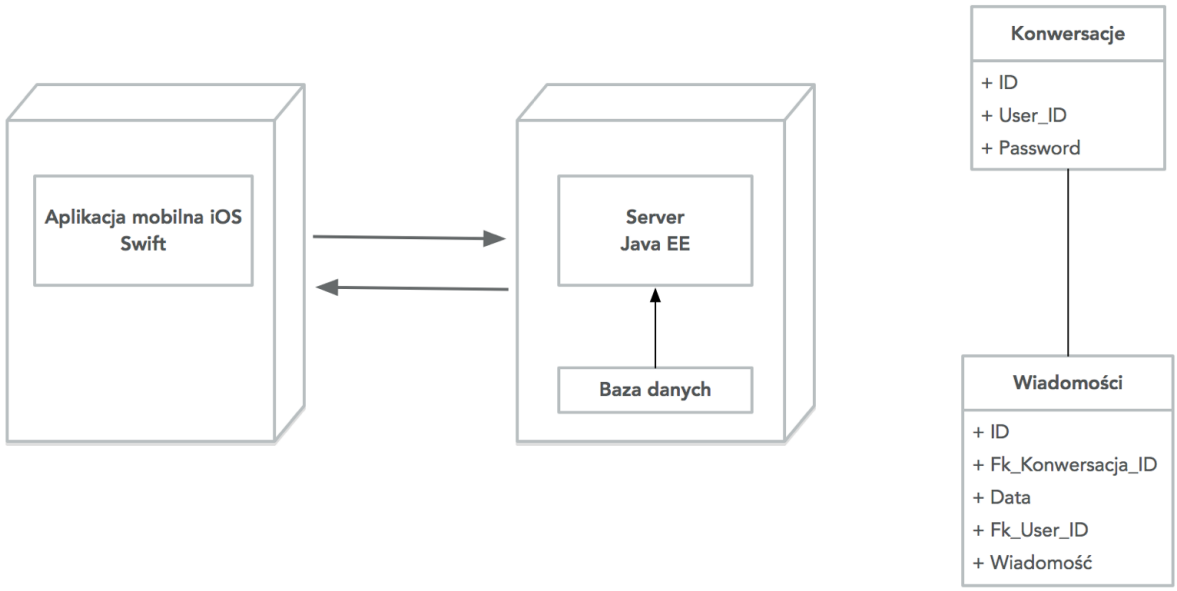
Podstawowe założenia aplikacji:
- serwis internetowy służący do komunikacji wielu osób
- dane o użytkownikach i konwersacjach będą zapisywane w bazie danych
- autoryzacja poprzez login i hasło
- możliwość utworzenia konwersacji dla dwóch lub więcej osób
- możliwość dodawania lub usuwania uczestników konwersacji

Aplikacja mobilna zostanie stworzona w języku Swift stworzonym przez Apple Inc. Dostępna będzie na systemy IOS. Będzie umożliwiała zarejestrowanie nowego użytkownika w systemie oraz zalogowanie się do systemu. Będzie pozwalała na utworzenie konwersacji dla dwóch lub więcej osób, wysyłanie i odbiór wiadomości. Wymiana informacji pomiędzy aplikacją mobilną a serwerem będzie możliwa poprzez komunikację REST. Aplikacja będzie wysyłała odpowiednie żądania do serwera, a następnie odbierała komunikaty zwrotne.

Aplikacja webowa zostanie poprzez takie technologie jak HTML, CSS, Jquery, Bootstrap. Dostępna będzie poprzez przeglądarkę WWW. Będzie umożliwiała zarejestrowanie nowego użytkownika w systemie oraz zalogowanie się do systemu. Będzie pozwalała na utworzenie konwersacji dla dwóch lub więcej osób, wysyłanie i odbiór wiadomości. Aplikacja będzie wysyłała odpowiednie żądania do serwera, a następnie odbierała komunikaty zwrotne.



Rys. 1: Diagram systemu.

Git – jest to nowoczesny, rozproszony system kontroli wersji. Git powstał w 2005 roku na potrzeby zarządzania jądrem Linux-a.

Program GIT dostępny jest do pobrania na stronie: https://git-scm.com/downloads.

Podstawowa konfiguracja GIT-a:

git config -l => sprawdzenie ustawień programu git
git config –global user name ”name” => ustawia autora jako name
git config –global user email ”email” => ustawia email autora jako email

Git śledzi zmiany plików w obrębie konkretnego folderu. Nie ma znaczenia czy folder zawiera kod źródłowy programu komputerowego , rękopis czy stronę WWW. Folder, którego zawartość jest kontrolowana przez Git-a to repozytorium. Repozytoria zawierają specjalny podfolder .git w którym zapisywane są szczegółowo dane o śledzonych plikach.
Do wprowadzania zmian służy specjalna operacja zatwierdzania zmian (commit). Nie jest ona nigdy wykonywana przez Git-a samodzielnie.
Rewizje są kolejnymi stanami projektu.

Repozytorium możemy tworzyć na dwa sposoby:
- inicjalizując nowy projekt,
- klonując istniejące repozytorium.

Inicjaliacja nowego repozytorium:

```
git init
```

Klonowanie istniejącego repozytorium:

```
git clone [adres]
```

Sprawdzenie zmian wprowadzonych w projekcie:

```
git log
```

Aby przywrócić stan projektu do stanu ostatniej rewizji należy wykonać polecenie:

```
git reset –hard
```

Aby przywrócić stan projektu do stanu rewizji określonej jako [SHA-1] należy wykonać polecenie:

```
git reset –hard [SHA-1]
```

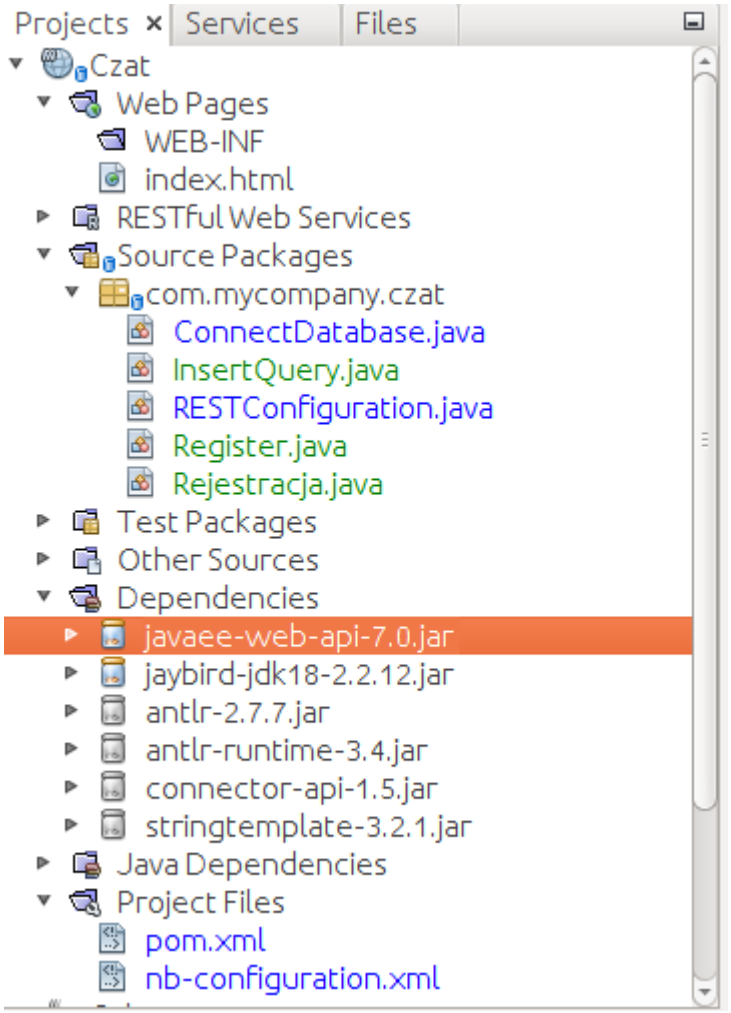
Dołączenie kolejnej rewizji możliwe jest za pomocą komendy:

```
git add -A
git commit -m ”Opis rewizji”
```

Źródło: Git. Rozproszony system kontroli wersji, Włodzimierz Gajda.

Aplikacja serwera zostanie napisana w języku Java. Jako serwer aplikacji użyty zostanie serwer GlassFish, dostępny na licencji GNU GPL. Serwer zostanie napisany zgodnie ze stylem REST. Będzie umożliwiał rejestrację i logowanie użytkowników, przechowywanie danych konwersacji, tworzenie grup konwersacyjnych. Będzie udostępniał i zapisywał odpowiednie dane w bazie danych. Jako system bazodanowy wybrany został system Firebird w wersji 2.5, dostępny na licencji wolnego oprogramowania: InterBase Public License.

Struktura plików serwera:



Kod dotychczas stworzony na potrzeby serwera:

```
package com.mycompany.czat;

import java.sql.SQLException;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

/**
 *
 * @author hubert
 */

//klasa odpowiedzialna za pobranie parametrów rejestracji z adresu URI
@Path("logowanie") //adres URI
public class Rejestracja {

    //pobierz za pomocą metody POST
    @POST
    public boolean Register(@QueryParam("login")
            String login, @QueryParam("password") String password)
            throws ClassNotFoundException, SQLException
    {
        return Register.register(login, password); //zwróć wynik rejestracji
    }

}
```

```
package com.mycompany.czat;

import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 *
 * @author hubert
 */

//klasa odpowiedzialna za rejestrację użytkowników
public class Register {


    static public boolean register(String login, String password)
            throws ClassNotFoundException, SQLException
    {
        String query = "INSERT INTO USERS "
                + "VALUES(?, ?)"; //zapytanie do bazy danych

        PreparedStatement preparedStmt = ConnectDatabase.Connect().prepareStatement(query);
        preparedStmt.setString (1, login);
        preparedStmt.setString (2, password);

        return InsertQuery.ExecuteQuery(preparedStmt); //zwróć wynik wykonania zapytania
    }
}
```

```
package com.mycompany.czat;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

/**
 *
 * @author hubert
 */

//klasa konfiguracyjna aplikacji
@ApplicationPath("/")
public class RESTConfiguration extends Application {

}
```

```
package com.mycompany.czat;

import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 *
 * @author hubert
 */

//klasa odpowiedzialna za wykonywanie zapytań do bazy danych
public class InsertQuery {

    public static boolean ExecuteQuery(PreparedStatement query) throws ClassNotFoundException,
SQLException
    {
        if(ConnectDatabase.Connect() == null) //jeżeli wystąpił błąd zwróć false
        {
            return false;
        }
        else
        {
                return query.execute(); //zwróć wynik wykonania zapytania
        }
    }

}
```

```
package com.mycompany.czat;

import java.sql.Connection;
import java.sql.SQLException;

/**
 *
 * @author hubert
 */

//klasa odpowiedzialna za łączenie z bazą danych
public class ConnectDatabase {

    public static Connection Connect() throws ClassNotFoundException, SQLException
    {
        Class.forName("org.firebirdsql.jdbc.FBDriver"); //załadowanie sterownika

        String URL = "jdbc:firebirdsql:188.213.165.178/3050:"
                + "/czat.fdb?lc_ctype=WIN1250";  //adres do pliku bazy danych

        java.sql.Connection connection = null;

        try
        {
            connection =
                    java.sql.DriverManager.getConnection(URL, "SYSDBA", "root");
            //utworzenie połączenia
        }
        catch(Exception e)
        {
            return null;
        }

        return connection; //zwróć połączenie do bazy danych
    }

}
```