



Comunicação entre Sockets

Sistemas paralelos e distribuídos

Alunos: Adrian Grosch e Matheus Parro

Professor: Fernando dos Santos

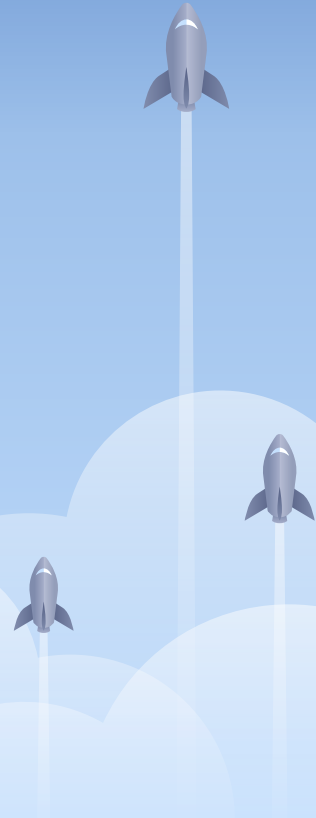
Sockets

- Utilizado para representar um ponto de conexão para uma rede de computadores que utiliza o protocolo TCP/IP.
- O pacote `java.net` contém todas as classes necessárias para criar aplicações de rede. As classes `ServerSocket` e `Socket` também fazem parte desse pacote e são utilizadas para aplicações que fazem uso do protocolo TCP.



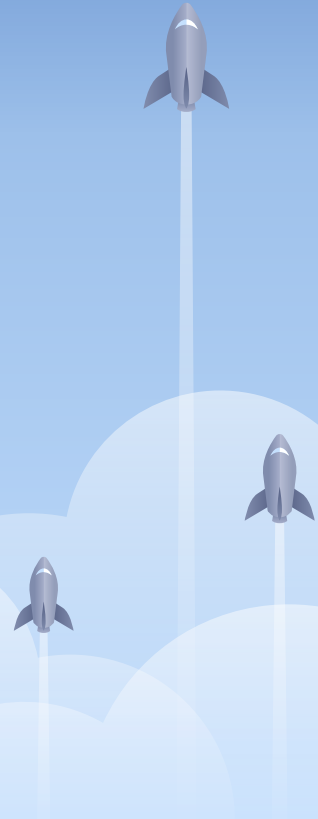
Arquitetura

- Intelli (Interface)
- Java JDK_1.8
- GitHub
- MVC como estrutura do projeto



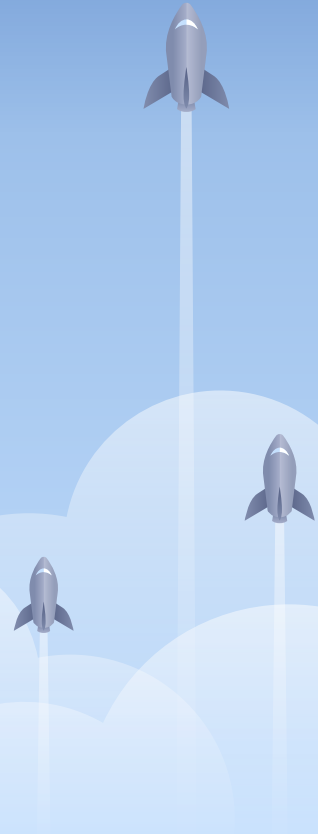
Models

- Sala de aula
- Professor
- Aluno
- Banco de dados
- Servidor
- Pessoa

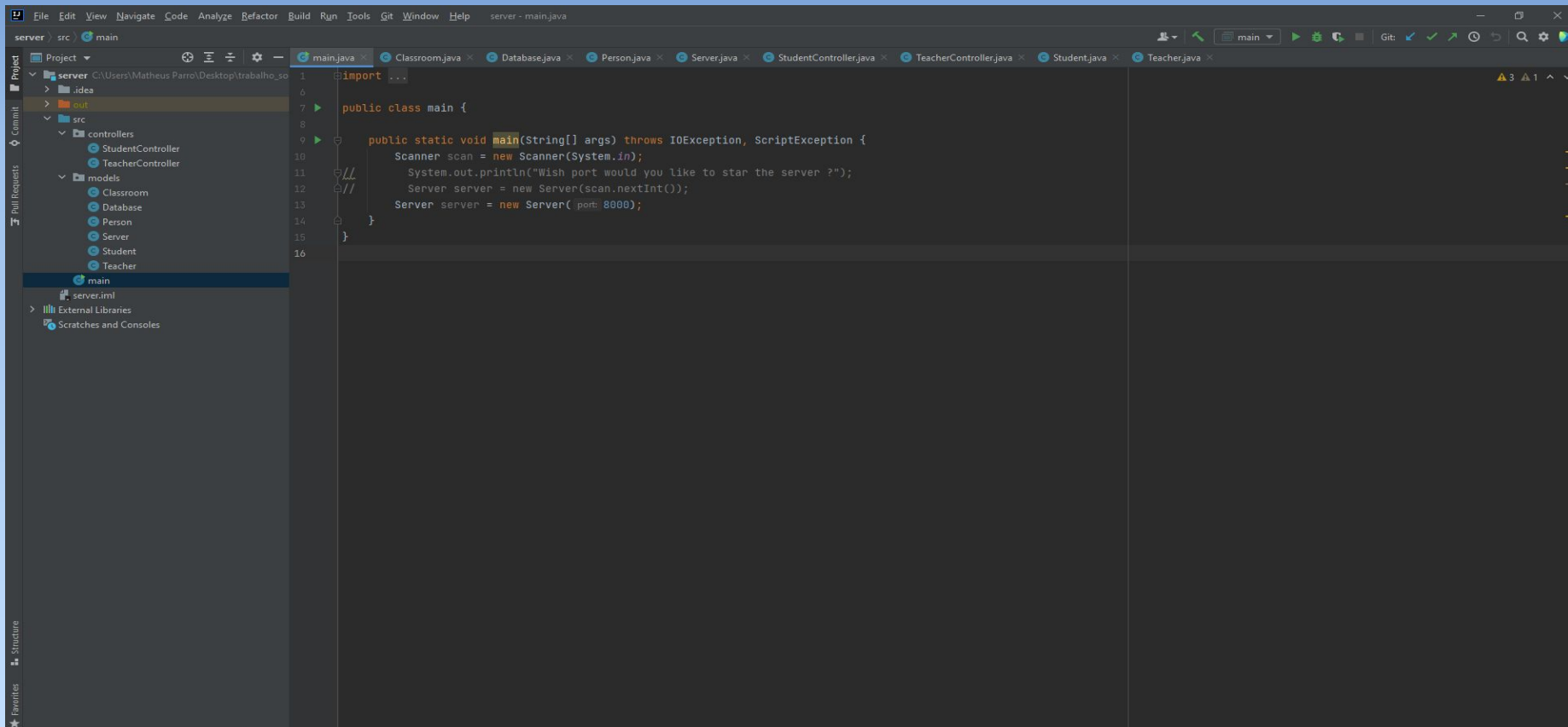


Controllers

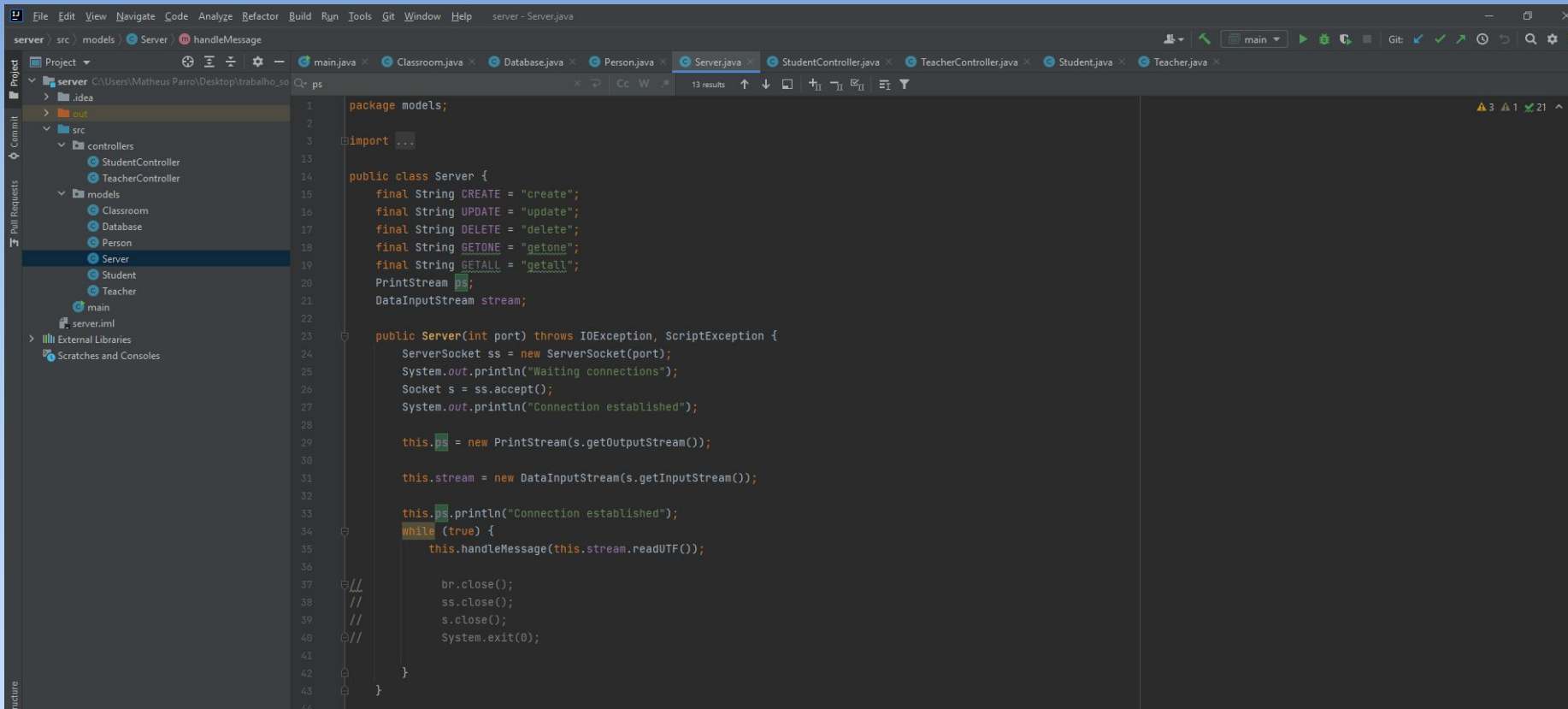
- Sala de aula
- Professor
- Aluno



Código (Server)



Código (Server)



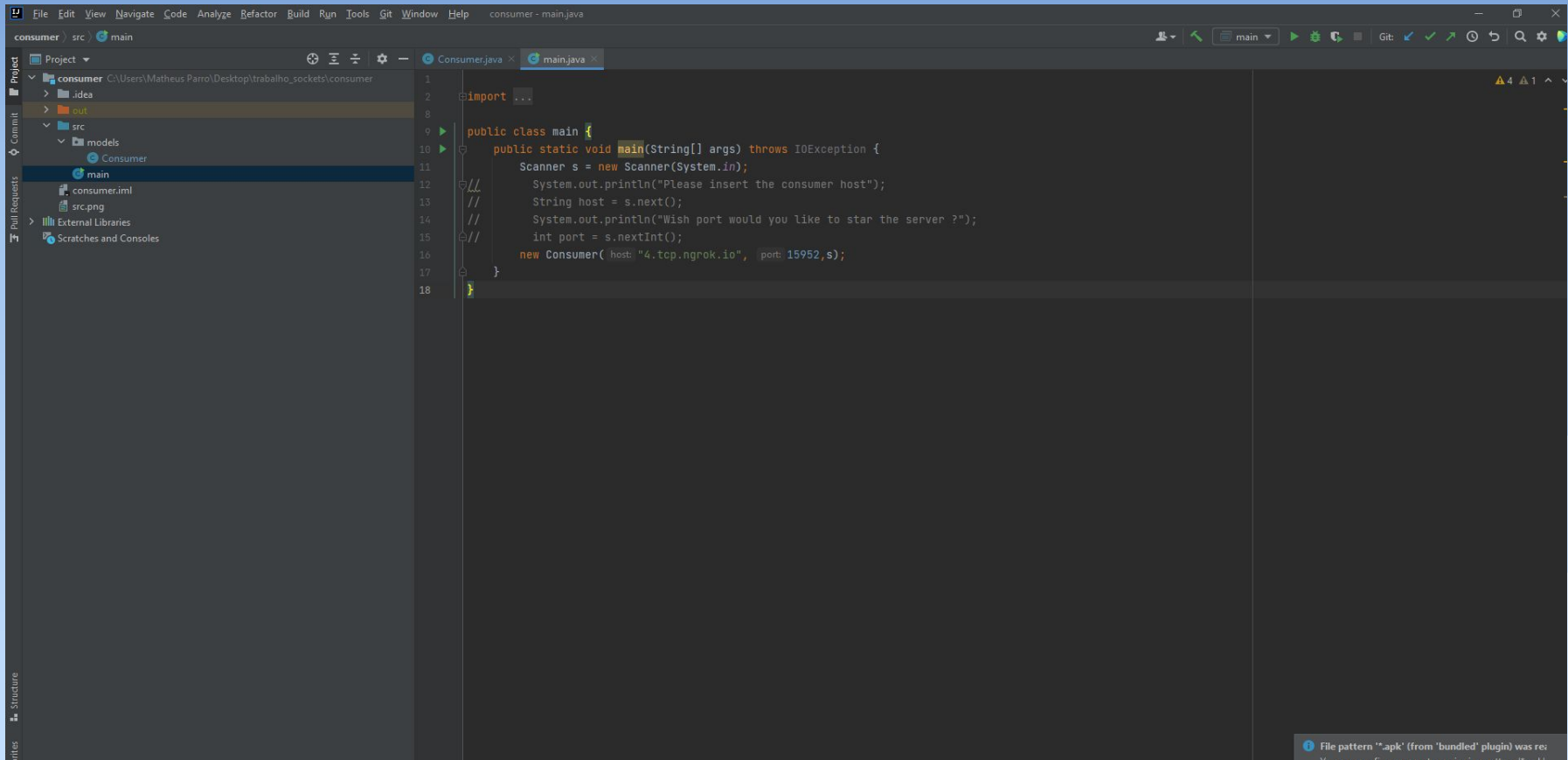
The screenshot shows an IDE window with the title bar "server - Server.java". The menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, and Help. The toolbar shows icons for running, debugging, and other IDE functions. The Project view on the left shows a tree structure with folders like "server", ".idea", "out", "src", "controllers", "models", and "main". The "Server" class is selected in the "models" folder. The main editor area displays the code for the "Server" class, which is a public class that implements a simple HTTP server. The code includes package declarations, imports, and a main method that listens for connections and handles messages.

```
1 package models;
2
3 import ...
4
5 public class Server {
6     final String CREATE = "create";
7     final String UPDATE = "update";
8     final String DELETE = "delete";
9     final String GETONE = "getone";
10    final String GETALL = "getall";
11    PrintStream ps;
12    DataInputStream stream;
13
14    public Server(int port) throws IOException, ScriptException {
15        ServerSocket ss = new ServerSocket(port);
16        System.out.println("Waiting connections");
17        Socket s = ss.accept();
18        System.out.println("Connection established");
19
20        this.ps = new PrintStream(s.getOutputStream());
21
22        this.stream = new DataInputStream(s.getInputStream());
23
24        this.ps.println("Connection established");
25        while (true) {
26            this.handleMessage(this.stream.readUTF());
27
28            br.close();
29            ss.close();
30            s.close();
31            System.exit(0);
32        }
33    }
34 }
```

Código (Server)

```
import ...  
  
public class TeacherController {  
    Teacher teacher;  
  
    public static void create(HashMap<String, String> params, PrintStream ps) {  
        Teacher s = new Teacher(params.get("name"), params.get("cpf"), params.get("address"));  
        Database.teachers.put(s.getCpf(), s);  
        ps.println("Teacher created successfully");  
    }  
  
    public static void update(HashMap<String, String> params, PrintStream ps) {  
        Teacher s = Database.teachers.get(params.get("cpf"));  
        if (s != null) {  
            s.setAddress(params.get("address"));
```


Código (Consumer)

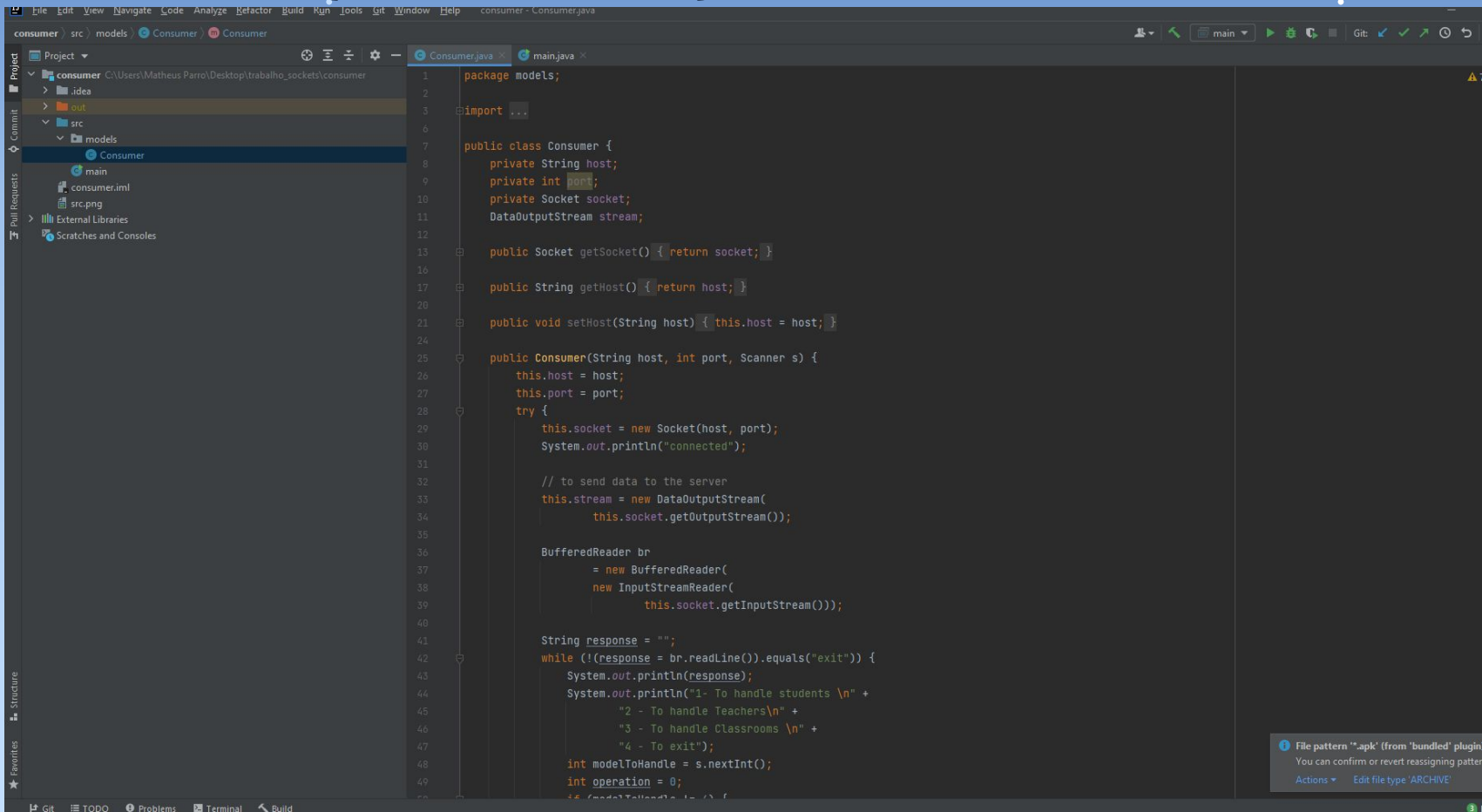


The screenshot shows an IDE window titled "consumer - main.java". The left sidebar displays the project structure for "consumer", including folders like ".idea", "out", "src", and "models", and files like "main". The main editor area shows the code for "main.java".

```
1  import ...
2
3
4
5
6
7
8
9  public class main {
10     public static void main(String[] args) throws IOException {
11         Scanner s = new Scanner(System.in);
12         System.out.println("Please insert the consumer host");
13         // String host = s.next();
14         System.out.println("Wish port would you like to star the server ?");
15         // int port = s.nextInt();
16         new Consumer(host: "4.tcp.ngrok.io", port: 15952,s);
17     }
18 }
```

A status bar at the bottom right indicates: "File pattern '*.apk' (from 'bundled' plugin) was re... You can confirm or avoid reassigning pattern '*.apk'..."

Código (Consumer)



```
1 package models;
2
3 import ...
4
5
6
7 public class Consumer {
8     private String host;
9     private int port;
10    private Socket socket;
11    private DataOutputStream stream;
12
13    public Socket getSocket() { return socket; }
14
15
16    public String getHost() { return host; }
17
18
19
20    public void setHost(String host) { this.host = host; }
21
22
23
24    public Consumer(String host, int port, Scanner s) {
25        this.host = host;
26        this.port = port;
27        try {
28            this.socket = new Socket(host, port);
29            System.out.println("connected");
30
31
32            // to send data to the server
33            this.stream = new DataOutputStream(
34                this.socket.getOutputStream());
35
36            BufferedReader br
37                = new BufferedReader(
38                    new InputStreamReader(
39                        this.socket.getInputStream()));
40
41            String response = "";
42            while (!response.equals("exit")) {
43                System.out.println(response);
44                System.out.println("1- To handle students \n" +
45                    "2 - To handle Teachers\n" +
46                    "3 - To handle Classrooms \n" +
47                    "4 - To exit");
48                int modelToHandle = s.nextInt();
49                int operation = 0;
50                if (modelToHandle == 1) {
```