

# Projekt zespołowy elektryczny

## Czat internetowy OKNO

Tematem niniejszego opracowania jest zaprojektowanie i implementacja aplikacji webowej do komunikacji tekstowej z wykorzystaniem nowoczesnych technologii takich jak: HTML, CSS, JavaScript, JQuery, PHP, MySql. Dokument stanowi uszczegółowienie tematu projektu zespołowego poprzez opisanie najważniejszych elementów składających się na projekt, jak również pokazanie potencjalnych możliwości rozbudowy już zaimplementowanego i działającego produktu w postaci aplikacji webowej czat okno.

Skład zespołu projektowego:

Artur Borkowski  
Mariusz Bielawski  
Rafał Litwińczuk

## Table of Contents

Czat internetowy OKNO .....	1
<b>1. Założenia wstępne .....</b>	<b>3</b>
1.1 Baza danych. ....	3
1.2 Język po stronie serwera.....	4
1.3 Interfejs użytkownika .....	4
1.4 Dynamiczny interfejs, czyli JavaScript i JQuery .....	4
1.5 Asynchroniczne połączenia, czyli Ajax .....	5
<b>2. Baza danych MySql.....</b>	<b>5</b>
<b>3. PHP PDO .....</b>	<b>8</b>
<b>4. Asynchroniczny Ajax.....</b>	<b>9</b>
<b>6. Budowa aplikacji czat okno .....</b>	<b>10</b>
<b>7. Chatbot.....</b>	<b>13</b>
<b>8. Przypadki użycia.....</b>	<b>16</b>
8.2 Logowanie .....	18
8.3 Dodawanie wiadomości na czacie .....	18
8.4 Formatowanie tekstu .....	20
8.5 Hiperłącze.....	20
8.6 Zmiana opisu i dodanie awatara .....	20
8.7 Wysyłanie maila .....	20
8.8 Zmiana obszaru rozmowy (zmiana pokoju) .....	20
8.9 Czat prywatny .....	22
<b>9. Screeny z działającej instancji aplikacji czat okno uruchomionej na hostingu <a href="http://www.czatokno.warszawa.pl">www.czatokno.warszawa.pl</a> .....</b>	<b>23</b>
9.1 Panel admina 1 .....	23
9.2 Panel admina 2.....	23
9.3 Widok ogólny.....	24
9.4 Screen paska do wpisywania wiadomości .....	24
9.5 Informacje o zalogowanym użytkowniku i link do prywatnego czata.....	24
9.6 Rozwijana lista zarejestrowanych uczestników z opcją wysłania wiadomości email.....	25
<b>10. Podsumowanie .....</b>	<b>25</b>
Bibliografia.....	26

## 1. Założenia wstępne

Usługa czata internetowego istnieje tak długo jak długo istnieje globalna sieć komputerów znana powszechnie jako Internet. Możliwość komunikowania się między komputerami i przesyłania informacji zapoczątkowała nową erę we współczesnej historii komputerów. Celem naszym nie było wynajdowanie koła od nowa, ale zaprojektowanie i stworzenie funkcjonalnej i przyjaznej dla użytkownika aplikacji internetowej, umożliwiającej wysyłanie wiadomości tekstowych, albo ujmując wszystko bardziej dosłownie, rozmowę różnej ilości użytkowników. Inspiracją do zaprogramowania takiej aplikacji nie była tylko sama potrzeba komunikacji, ale również temat pracy inżynierskich dla studentów OKNA. Jednym z takich tematów był właśnie temat własnej implementacji czata internetowego z wykorzystaniem technologii PHP+MySQL+Apache. Wzbogaceni przez lata studiów o wiedzę na temat programowania internetowego postanowiliśmy podjąć się realizacji tematu wzbogacając go o kilka innych technologii zgodnych z duchem czasów w których przyszło nam mieć ten przywilej korzystania z dobrodziejstw technologii informatycznych. Po wstępnym rozpoznaniu możliwości jakimi dysponujemy, uznaliśmy, że można stworzyć aplikację czata z wykorzystaniem tych technologii, ale można też dodatkowo zastosować mechanizmy działające po stronie przeglądarki, która będzie najważniejszym medium komunikacyjnym w aplikacji. Od razu pomyśleliśmy o schemacie klient-serwer. Realizacją zadań po stronie serwera miał zająć się serwer www Apache, baza danych MySQL i skryptowy język PHP. Po stronie klienta mieliśmy zamiar zaimplementować stronę internetową stanowiącą interface użytkownika. Postanowiliśmy do tego użyć HTML, CSS, JQuery i Ajax-a, o których napiszemy w następnych punktach tego opracowania. Na wstępie wiedzieliśmy już, że będzie to aplikacja klient-serwer. Pozostało jeszcze opracować szczegółowe rozwiązania dotyczące połączeń klientów z serwerem, zaprojektować wygląd interfejsu użytkownika i baze danych.

### 1.1 Baza danych.

Na magazyn dla wszystkich danych gromadzonych po stronie serwera wybraliśmy bazę MySQL. Od samego początku projektowania aplikacji wiedzieliśmy, że będzie to MySQL. Jest to wieloplatformowa baza danych wykorzystywana w tysiącach aplikacji na całym świecie. Podobnie jak Apache, który jest najpopularniejszym serwerem www, baza MySQL jest dobrze udokumentowana. Nasza baza miała za zadanie w sposób bezpieczny przechowywać wszystkie wprowadzone przez użytkowników dane i umożliwiać szybkie i wydajne połączenia. MySQL całkowicie zapokaja te

wymagania. Razem z bazą często używany jest system do zarządzania bazą o nazwie phpMyAdmin uruchamiany w przeglądarce.

### 1.2 Język po stronie serwera

Baza danych to nie wszystko. Chcieliśmy, żeby wszelkie zapytania formułowane do bazy w języku SQL, były realizowane za pośrednictwem wydajnego języka.

Językiem tym jest PHP, który również od lat doskonale sprawdza się w tego typu zastosowaniach. Obiektowy język PHP z interfejsem PDO. Skrót PDO oznacza PHP Data Objects. Jest to zupełnie nowy interfejs języka PHP przeznaczony do komunikacji z bazami danych, po raz pierwszy napisany wyłącznie w OOP. Jego najważniejszą zaletą jest to, że możemy za jego pomocą łączyć się bezpiecznie z bazą danych MySQL dzięki metodzie `prepare()` i bindowaniu danych.

### 1.3 Interfejs użytkownika

Zdawaliśmy sobie sprawę, że powinniśmy możliwie jak najlepiej zadbać o efektowny i przyjazny dla użytkownika wygląd. Wiedzieliśmy, że wykorzystamy do tego celu CSS, czyli kaskadowe arkusze stylów. Projektowanie stylu dla poszczególnych elementów aplikacji webowej to zajęcie dające szerokie pole dla kreatywności. Ostatecznie jednak zdecydowaliśmy się na wykorzystanie bardzo popularnego frameworka CSS, a mianowicie Bootstrapa. Framework ten w żaden sposób nas nie ograniczał, a wręcz przeciwnie. Dzięki niemu mogliśmy korzystać już na początku z wielu rozwiązań, które są szeroko stosowane w nowoczesnych aplikacjach webowych. Nasz wybór dał nam solidny fundament do tworzenia i późniejszego modyfikowania wielu ważnych elementów strony. Nie ograniczał nas również w stosowaniu własnego stylu CSS, ponieważ Bootstrap w pełni umożliwia dodawanie własnego stylu bez kolizji z samym frameworkiem. CSS to kaskada stylów w której wykorzystywana jest zasada dziedziczenia, a więc o ile framework świetnie ustala pozycje danego elementu względem całości, to nasza inwencja pozwala nam również na wykorzystanie własnych rozwiązań. Struktura aplikacji pisanej w oparciu o framework Bootstrap przypomina siatkę rzędów i kolumn w której umieszczamy inne elementy strony i nadajemy im własny styl.

### 1.4 Dynamiczny interfejs, czyli JavaScript i JQuery

Chcieliśmy również, żeby aplikacja nie była statycznym dokumentem html, ale żeby umożliwiała różne interakcje z użytkownikiem takie jak dodawanie url do wpisywanego tekstu, dodawanie emotów, zmianę stylu wpisywanego tekstu i inne efekty. Rozsuwany panel menu w panelu użytkownika i kilka innych dodatków poprawiających jakość korzystania z interfejsu. Wykorzystaliśmy do tego celu

JavaScript i bibliotekę JQuery. Szczegółowy opis zastosowań w dalszej części opracowania.

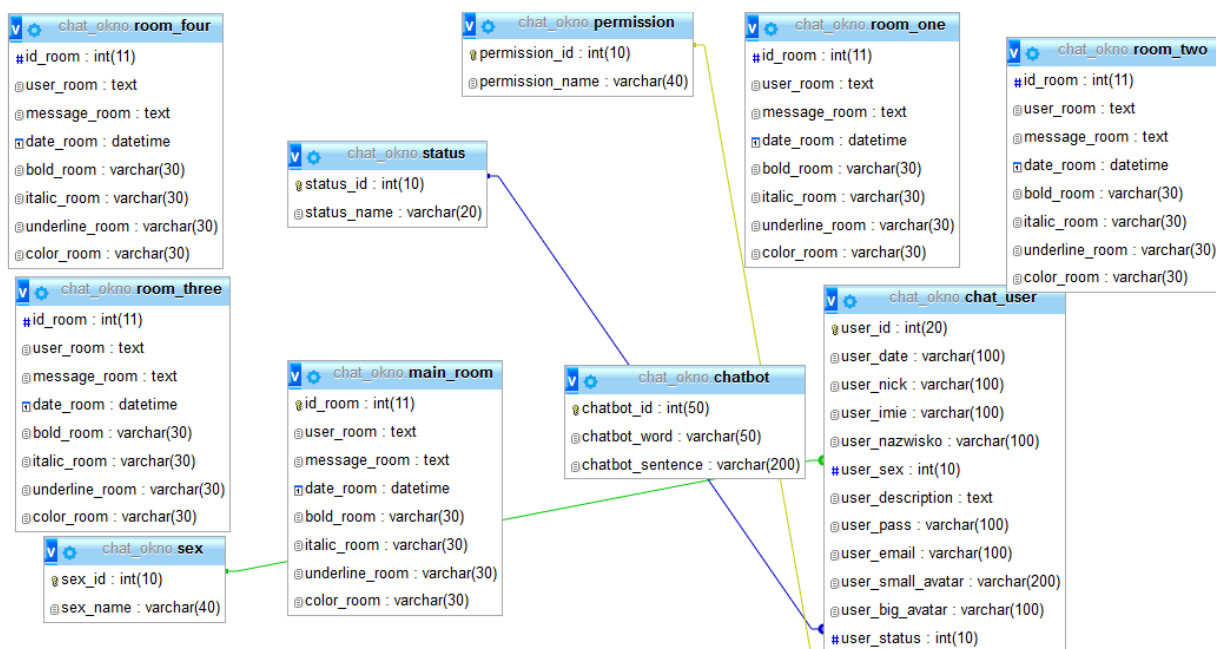
#### 1.5 Asynchroniczne połączenia, czyli Ajax

Wiedzieliśmy już, że nasza aplikacja będzie wysyłać zapytania do serwera. Gdybyśmy pozostali przy rozwiązaniu z PHP po stronie serwera, to każde uaktualnienie stanu rozmowy po stronie klienta wymagałoby odświeżenia całego dokumentu w przeglądarce. Jest to bardzo uciążliwe i w dalszej perspektywie przeszkadza w płynnej wymianie informacji. Potrzebowaliśmy rozwiązania, które pozwoli bez zakłóceń i męczącego “restartowania” aplikacji wyświetlać aktualny stan rozmów na czacie. Nasz wybór padł na technologie Ajax. Zastanawialiśmy się przez chwilę nad node.js, które również nadaje się do tego typu zastosowań, ale czuliśmy się pewniej z naszym wcześniejszym wyborem. Ostatecznie pozostaliśmy przy wyborze Ajaxa, który umożliwia asynchroniczną komunikację na linii przeglądarka-serwer, bez konieczności uciążliwego przeładowywania strony.

## 2. Baza danych MySQL

Relacyjna baza danych MySQL jest najczęściej stosowaną bazą w aplikacjach bazodanowych. Internetowa aplikacja bazodanowa jaką zaprojektowaliśmy opiera się na klasycznej architekturze internetowej bazy danych. Przeglądarka użytkownika wysyła żądanie udostępnienia określonej strony www. Serwer przyjmuje żądanie, następnie odnajduje plik php i przekazuje go do interpretera PHP. Interpreter rozpoczyna przetwarzanie skryptu. Wewnątrz skryptu zawarte jest polecenie połączenia z bazą danych i wykonanie zapytania SQL. Następuje otwarcie połączenia z serwerem MySQL i przesłanie zapytania. Serwer MySQL przyjmuje zapytanie i je przetwarza, po czym rezultat odsyła do interpretera PHP. Interpreter kończy wykonywanie skryptu, który formatuje otrzymane wyniki zgodnie ze standardami HTML i przesyła wynikowy kod HTML do serwera www. Serwer www przesyła kod HTML do przeglądarki.

Stworzyliśmy bazę danych chat\_okno, w której umieściliśmy kilka tabel. Bardzo pomocny okazał się program do zarządzania bazami danych phpMyAdmin. Wykreowaliśmy użytkownika aplikacji i nadaliśmy mu ograniczone uprawnienia. Struktura bazy przedstawimy na diagramie:



Na diagramie widać wszystkie tabele i niektóre relacje między nimi. Koncepcja aplikacji przewiduje istnienie tabeli do zapisywania danych na temat użytkownika. Tabele przechowujące wiadomości wpisane podczas czatowania. Są cztery tabele odzwierciedlające 4 lata studiów i jedna tabela do ogólnych rozmów na czacie `main_room`. Dodatkowo w bazie tworzą się tabele tymczasowe w przypadku kiedy dwaj użytkownicy zechcą skorzystać z opcji czata prywatnego. W tabeli `chat_user` przechowywane są informacje o użytkownikach. Podczas rejestracji użytkownik podaje swój nick, którym chce się posługiwać, imię, nazwisko, płeć, adres email oraz hasło. Po rejestracji użytkownik może dodatkowo ustawić swój opis i uploadować awatara, który będzie widoczny w panelu zalogowanych użytkowników. Tabele przechowujące wiadomości pisane przez użytkowników składa się z kilku pól, gdzie najważniejszym polem jest `message`. Pozostałe pola to data dodania wiadomości, nick użytkownika, który dodał wiadomość oraz kilka pól formatujących tekst, czyli pogrubienie, pochylenie, podkreślenie, kolor tekstu. Zgodnie z zasadą, która mówi o unikaniu powielania danych w tabelach, takie informacje o użytkowniku jak płeć, status i uprawnienia przechowujemy w innych tabelach połączonych relacjami z tabelą `chat_user`. Za połączenia z bazą i wysyłanie zapytań odpowiedzialny jest skrypt php, a dokładniej klasa `dbConnection`. Klasa ta jest klasą po której dziedziczą inne klasy wysyłające zapytania do bazy MySQL. Na rysunku przedstawiamy klasę `dbConnection()`.

```

class dbConnection
{
    protected $db_conn;

    public function __construct()
    {
        try
        {
            require_once('db.php');

            $this->db_conn = new PDO('mysql:host='.$host.';dbname='.$dbname, $usr, $pass);
            $this->db_conn->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );

            .....
        }
        catch(DBException $e)
        {
            echo 'The connection can not be created: ' . $e->getMessage();
        }
    }

    public function __destruct()
    {
        .....
        $this->db_conn = null;
    }
}

```

Wybrane zapytania SQL, realizowane do bazy:

Zapytanie wyciągające 44 ostatnie wpisy w kolejności od najnowszego w taki sposób, aby świeżo dodane wpisy na czacie wyświetlały się u dołu ekranu.

1. *\$query = "SELECT id\_room, user\_room, message\_room, date\_room, bold\_room, italic\_room, underline\_room, color\_room FROM (SELECT id\_room, user\_room, message\_room, date\_room, bold\_room, italic\_room, underline\_room, color\_room FROM \$target ORDER BY id\_room DESC LIMIT 44 ) tmp ORDER BY tmp.id\_room";*

Zapytanie dodające do bazy nową wiadomość

2. *\$sql = "INSERT INTO \$target (user\_room, message\_room, date\_room, bold\_room, italic\_room, underline\_room, color\_room) VALUES ( :user\_room, :message\_room, :date\_room, :bold\_room, :italic\_room, :underline\_room, :color\_room )";*

Zapytanie pobierające z bazy wszystkie informacje o użytkownikach.

```
3 $q = "SELECT chat_user.user_id, chat_user.user_date, chat_user.user_nick,
chat_user.user_imie, chat_user.user_nazwisko, sex.sex_name,
chat_user.user_description, chat_user.user_email, status.status_name,
permission.permission_name FROM chat_user
```

```
LEFT JOIN sex
```

```
ON chat_user.user_sex = sex_id
```

```
LEFT JOIN status
```

```
ON chat_user.user_status = status_id
```

```
LEFT JOIN permission
```

```
ON chat_user.user_permission = permission_id";
```

### 3.PHP PDO

PHP jest skryptowym językiem programowania wykonywanym po stronie serwera. W przeglądarce internetowej po wysłaniu żądania mamy do czynienia już z przetworzonym zapytaniem na serwerze, bo właśnie na serwerze działa PHP. Składnia podobna do składni innych języków wysokiego poziomu powoduje, że PHP to język łatwy z bogatą dokumentacją ogólnie dostępną w manualu do tego języka. W PHP z powodzeniem pisze się wszystkie rodzaje aplikacji bazodanowych, ponieważ jest to język niezwykle elastyczny i ciągle rozwijany. Na znacznej większości hostingów w standardzie jest PHP i baza MySQL. Do licznych zalet PHP można zaliczyć szybkość i stabilność. PDO natomiast, to obiektowa biblioteka języka PHP służąca do obsługi bazy danych. PDO jest równie łatwe w implementacji jak inne biblioteki. Dużą zaletą tej biblioteki jest fakt, że baza MySQL jest chroniona przed niebezpiecznymi atakami typu Sql Injection. Wszystko dzięki metodzie `prepare()`, którą przygotowuje się szkielet zapytania do bazy zanim jeszcze umieścimy w niej nowy rekord. Drugim ważnym powodem decydującym o bezpieczeństwie zapytań do bazy jest binding, czyli podpinanie. Podpinanie polega na przeniesieniu spajania danych z zapytaniem z języka programowania na serwer DB. Do bazy wysyłamy tutaj tak naprawdę szkielet zapytania ze specjalnymi wstawkami, do których później podpinamy interesujące nas dane za pomocą



specjalnej metody, gdzie możemy dodatkowo określić ich typ (tekst, liczba itd.). Podpinanie jest odporne na ataki SQL Injection. MySQL ma jasno określone, co jest danymi, a co zapytaniem i ściśle się tego trzyma. Ponadto jest także wydajniejsze, niż samodzielne spinanie wszystkiego po stronie PHP. Z tych właśnie wyżej wymienionych powodów zdecydowaliśmy się na PDO.











#### 4. Asynchroniczny Ajax

Ajax to asynchroniczny JavaScript i XML, tymczasem w aplikacji czata wykorzystujemy JavaScript i Json. W niektórych przypadkach odpowiedź z serwera nadchodzi w formie zwykłego tekstu. Dzięki asynchroniczności wszystkie interakcje użytkownika z serwerem zachodzą bez potrzeby przeładowania całego dokumentu. Można obrazowo napisać, że w przypadku czata, serwer jest bombardowany dużą ilością żądań, a użytkownik w ogóle tego nie zauważa, ponieważ aplikacja pozostaje ciągle w tym samym stanie i nie przeładowuje się. We wszystkich wywołaniach ajaxa wykorzystujemy właściwości obiektu XMLHttpRequest. Technologia Ajax rozwiązuje problem zrównoważenia komunikacji między klientem a serwerem, przenosząc ten proces w tło innych działań użytkownika, który może w tym czasie pracować czynnie na danej stronie www. W aplikacji czata nie wykorzystujemy bezpośrednio obiektu XMLHttpRequest, ale robimy to poprzez bibliotekę JQuery w której metodach znajduje się funkcja \$.ajax(). W naszej aplikacji wykorzystujemy tą funkcję kiedy użytkownik dodaje nową wiadomość. Metoda JQuery click() uruchamia funkcję ajax, która zawiera w swoich parametrach sposób przesyłania zmiennych, adres url skryptu, dane do przesłania, typ danych zwracanych przez serwer. Jak już wspomniałem wcześniej, w większości przypadków korzystamy z formatu JSON. Aktualizowanie obszaru rozmów również odbywa się przy wsparciu funkcji ajax. Serwer co pół sekundy odpowiada na żądanie. Użytkownik nie dostrzega tych procesów działających w tle i właśnie dlatego uznaliśmy, że ajax() można zaimplementować w aplikacji czata. Udało nam się uzyskać zamierzony efekt. Dodatkowo na stronie interfejsu użytkownika działają jeszcze inne funkcje z ustawionym interwałem czasowym, które aktualizują stan i zawartość takich obszarów strony jak panel zalogowanych użytkowników, panel pod menu głównym w którym pokazują się linki do prywatnego czata, jeśli któryś z użytkowników wybierze opcje prywatnej rozmowy z innym użytkownikiem. Na screenie fragment funkcji \$.ajax()

```
$.ajax({
    type: 'POST',
    url: "http://localhost/czatokno/php/chat.php",
    cache: false,
    data: $.param({
        'mode': mode,
        'target': target
    }),
    dataType: 'json',
    error: function(xhr, textStatus, errorThrown) {
        //displayError(textStatus);
    },
    success: function(data, textStatus)
    {
        if(data.errno != null)
            displayPHPError(data);
        else
            readMessages(data);
    }
});
```

## 6. Budowa aplikacji czat okno



























Omawianie budowy aplikacji zaczniemy od przedstawienia struktury katalogów.

	Name
	admin
	avatars
	css
	emoticon
	font
	html
	img
	js
	php
	index.php














W katalogu avatars znajdują się katalogi z avatarami zarejestrowanych użytkowników. W katalogu css są umieszczone wszystkie style. Pliki php i js są umieszczone odpowiednio w katalogach.

Plikiem startowym jest index.php, który składa się z dwóch plików inkludowanych w momencie startu aplikacji i zawartości pomiędzy nimi. Są to pliki: header.html.php i footer.html.php, które znajdują się w katalogu html. Wszystkie pliki html posiadają rozszerzenie .php, a sam kod PHP jest w nich zagnieżdżony.

Zawartość katalogu php:

	avatar.php
	chat.class.php
	chat.php
	chatbot.php
	db.php
	dbconnection.class.php
	email_validate.php
	error_handler.php
	logged_user.php
	logging.php
	logout.php
	nick_validate.php
	priv.class.php
	priv.php
	register_top.php
	reset_form.php
	scal.class.php
	send_email.php
	tables.php
	upload_avatar.php
	user.class.php
	user.php
	user_panel.php
	users_list.php
	validate.class.php
	validate.php

Zawartość katalogu js:

	bootstrap.js
	bootstrap.min.js
	chat.js
	email_validate.js
	inicjator.js
	jquery-1.11.3.min.js
	logout.js
	main.js
	nick_validate.js
	npm.js
	register.js
	user.js
	user_menu.js

Najważniejsze skrypty .php działają poprzez duet dwóch plików z których jeden jest kontrolerem, a drugi zawiera klasę, której instancja obiektu powstała przy wywołaniu kontrolera.

Np. chat.php->chat.class.php

user.php->user.class.php

validate.php->validate.class.php

priv.php->priv.class.php

Struktura aplikacji jest dosyć prosta i dlatego początkowo zastanawialiśmy się nad umieszczeniem wszystkich metod w jednej klasie, ale z drugiej strony jest na tyle skomplikowana, że potrzebuje kilku klas do prawidłowego działania zgodnie ze swoim przeznaczeniem. Dodam jeszcze, że wszystkie klasy dziedziczą po klasie dbConnection odpowiedzialnej za połączenie z bazą MySQL. Przedstawiliśmy najważniejsze pliki. Możemy dodać jeszcze screen zawartości katalogu html i diagram klas .

Zawartość katalogu html:




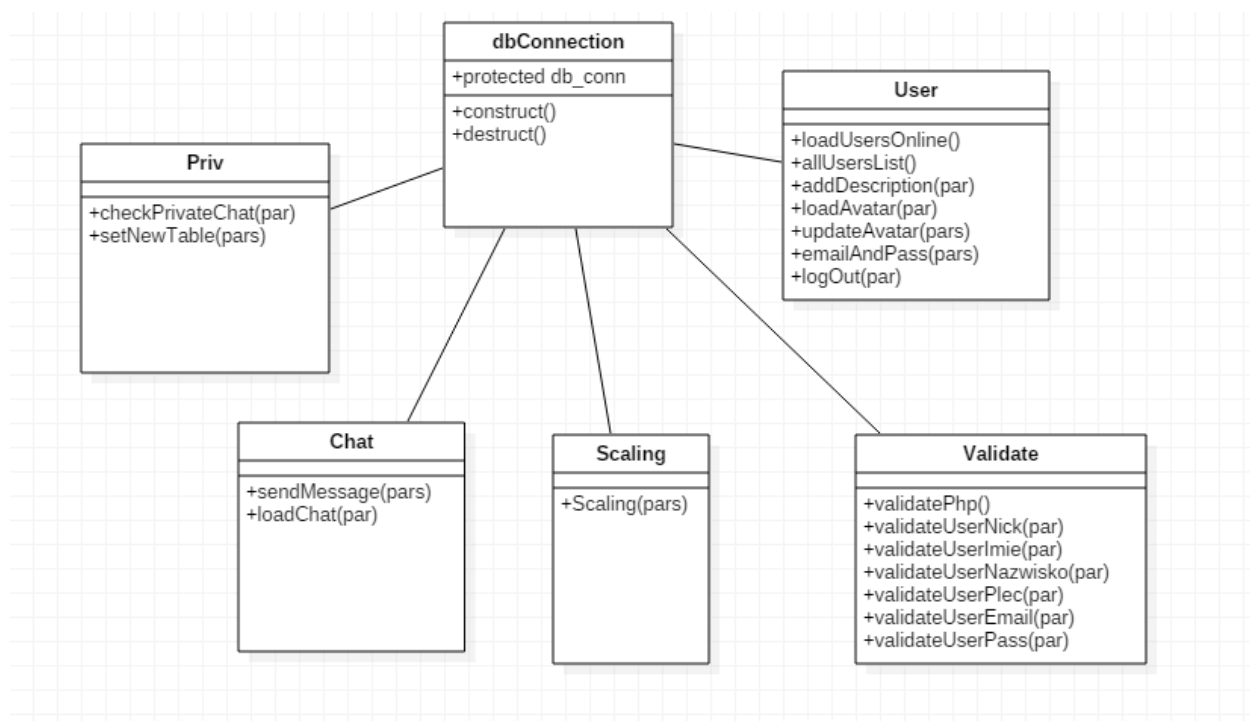
	footer.html.php
	header.html.php
	login form.html.php

Diagram klas:



## 7. Chatbot

### 7.1 Implementacja dynamiczna

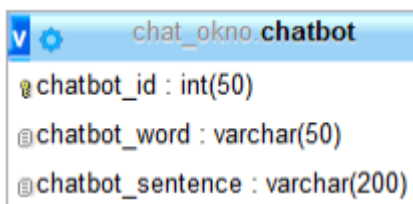
Chatbot to program, który imituje działanie realnego użytkownika. Implementacja takiego programu polega na zaprojektowaniu tabeli w bazie MySql i wypełnieniu jej "wiedzą" chatbota. Jest to tabela z polem słowo i drugim polem, które zawiera sentencje przyporządkowaną do tego słowa. Tabela ta stanowi zbiór rekordów słowo>sentencja. Działanie chatbota inicjowane jest skryptem inicjator.js po stronie klienta. W pliku inicjator.js jest funkcja `chatBot()`, która realizuje zapytanie do

serwera w ustalonym wcześniej odstępie czasu. Chatbot zaimplementowany w aplikacji czat okno działa następująco:

Ustawiony wcześniej interwał czasowy w funkcji setTimeout, uruchamia funkcję chatBot(), której zadaniem jest wysłanie żądania do pliku chatbot.php po stronie serwera. Chatbot.php tworzy obiekt klasy Chat i realizuje metodę chatBot(). Metoda ta pobiera najnowszą wiadomość zapisaną w tabeli main\_room i dzieli ją na pojedyncze słowa, które następnie zapisuje do tablicy.

Tak przygotowana tablica stanowi strukturę do przeszukania pod kątem istnienia w niej słów, które znajdują się również w tabeli chatbot w bazie MySQL. Jeśli słowo z tablicy pasuje do któregoś ze słów w tej tabeli, to realizuje się statycznie metoda dodająca wiadomości do bazy. Metoda sendMessage() zapisuje nową wiadomość do tabeli main\_room, gdzie treść tej wiadomości jest zgodna ze słowem w tabeli chatbot.

Struktura tabeli chatbot w bazie MySQL:



chat_okno.chatbot	
chatbot_id	int(50)
chatbot_word	varchar(50)
chatbot_sentence	varchar(200)

Przykładowe sentencje odpowiadające słowom tak i nie:

chatbot_id	chatbot_word	chatbot_sentence
1	tak	tez tak sadze, jak zawsze warto pomyslec o tym
2	nie	nie, nie...chociaz trudno tak jednoznacznie stwierd...

Jak to wygląda na czacie? Artbax, to realna osoba, a ewa i ala to chatbot.

[2015-08-23 22:32:24] <b>artbax:</b> tak
[2015-08-23 22:33:54] <b>ewa:</b> <i>tez tak sadze, jak zawsze warto pomyslec o tym</i>
[2015-08-23 22:34:19] <b>artbax:</b> <i>naprawde ewa? tez tak sadzisz? a moze nie</i>
[2015-08-23 22:34:24] <b>ala:</b> <i>nie, nie...chociaz trudno tak jednoznacznie stwierdzic..</i>

Pomysł na chatbota powstał w ciągu jednego dnia. Również tego samego dnia program chatbot został zaimplementowany na czacie okno.

Zamieszczam fragment notatek, które pozwoliły mi zrealizować pomysł na chatbota. Wykorzystałem w nim algorytm dzielenia łańcucha znaków na pojedyncze słowa.

```
1.rozbijam lancuch tekstowy na slowa i umieszczam w tablicy
- zapisuje do zmiennej dlugosc lancucha
- deklaruje zmienna tablicowa
- deklaruje zmienna na slowo
- po napotkaniu spacji zapisuje slowo do tablicy i zeruje zmienna na slowo, petla rusza dalej
return: tablica ze slowami

$wyraz = "Ala ma kota a kot ma Ale";
$length = strlen($wyraz);
$tab = array();
$str = "";

for($i = 0; $i < $length; $i++)
{
    $str = $str.$wyraz[$i];
    if($wyraz[$i] == " ")
    {
        $tab[] = $str;
        $str = " ";
    }
}

$len = strlen($tab);
// tej petli rozpoczne przeszukiwanie bazy
for($j = 0; $j < $len; $j++)
{
    echo $tab[$j]."<br />";
}

2.pobieram z tablicy pierwsze slowo i przeszukuje tabele chatbota
3.jesli znajduje slowo w tabeli chatbota, zapisuje do tabeli pokoju sentencje, ktora odpowiada temu slowu
4. jesli nie znajde slowa, zapisuje do tabeli pokoju losowa sentencje, losuje z tabeli chatbota
```

## 7.2 Implementacja statyczna

Implementacja statyczna jest podobna, ale skrypt chatbota, nie przeszukuje wiadomości, lecz dodaje do tabeli main\_room losową wiadomość zapisaną w tabeli chatbot w bazie MySQL. Mogą to być informacje o studiach, dowcipy, przypadkowe zdania i inne efektowne wiadomości, które wyświetlane wśród tekstów napisanych przez rzeczywiste osoby czatujące, mogą sprawiać wrażenie działania prawdziwego uczestnika czata.

Zamieszczam również fragment notatek:

```

CHATBOT PHP:
tabela chatbot, skrypt php
tabela w bazie z id, slowo klucz, zdanie., kilkaset hasel kluczowych...kilka tysiecy,
chatbot co 2 minuty przeszukuje ostatni wpis na czacie, jesli znajduje slowo klucz, dodaje do bazy wp:
0.losowanie imienia chatbota z tablicy dostepnych imion, losowanie koloru i stylu tekstu...
1.przeszukiwanie tekstu // algorytm slownikowy
2.jesli znaleziono slowo, to dodaje do tabeli czata sentencje
3. jesli nie znajdzie, dodaje losowa sentencje domyslne..

Implementacja:
inicjator.js - skrypt z interwalem czasowym co 60000 (minuta)
chatbot.php - przeszukiwanie bazy, porownywanie z tabela chatbot (id, chatbot_word, chatbot_sentence)
return: sentence + losowy nick z tablicy nickow
kilka chatbotow, rozne osobowosci, rozne plcie

chatbooty bez przeszukiwania, z tabela wiedzy rand() w php
(id, sentencja)
1.powazny, udzielajacy informacji na temat studiowania w OKNO // 5 minut
2.zadajacy pytania // 3 minuty
3.opowiadajacy dowcipy // 6minut
4.snujacy refleksje // 4 minuty
5.przeklinajacy w dopuszczalny sposob (np. niech to jasny gwint, znowu sie nie udalo...) // 10 minut

```

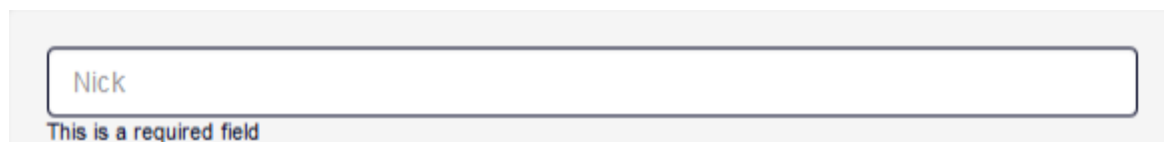
## 8. Przypadki użycia

### 8.1 Rejestracja nowego użytkownika

Po wpisaniu adresu strony [www.czatokno.warszawa.pl](http://www.czatokno.warszawa.pl) w oknie przeglądarki pojawia się strona startowa aplikacji czat okno. Na stronie powitalnej znajdują się informacje o ilości zarejestrowanych użytkowników i użytkowników online, czyli tych, którzy aktualnie czatują. Nieco niżej znajduje się formularz logowania do którego należy wpisać adres email i hasło. Po zalogowaniu użytkownik uzyskuje dostęp do czata i pokoju rozmów. Jeszcze niżej znajduje się przycisk służący do rejestracji nowego użytkownika. Po naciśnięciu przycisku pojawia się formularz rejestracyjny. Należy wypełnić wszystkie pola.

Działanie skryptów odpowiedzialnych za rejestrację.

Po stronie klienta, a więc po stronie przeglądarki działa skrypt JQuery o nazwie register.js, który sprawdza, czy wszystkie pola formularza zostały wypełnione. Jeśli któreś z pól formularza pozostanie puste, wyświetla się komunikat informujący o konieczności wypełnienia tego pola formularza.



The image shows a web form with a single input field labeled 'Nick'. The field has a red border, indicating it is required. Below the field, the text 'This is a required field' is displayed in a small, red font.



W skrypcie register.js odpowiednia funkcja sprawdza pola formularza. Jeśli, któreś z pól jest klasy required i jednocześnie zostało niewypełnione, funkcja dodaje klasę error i wyświetla komunikat o wymaganym polu.

Jednak walidacja po stronie klienta nie polega tylko na sprawdzeniu wypełnienia wszystkich pól formularza. Dwa pola formularza muszą być unikatowe. Jest to pole Nick, który nie może się powtórzyć i pole adres email, który też nie powinien być powtarzany. Nowe konto użytkownika charakteryzuje się unikatowym loginem(nick) i niepowtarzalnym adresem email. W celu spełnienia tych dwóch warunków zrealizowaliśmy dwa dodatkowe skrypty sprawdzające oba pola formularza jeszcze przed utworzeniem nowego konta. Walidacje tych dwóch pól realizuje się przy pomocy dwóch skryptów JQuery: validate\_nick.js i validate\_email.js. W skryptach tych wykorzystujemy asynchroniczną komunikację z serwerem za pomocą funkcji ajax. Ajax działa asynchronicznie, czyli komunikuje się z serwerem bez odświeżania całego dokumentu. Aktualizowany jest tylko wybrany fragment struktury drzewa DOM. Struktura funkcji JQuery jest bardzo prosta. Po stronie serwera działa skrypt php, który realizuje zapytanie do bazy polegające na przeszukiwaniu bazy i sprawdzeniu czy podany nick lub email już istnieje. Jeśli okaże się, że nick lub email wpisany przez nowego użytkownika już istnieje w bazie, skrypt JQuery wyświetla stosowny komunikat. Skrypt validate\_nick.js:

```
function nick_validate()
{
    var value = $('#nick').val();

    $.ajax({
        type: 'POST',
        cache: false,
        url: "http://localhost/czatokno/php/nick_validate.php",
        data: {
            'value': value
        },
        dataType: 'text',

        success: function(data) {

            if(data == 1)
            {
                $('#nick_error').text('ten nick jest juz zajety');
            }
            else
            {
                $('#nick_error').text('');
            }
        }
    });
}
```

Skrypt validate\_nick.php po stronie serwera:

```
class nickValidate extends dbConnection
{
    public function checkNick($value)
    {
        $sql = 'SELECT user_nick FROM chat_user WHERE user_nick="' . $value . '"';

        $log = $this->db_conn->prepare($sql);
        $log->bindParam(':user_nick', $value, PDO::PARAM_STR);
        $result = $log->execute();
        $rows = $log->fetch();
        $n = count($rows);

        if ($n == 1)
        {
            echo 0; // true
        }
        else
        {
            echo 1; // false
        }
    }
}

$nick_validate = new nickValidate();
$nick_validate->checkNick($_POST['value']);
```

## 8.2 Logowanie

Użytkownik wpisuje adres e-mail i hasło jakie podał podczas rejestracji. Po naciśnięciu przycisku Wyślij, zawartość pól formularza logowania przesyłana jest do pliku php metodą post. W skrypcie logging.php tworzy się obiekt klasy User, a następnie realizuje się metoda emailAndPass. Metoda przyjmuje dwa argumenty, adres e-mail i hasło. Po pozytywnej weryfikacji tworzy się nowa sesja użytkownika.

```
session_start();
$_SESSION['access'] = true;
$_SESSION['user_nick'] = $nickname;
$_SESSION['user_email'] = $email;
header('Location: http://localhost/czatokno');
```

## 8.3 Dodawanie wiadomości na czacie

Użytkownik wpisuje wiadomość w dolnym obszarze głównego okna czata.

Tekst wpisany w pole textowe można dodatkowo sformatować tzn. pogrubić, zastosować pochylenie, zmienić kolor czcionki i dodać emotka. Za wysłanie nowej wiadomości odpowiedzialny jest skrypt chat.js w którym znajdują się wszystkie główne funkcje czata po stronie klienta. W skrypcie chat znajduje się funkcja JQuery. Po zatwierdzeniu wiadomości poprzez kliknięcie w button funkcja JQuery obsługuje to zdarzenie najpierw ustawiając wartości zmiennych wpisanego tekstu, a potem dzięki ajax i metodzie post wysyła kompletną wiadomość na serwer do skryptu php, który doda nowy wpis do tabeli w bazie MySql. Fragment kodu funkcji:

```
$('#submit_button').click(function(event)
{
    var mode = 'sendMessage';
    var bold = $('#text_bold').attr('value');
    var italic = $('#text_italic').attr('value');
    var color = $('#text_color').css('background-color');
    var message = $('#content').val();

    if(message == '')
    {
        event.preventDefault();
    }
    else
    {
        $.ajax({
            type: "POST",
            url: "http://localhost/czatokno/php/chat.php",
            cache: false,
            data: {
                'mode': mode,
                'message': message,
                'bold': bold,
```

Po stronie serwera skrypt chat.php stanowi kontroler obsługujący żądanie wysłane asynchronicznie ze skryptu JQuery. W zależności od tego jaką wartość ma zmienna \$mode odebrana metodą post, realizuje się inna metoda klasy Chat. W przypadku dodawania nowej wiadomości wykonuje się metoda sendMessage, która przyjmuje 7 argumentów. Najważniejsze to zmienna przechowująca informacje kto wysłał wiadomość, gdzie wiadomość ma zostać zapisana i treść wiadomości. Zmienna \$target przechowuje nazwę czat roomu z którego została wysłana wiadomość.

## 8.4 Formatowanie tekstu

Użytkownik może nadać szczególny charakter wpisanej wiadomości. Może tekst pogrubić lub pochylić. Służą do tego obszary oznaczone ikonami B, I oraz U. Funkcja JQuery odpowiedzialna za formatowanie

```
$(".text_bold").click(function() {      // pogrubienie, pochylenie, podkreślenie
    var checkItem = $(this).attr('value');
    if(checkItem == 0){
        $(this).addClass('sel_item'); //zmienia kolor tła
        $(this).attr('value','bold');
    }
    else {
        $(this).removeClass('sel_item');
        $(this).attr('value','0');
    }
});
```

## 8.5 Hiperłącze

Kiedy użytkownik ma zamiar opublikować link, może skorzystać dodawania hiperłącza obsługiwane przez skrypt JQuery. Po dodaniu linku, treść wpisana w miniformie jest formatowana w zależności od tego czy została wpisana z przedrostkiem http:// czy też nie. Po dodaniu linku, inni użytkownicy czata mogą kliknąć w sformatowane hiperłącze w oknie czata.

## 8.6 Zmiana opisu i dodanie avatara

Zarejestrowany użytkownik po zalogowaniu może ustawić opis, który będzie widoczny dla innych czatujących. W panelu użytkownika znajduje się forma z polem do wpisania tekstu opisu. W podobny sposób może wzbogacić swój profil zmieniając avatar, czyli graficzny identyfikator, zdjęcie lub rysunek.

## 8.7 Wysyłanie maila

Użytkownik ma możliwość wysłania wiadomości e-mail do wybranego, zarejestrowanego użytkownika. W panelu po lewej stronie jest link do rozwijanej listy wszystkich zarejestrowanych użytkowników. Po rozwinięciu listy i znalezieniu szukanego nicku można kliknąć w opcje email po czym ukaze się forma do wpisania wiadomości. Po zatwierdzeniu tekst wiadomości email zostaje wysłany do skryptu php po stronie serwera.

## 8.8 Zmiana obszaru rozmowy (zmiana pokoju)

Po zalogowaniu użytkownik ma dostęp do pięciu pokoi w których może dodawać posty. Mechanizm swobodnej zmiany obszaru czata polega na przypisaniu nazwy pokoju do zmiennej globalnej target. Pokój główny to main\_czat. Jest to obszar ogólnych rozmów na różne tematy. Menu w górnej części witryny umożliwia łatwe przechodzenie z

jednego pokoju do innego. Po kliknięciu w wybrany pokój np. Rok 3, zmienna globalna `target` nie przechowuje już nazwy `main_czat`, ale teraz w zmiennej tej jest nazwa `room_three`. Zmiana zawartości zmiennej zmienia tylko nazwę tabeli z jakiej skrypt pobierze dane i wyświetli w obszarze rozmów. Po stronie klienta wszystkie funkcje odpowiedzialne za wyświetlanie wiadomości znajdują się w skrypcie `chat.js`. Funkcja `loadChat()` ładuje posty z tabeli w bazie zgodnie z tym jaką aktualnie wartość przechowuje zmienna `target`. Funkcja ta wywołuje się co 500 ms dzięki ustawieniu interwału czasowego w innej funkcji JavaScript, a mianowicie `setTimeout`.

```
setTimeout("loadChat();", updateChatInterval);
```

Drugą ważną funkcją odpowiedzialną za wyświetlanie wiadomości na czacie jest funkcja `readMessages`. Jeśli funkcja `loadChat()` nie napotka żadnych błędów, to wywoływana jest funkcja `readMessages`, która wypisuje 40 ostatnich wiadomości zapisanych do tabeli w bazie danych.

```
function readMessages(data, textStatus)
{
    $('#chat_room')[0].innerHTML = "";

    $.each(data.messages, function(i, message) {
        // Tworzy kod HTML odpowiedzialny za wyświetlenie wiadomości.
        var htmlMessage = "";
        htmlMessage += "<div class='post' style='color:" + message.color + "; font-style:" + message.italic + "; font-weight:" + message.bold + "'>";
        htmlMessage += "<small>[" + message.time + "]</small> <b>" + message.name + " :</b> ";
        htmlMessage += message.post;
        htmlMessage += "</div>";

        var isScrolledDown = ($('#chat_room')[0].scrollHeight - $('#chat_room')[0].scrollTop <=
            $('#chat_room')[0].offsetHeight);

        $('#chat_room')[0].innerHTML += htmlMessage;

        $('#chat_room')[0].scrollTop = isScrolledDown ? $('#chat_room')[0].scrollHeight : $('#chat_room')[0].scrollTop;
    });
}
```

Dodatkowo należy wyjaśnić, że skrypty php działające po stronie serwera wysyłają wiadomości zwrotne w formacie JSON. Po stronie serwera również oczekuje skrypt `chat.php`. Skrypt ten realizuje taką metodę klasy, jaka zostanie przesłana postem przez funkcję Jquery. W przypadku pobierania wiadomości z tabeli, czyli ładowania wiadomości czat, realizuje się wywołanie metody `loadChat($target)`.

Fragment kodu metody `loadChat($target)`:

```

$response = array();
$response['messages'] = array();

while($row = $q->fetch())
{
    $message = array();
    $message['name'] = $row['user_room'];
    $message['post'] = $row['message_room'];
    $message['time'] = $row['date_room'];
    $message['color'] = $row['color_room'];
    $message['italic'] = $row['italic_room'];
    $message['bold'] = $row['bold_room'];
    array_push($response['messages'], $message);
}

return $response;
}

```

## 8.9 Czat prywatny

Ważną właściwością aplikacji czat okno, jest możliwość prowadzenia prywatnej rozmowy między dwoma użytkownikami. Zalogowany użytkownik klikając w nick innego użytkownika znajdujący się na liście zalogowanych użytkowników uruchamia zapytanie SQL, które kreuje tymczasową tabelę w bazie o nazwie złożonej z dwóch nicków. Tabela stanowi prywatny obszar rozmowy dla dwóch uczestników czata. Obszar ten jest dostępny tylko dla dwóch użytkowników.

Screen pokazujący, gdzie wyświetla się button do prywatnego czata:



## 9. Screeny z działającej instancji aplikacji czat okno uruchomionej na hostingu [www.czatokno.warszawa.pl](http://www.czatokno.warszawa.pl)

### 9.1 Panel admina 1

Admin	Data	Nick	Imie	Nazwisko	Plec	Opis	Email	Status	Poziom	Edit	Del	Ban
Wyloguj	2015-08-01 17:04:15	artbax	Artur	Borkowski	mezczyzna	czas powoli konczyc zabawe z tym projektem, bo inne wyzwania czekaja w sesji wrzesniowej	artbaxcpp@gmail.com	online	admin	edit	del	ban
Uczestnicy	2015-08-16 10:06:40	maribiel	Mariusz	Bielawski	mezczyzna	Uzytkownik jeszcze nic o sobie nie napisal..	bielawski.mariusz@gmail.com	offline	user	edit	del	ban
Main room	2015-08-20 19:27:18	john123	John	Brown	mezczyzna	Uzytkownik jeszcze nic o sobie nie napisal..	john@wp.pl	online	user	edit	del	ban
Room one	2015-08-20 19:28:48	anna	Anna	Pijawka	kobieta	Uzytkownik jeszcze nic o sobie nie napisal..	anna@gmail.com	online	user	edit	del	ban
Room two												
Room three												
Room four												

### 9.2 Panel admina 2

Admin	[29][2015-08-18 09:46:11] artbax: WITAJ W POKOJU MAIN CHAT 😊 delete
Wyloguj	[30][2015-08-18 09:46:47] artbax: Po testach rafała wymiotlo całą bazę delete
Uczestnicy	[31][2015-08-19 09:54:10] artbax: underline 🐱 delete
Main room	[32][2015-08-19 09:54:37] artbax: dodałem underline..kolejna funkcja jquery delete
Room one	[33][2015-08-19 09:55:20] artbax: do zmiany jest jeszcze wyglad zalogowanych uzytkownikow po prawej stronie delete
Room two	[34][2015-08-19 09:56:22] artbax: zmieniłem też logowanie, teraz hasłem jest posolony md5...wiec w bazie sa hashe delete
Room three	[35][2015-08-19 09:57:32] artbax: z readMessages w pliku js usunąłem kontrole scrollbar, który uniemożliwiał odczytanie wiadomości, które zostały dodane wcześniej delete
Room four	[36][2015-08-19 09:58:30] artbax: loadChat() co pół sekundy wyciąga 40 ostatnich wpisów z bazy.. delete
	[37][2015-08-19 09:59:23] artbax: zabezpieczyłem formularz rejestracyjny, do inputów dodałem maxlength, czyli ograniczenie długości wpisywanego tekstu delete

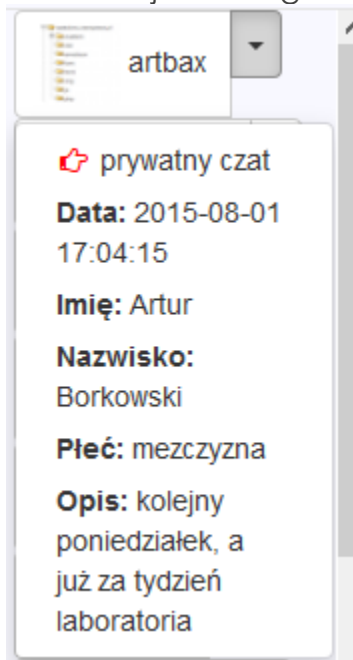
### 9.3 Widok ogólny



### 9.4 Screen paska do wpisywania wiadomości





### 9.5 Informacje o zalogowanym uzytkowniku i link do prywatnego czata







## 9.6 Rozwijana lista zarejestrowanych uczestników z opcją wysłania wiadomości email


 artbax

 Wyloguj

MAIN ROOM

 Avatar

 Dodaj opis

 Uczestnicy

nick	email	status
artbax	email	online
maribiel	email	online
john123	email	online
anna	email	online
monika	email	online
rafal	email	online
ewa	email	offline
bronek	email	online
yeti456	email	online

## 10. Podsumowanie

Jesteśmy przekonani, że udało nam się zaprojektować i zaimplementować aplikację webową służącą do komunikacji tekstowej. Jest to działająca aplikacja z przyjaznym dla oka interfejsem. Mamy również nadzieję, że przedstawiliśmy szczegółowo naszej implementacji wystarczająco dokładnie i wszystkie mechanizmy funkcjonalności, które są dostępne w aplikacji, zostały zaprezentowane możliwie wyczerpująco. Jednocześnie zdajemy sobie sprawę, że nie wszystkie elementy zostały całkowicie ukończone i niektóre z nich wymagają dalszej rozbudowy np. Panel administratora, a jeszcze inne mogły by zostać zaprogramowane zupełnie inaczej. Ostatecznie jednak oddajemy ukończony projekt gotowy do realizacji zadań jakie przed nim postawiliśmy.

## Bibliografia

1. „*Ajax i PHP. Tworzenie interaktywnych aplikacji internetowych*” Bogdan Brinzarea-lamandi, Cristian Darie, Audra Hendrix
2. „*jQuery. Poradnik programisty*” Włodzimierz Gajda
3. „*Mistrz PHP. Pisz nowoczesny kod*” Davey Shafik, Lorna Mitchell, Matthew Turland
4. „*jQuery 2.0 Development Cookbook*” Leon Revill