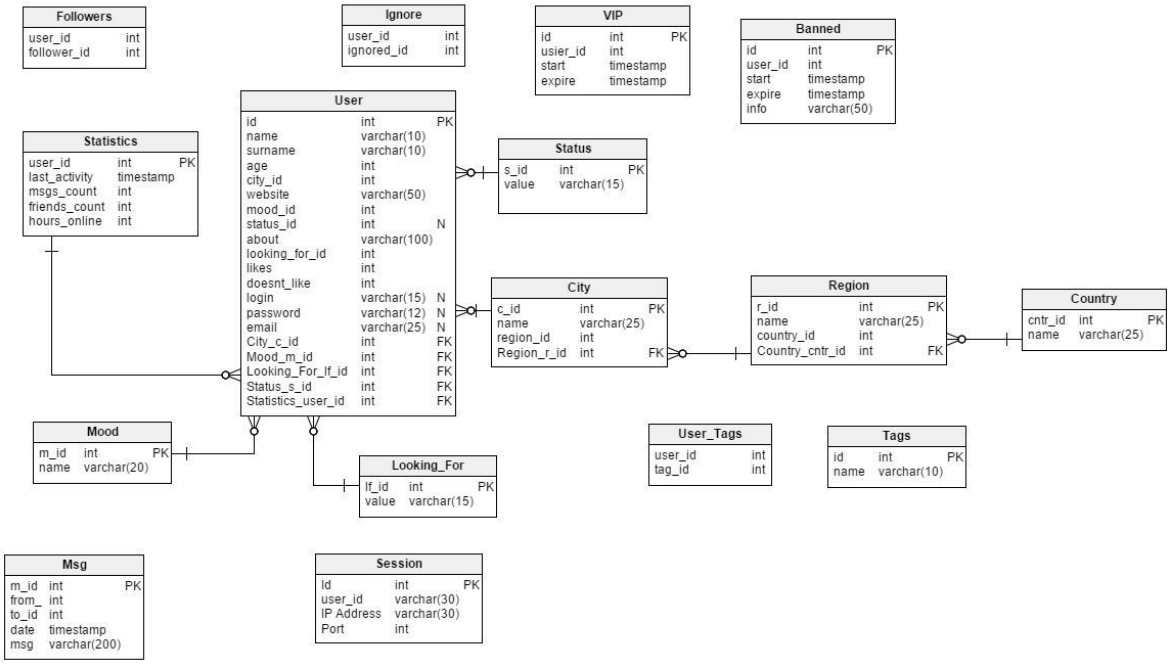
Jakub Niżyński

Bazy danych - Projekt

Komunikator internetowy

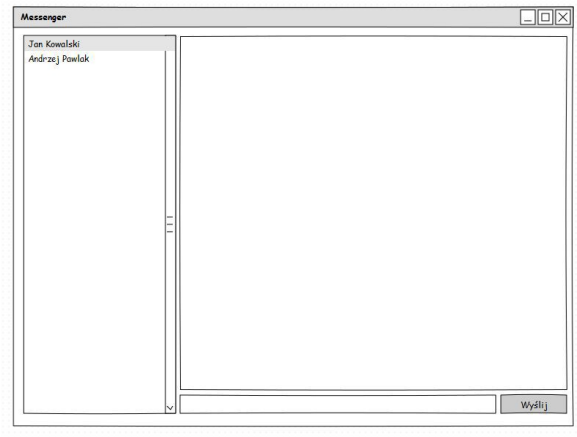
Struktura Bazy danych



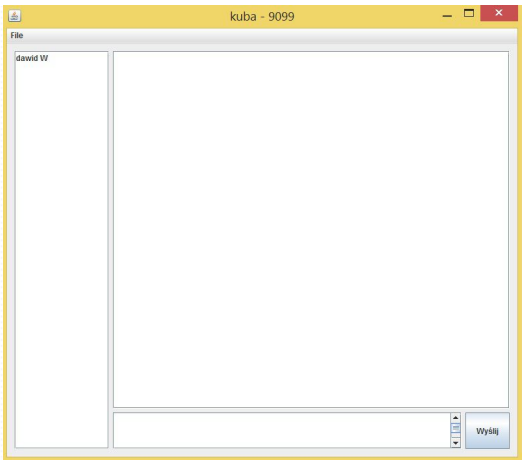
Opis projektu:

Program został napisany w Javie wraz z JDBC. Jako DBMS użyłem MySQL Workbench.

Model GUI programu:



Screen:



Aplikacja składa się z dwóch współdziałających ze soba aplikacji. Części serwera, oraz klienta.

Serwer:
Posiada 3 klasy:
1) Server:
Jej głównym zadaniem jest informowanie o logowaniu się, niepoprawnych danych do logowania lub końcu sesji. Robi to za pomocą metody start(), która parsuje komunikaty otrzymywane od klienta.
2) DatabaseHelper:
Służy do obsługi bazy danych.

getUserId - sprawdza czy user o podanym loginie i haśle istnieje w bazie danych.

```
statement = connection.createStatement();
resultSet = statement.executeQuery("SELECT Id FROM messenger.user WHERE Login = '" + login + "' AND
Password = '" + password + "';");
resultSet.next();
result = resultSet.getInt(1);
```

addMessage - dodaje wiadomości które wysłał user do bazy danych, dzięki temu może ona być pobrana przez drugiego rozmówcę.

```
String sql = "INSERT INTO `messenger`.`msg` (`FromID`, `ToID`, `CreateDate`, `Message`) VALUES (" +
from + ", " + to + ", '" + date + "', '" + message + "');";
result = statement.executeUpdate(sql);
```

addSession - dodawanie sesji użytwkonika

```
statement = connection.createStatement();
String sql = "INSERT INTO `messenger`.`session` (`Id`, `User`, `IpAddress`, `Port`) VALUES ('" + id
+ "', " + userId + ", '" + ipAddress + "', " + port + ");";
result = statement.executeUpdate(sql);
```

removeSession - usuwanie sesji użytkownika:

```
statement = connection.createStatement();
String sql = "DELETE FROM `messenger`.`session` WHERE User =" + userId + ";";
result = statement.executeUpdate(sql);
```

getPort - pobiera port przez jaki uzytkownik się łączy z serwerem

```
String sql = "SELECT Port FROM `messenger`.`session` WHERE User =" + userId + ";";
resultSet = statement.executeQuery(sql);
resultSet.next();
result = resultSet.getInt(1);
```

3) Client:
Służy do komunikacji z klientem.

Klient:

Jest bardziej złożony i posiada 9 klas. Najważniejsze to:

1) PropertiesHelper - obsługa portu.
2) Program - odświeżanie, tworzenie nowej sesji
3) DatabaseHelper -

getContects - pobieranie listy kontaktów

```
resultSet = statement.executeQuery("SELECT * FROM messenger.user WHERE Id != " +
SessionContext.getCurrentUserId() + ";");


while(resultSet.next()){
   User user = new User(resultSet.getInt(1), resultSet.getString(2) + " " + resultSet.getString(3));
   result.add(user);
}
```

getMessage - pobiera wiadomosc z serwera

```
statement = connection.createStatement();


PreparedStatement st = connection.prepareCall("{call get_messages(?, ?)}");
st.setInt(1, SessionContext.getCurrentUserId());
st.setInt(2, contactId);
resultSet = st.executeQuery();
```

Procedura:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_messages`(
	IN user_id INT,
	IN contact_id INT)
BEGIN
	SELECT *
	FROM messenger.message
	WHERE FromID = user_id
		AND ToID = contact_id
		OR FromID = contact_id
		AND ToID = user_id
	ORDER BY CreateDate;
END
```

changePassword - zmienia hasło

```
statement = connection.prepareCall("{call set_password(?, ?, ?, ?)}");
statement.setInt(1, SessionContext.getCurrentUserId());
statement.setString(2, oldPassword);
statement.setString(3, newPassword);
statement.registerOutParameter(4, BIT);
resultSet = statement.executeQuery();
```

```
result = statement.getBoolean(4);
```

Procedura:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `set_password`(IN userID INT, IN old_password VARCHAR(64), IN new_password VARCHAR(64), OUT succes BIT)
BEGIN
	IF check_password(userID, old_password) = 1 THEN
		UPDATE messenger.user u SET u.Password = new_password WHERE u.Id = userID;
		SET succes = 1;
	ELSE
		SET succes = 0;
	END IF;
END
```

Funkcja pomocnicza:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `check_password`(userID INT, password VARCHAR(64)) RETURNS bit(1)
BEGIN
	RETURN (SELECT COUNT(u.Password) FROM messenger.user u WHERE u.Id = userID AND u.Password = password);
END
```

Widok:

```
CREATE
	ALGORITHM = UNDEFINED
	DEFINER = `root`@`localhost`
	SQL SECURITY DEFINER
VIEW `messenger`.`message` AS
	SELECT
		`m`.`FromID` AS `FromID`,
		`m`.`ToID` AS `ToID`,
		`m`.`CreateDate` AS `CreateDate`,
		`m`.`Message` AS `Message`,
		CONCAT(`u`.`Name`, ' ', `u`.`Surname`) AS `Display name`
	FROM
		(`messenger`.`msg` `m`
		JOIN `messenger`.`user` `u` ON ((`m`.`FromID` = `u`.`Id`)))
```