ALTERAÇÕES DE PROJETO

Inicialmente a aplicação servidor estava projetada para conter a seguinte estrutura:

Pacote control.classes:

1. ServerControl;
    1.1. Atributo: String;
    1.2. Atributo: ClienteServer;
    1.3. Atributo: List;
    1.4. Atributo: Gson;
    1.5. Atributo: List;
    1.6. Atributo: RequestManager;
    1.7. Atributo: int;
    1.8. Atributo: ServerSocket;
    1.9. Atributo: Socket;
    1.10. Atributo: List;
    1.11. Atributo: StreamServer;
    1.12. Método: ServerControl( );
    1.13. Método: assinchronous void addContact(String, int);
    1.14. Método: assinchronous void addresser(String, int);
    1.15. Método: assinchronous void authenticateUser(String, int);
    1.16. Método: assinchronous void connector( );
    1.17. Método: assinchronous void isClienteAlive(String, int);
    1.18. Método: assinchronous void logout(String, int);
    1.19. Método: assinchronous void removeContact(String, int);
    1.20. Método: void setServer( );
    1.21. Método: assinchronous void storeUser(String, int);

2. StreamServer;
    2.1. Atributo: BufferedReader;
    2.2. Atributo: PrintWriter;
    2.3. Atributo: Socket;
    2.4. Método: void closeStream( );
    2.5. Método: void createStream(Socket);
    2.6. Método: String readMessage(String);
    2.7. Método: void sendMessage(String);

3. RequestManager;
    3.1. Atributo: int;
    3.2. Atributo: String;
    3.3. Atributo: ServerControl;
    3.4. Atributo: StreamServer;
    3.5. Método: RequestManager(StreamServer, ServerControl, int);

3.6. Método: void run( );

Pacote model.classes:

1. ClienteServer;
   1.1. Atributo: String;
   1.2. Atributo: String;
   1.3. Atributo: String;
   1.4. Atributo: String;
   1.5. Atributo: int;
   1.6. Atributo: String;
   1.7. Atributo: String;

Pacote dao.classes:

1. ClienteDao;
   1.1. Atributo: Conexao;
   1.2. Atributo: ResultSet;
   1.3. Atributo: PreparedStatement;
   1.4. Método: List Consultar( );
   1.5. Método: ClienteServer ConsultarID(String);
   1.6. Método: boolean store (ClienteServer);
   1.7. Método: boolean storeContact(ClienteServer);
   1.8. Método: boolean update(ClienteServer);

2. Conexao;
   2.1. Atributo: Connection;
   2.2. Método: ResultSet executaBusca(String);
   2.3. Método: Connection getConnection( );

No decorrer do desenvolvimento, foram identificadas alterações a serem feitas conforme segue:

Pacote control.classes:

1. ServerControl;
   1.1. Atributo: ClienteDao;
   1.2. Atributo: List;

Foram criados atributos adicionais para acessar o banco de dados e outro para controlar os clientes que estão online.

   1.3. Método: void addContact(String, int);
   1.4. Método: void addresser(String, int);
   1.5. Método: void authenticateUser(String, int);
   1.6. Método: void connector( );
   1.7. Método: void isClienteAlive();

1.8. Método: void logout(String, int);
1.9. Método: void removeContact(String, int);
1.10. Método: void register(String, int);
1.11. Método: void validatePass(String, int);
1.12. Método: void getUserList(int, int, int);

Os métodos acima deixaram de ser assíncronos, pois o sistema ficava em espera e não entrava na execução. Foi também criado um método adicional validatePass, com a finalidade de validar a senha do usuário quando ele for editar seu cadastro. Também criado o método getUserList para gerenciar a busca de listas de contatos para devolver ao usuário.

2. StreamServer;
    2.1. Atributo: int;

Criado um atributo integer, para que o controller consiga gerenciar as requisições vindas de determinado Socket vinculado a esta classe. Este atributo é definido através de um parâmetro no método abaixo.

    2.2. Método: void createStream(Socket, int);

Pacote dao.classes:

3. ClienteDao;
    3.1. Método: boolean removeContact(int, int);
    3.2. Método: void desconectar();

Foi criado o método removeContact que recebe dois ids e faz a desvinculação do contato. Também criado o método desconectar a conexão.

4. Conexao;
    4.1. Método: boolean desconectar();

Criado esse método para fechar a conexão.