik, server dodaje ten plik do wspólnego folderu, gdzie trzymane są wszystkie pliki, a wpis w bazie mapuje odpowiednią ścieżkę wirtualną jaką widzi klient, na fizyczną ścieżkę pliku na dysku. Rozróżnianie czy wpis w bazie dotyczy folderu czy pliku jest bardzo proste ze względu iż te pierwsze posiadają w bazie tylko ścieżkę, a pliki posiadają ścieżkę oraz nazwę i na tej też podstawie dodawane są odpowiednie ikony w naszym eksploratorze plików.

5. Scenariusze użycia

1. Lekarz loguje się
2. Lekarz wybiera obraz do wysłania na serwer
3. Wysyła wybrany obraz na serwer
4. Lekarz tworzy nową konferencję
5. Dodaje obraz z serwera do nowo stworzonej konferencji

1. Lekarz loguje się do aplikacji
2. Lekarz wybiera konferencję której jest właścicielem i ją otwiera
3. Podczas otwierania aplikacja pobiera wszystkie obrazy z serwera
4. Pacjent loguje się do aplikacji
5. Wybiera konferencje którą podał mu jego lekarz
6. Potrzebne multimedia zostają pobrane przez klienta
7. Lekarz wyjaśnia pacjentowi diagnozę posługując się czatem, adnotacjami wskaźnikami…
8. Cały przebieg jest na bieżąco zapisywany na serwerze
9. Lekarz i pacjent wylogowują się i kończą konferencję

1. Pacjent chce powrócić do ustaleń z lekarzem
2. Pacjent loguje się do aplikacji
3. Wybiera konferencje w której wcześniej uczestniczył
4. Dołącza do niej
5. Zostają pobrane wszystkie obrazy które zostały wykorzystane we wcześniejszej sesji oraz historii przebiegu konferencji
6. Pacjent może odtworzyć cały przebieg konferencji
7. Pacjent wylogowuje się z aplikacji

6. Uzasadnienie zmian względem oryginalnego projektu

Początkowo nie przewidzieliśmy konieczności pobrania zdarzeń (krótka informacja o zmianie stanu). Założyliśmy że jeśli podczas trwania konferencji ktoś doda do niej obraz to serwer zainicjuje operację wysłania tego obrazu do pozostałych członków. Taka sytuacja zrzuca na serwer odpowiedzialność za rozpowszechnienie obrazu. Uznaliśmy że odpowiedzialność za spójność danych powinna spoczywać na aplikacji klienckiej. To on powinien inicjować pobranie obrazu, a nie serwer jego wysłania. Do klienta wysyłamy tylko informacje, że do konferencji

został dodany nowy obraz. Wynika to między innymi z tego, że w przyszłości w celu lepszej optymalizacji obrazy będą mogły być przechowywane tymczasowo po stronie klienta. W takim przypadku jeśli ten obraz już znajduje się u klienta ponieważ używał go już w innej konferencji, to nie ma konieczności ponownego jego pobierania.

Kolejnym powodem dodania zdarzeń było informowanie o dołączeniu nowych członków do konferencji. Jeśli w konferencji bierze udział kilka osób i w trakcie jej trwania ktoś chce się do niej dołączyć, to trzeba poinformować resztę o tym fakcie. Można by założyć, że klient ma co kilka sekund wysyłać do serwera zapytanie, czy nikt się nie dołączył do konferencji i czy nie ma czasem zaktualizować lokalnej listy uczestników konferencji. O ile w początkowym stadium konferencji, gdy wszyscy do niej dołączają mogłoby się to sprawdzić to przez pozostały czas trwania konferencji byłyby generowane niepotrzebne zapytania.

7. Opis możliwości rozszerzeń i dalszego rozwoju systemu

Rozbudowa tego projektu jest stosunkowa prosta, ponieważ na dłuższą metę wiąże się ona tylko z rozszerzaniem funkcjonalności klienta. Serwer zarządza tylko plikami, nawet nie muszą być to obrazy - mogą to filmy a nawet pliki exe. Docelowo aplikacja miałaby mieć charakter modułowy. Podstawową funkcjonalnością byłoby dokonywanie prostych operacji na obrazach. A w razie konieczności można by pobrać pakiety obsługujące dodatkowe funkcjonalności takie jak operacje na plikach dicom. Nie każdy potrzebuje obsługi dicom więc nie trzeba oferować mu tej funkcjonalności w wersji podstawowej. Dodatkowe moduły byłyby udostępniane na zasadzie możliwości pobrania wtyczek, które tak naprawdę byłyby dodatkowymi plikami jar. odpowiadające za dane funkcjonalności. Wszystkie pluginy byłyby przechowywane na serwerze i udostępniane na zasadzie podobnej do eclipse marketplace. Dodatkowym rozszerzeniem, mogłaby być możliwość użytkowania aplikacji na systemach Android. O ile w przypadku smartfonów całkowicie mijałoby się to z celem, o tyle na większych ekranach, np. tabletów, mogłoby mieć to już sens. Byłoby to bardzo wygodne dla użytkowników, ponieważ mogliby korzystać z aplikacji praktycznie w każdym miejscu.