Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №12
По дисциплине СПП

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Крощенко А. А.

Брест 2021

Лабораторная работа №12

ВАРИАНТ 13

Разработать клиент-серверное оконное приложение на Java с использованием сокетов и JavaFX.

Можно сделать одну программу с сочетанием функций клиента и сервера либо две отдельных (клиентская часть и серверная часть). Продемонстрировать работу разработанной программы в сети либо локально (127.0.0.1).

Игра «Цифровые войны» Генерируется ряд цифр от 0 до 9 в любой последовательности. Длина ряда 20. Например 5,3,6,9,0,8,4,6,1,3,2,4,8,7,0,9,5,2,6,7.
В свой ход каждый игрок может сделать одно из двух возможных в игре действий:
• изменить в меньшую сторону одну из цифр, максимум до 0 (отрицательных величин в игре нет);
• стереть любой ноль и все цифры справа него, сократив таким образом длину полосы.
Проигрывает тот, кто уничтожает последний ноль.

Для разработки выбираем формат клиент сервер.
Клиент — веб-браузер, программой является html страница с логикой подключения к серверу.
Сервер — standalone HTTP+socket сервер на spring web.

Код программы

backend src: live.ilyusha.Lab5Application\

```
package live.ilyusha.lab5;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Lab5Application {

    public static void main(String[] args) {
        SpringApplication.run(Lab5Application.class, args);
    }

}
```

backend src: live.ilyusha.GameController

```
package live.ilyusha.lab5;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.stereotype.Controller;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.List;

@Controller
public class GameController {
    private final GameService gameService;

    @Autowired
```

```
    public GameController(final GameService gameService) {
        this.gameService = gameService;
    }

    @MessageMapping("/start")
    @SendTo("/topic/digit")
    public GameObject greeting(GameObject gameObject) {
        gameService.setNumbers(gameObject.getNumbers());
        return gameObject;
    }

    @GetMapping("/digits")
    public @ResponseBody
    List<Integer> getGameNumbers() {
        return gameService.getNumbers();
    }

    @PostMapping("/digits")
    public @ResponseBody
    void updateGameNumber() {
        gameService.generateNumbers();
    }
}
```

backend src: live.ilyusha.GameObject

```
package live.ilyusha.lab5;

import java.util.List;

public class GameObject {
    private String userName;
    private List<Integer> numbers;

    public String getUserName() {
        return userName;
    }

    public GameObject setUserName(String userName) {
        this.userName = userName;
        return this;
    }

    public List<Integer> getNumbers() {
        return numbers;
    }

    public GameObject setNumbers(List<Integer> numbers) {
        this.numbers = numbers;
        return this;
    }
}
```

backend src: live.ilyusha.GameService

```
package live.ilyusha.lab5;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
```

```java
import java.util.List;
import java.util.Objects;
import java.util.Random;

@Service
public class GameService {
    private final Random random;
    private List<Integer> numbers;

    @Autowired
    public GameService() {
        this.random = new Random();
    }

    public List<Integer> getNumbers() {
        if (Objects.isNull(numbers)) {
            generateNumbers();
        }
        return numbers;
    }

    public void setNumbers(final List<Integer> numbers) {
        this.numbers = numbers;
    }

    public void generateNumbers() {
        this.numbers = new ArrayList<>();
        for (int i = 0; i < 20; i++) {
            this.numbers.add(random.nextInt(10));
        }
    }
}
```

backend src: live.ilyusha.WebSocketConfig

```java
package live.ilyusha.lab5;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import
org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker
;
import
org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import
org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfig
urer;

@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/connect").withSockJS();
```

```
    }
}
```

frontend src: static/index.html

```
<!DOCTYPE html>
<html>
<body style="text-align: center; padding-top: 20px">

<div id="main-content" class="container">
    <div id="startGame">
        <form id="startGameForm" class="form" style="padding-top: 20px;">
            <div>
                <input id="userNameInForm" type="text" placeholder="Username"
required>
            </div>
            <button style="margin-top: 20px;" id="newGame" type="submit">Host
game</button>
            <button id="connect" type="submit">Join existing</button>
        </form>
    </div>
    <button id="disconnect" type="submit">Exit</button>
    <div id="mainGame"></div>
</div>

<div class="modal fade" id="updateDialog" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content" style="padding: 20px">
            <input id="modalChangedNumber" type="number">
            <a href="#" data-dismiss="modal">Close</a>
            <a href="#" id="updateFormUrl">Update</a>
        </div>
    </div>
</div>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/
css/bootstrap.min.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.5.0/
sockjs.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/
stomp.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/
jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/
popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/
bootstrap.min.js"></script>
<script src="/index.js" defer="defer"></script>

</body>
</html>
```

frontend src: static/index.js

```
let stompClient = null
let gameNumber = null
let userName = null

function setConnected(connected) {
    if (connected) {
        $("#startGame").hide()
        $("#disconnect").show()
```

```
    } else {
        $("#startGame").show()
        $("#disconnect").hide()
    }
}

function connect() {
    let socket = new SockJS('/connect')
    stompClient = Stomp.over(socket)
    stompClient.connect({}, function (frame) {
        setConnected(true)
        $.ajax({
            url: "/digits",
            type: 'GET',
            success: function (data) {
                gameNumber = data
                showGameNumber("user")
            }
        })
        stompClient.subscribe('/topic/digit', function (gameObject) {
            let response = JSON.parse(gameObject.body)
            gameNumber = response.numbers
            if (gameNumber.length > 0) {
                showGameNumber(response.userName)
            } else {
                gameIsOver(response.userName)
            }
        })
    })
}

function resetGameNumber() {
    $.ajax({
        url: "/digits",
        type: 'POST'
    })
}

$('#updateDialog').on('show.bs.modal', function (event) {
    let target = $(event.relatedTarget)
    $("#updateFormErrorResponse").text("")
    $(this).find("#modalChangedNumber").val(target.data('number'))
    $("#updateFormUrl").attr("onclick", "updateNumber(" + target.data('id') +
", " +
        target.data('number') + ")")
})

function disconnect() {
    if (stompClient !== null) {
        stompClient.disconnect()
    }
    setConnected(false)
    $("#mainGame").html("")
}

function sendName() {
    stompClient.send("/app/start", {}, JSON.stringify({
        'userName': userName,
        "numbers": gameNumber
    }))
```

```
}

function updateNumber(numberId, number) {
    const newNumber = $("#modalChangedNumber").val()
    gameNumber[numberId] = parseInt(newNumber)
    sendName()
    $('#updateDialog').modal('hide')
}

function showGameNumber(lastUserName) {
    let response = ""
    if (lastUserName === userName) {
        response = response + "<p>Opponent's turn</p>"
        $("#mainGame").css('pointer-events', 'none')
    } else {
        response = response + "<p>Your turn</p>"
        $("#mainGame").css('pointer-events', 'auto')
    }
    $.each(gameNumber, function (i, l) {
        response = response + createNumberButton(i, l)
    })
    $("#mainGame").html(response)
}

function createNumberButton(index, number) {
    if (number > 0) {
        return "<span data-toggle='modal' data-target='#updateDialog' data-
id=" + index + " data-number=" + number + ">\n" +
            " <button data-toggle='tooltip' data-placement='top'>" + number +
"</button>\n" +
            " </span>"
    } else {
        return "<button style=\"background: blue; color: white;\"
onclick='reduceGameNumber(" + index + ")'>" + number + "</button>"
    }
}

function gameIsOver(lastUserName) {
    let response = "<h1>Game Over!</h1>"
    if (lastUserName === userName) {
        response = response + "<h3>You lost</h3>"
    } else {
        response = response + "<h3>You won</h3>"
    }
    $("#mainGame").html(response)
}

function reduceGameNumber(index) {
    gameNumber = gameNumber.slice(0, index)
    sendName()
}

$(function () {
    let clkBtn = ""
    $("#startGameForm").on('submit', function (e) {

        if (clkBtn === "newGame") {
            resetGameNumber()
        }
        connect()
```

```
        userName = $("#userNameInForm").val()
        e.preventDefault()
    })
    $("#connect").click(function (evt) {
        clkBtn = evt.target.id
    })
    $("#newGame").click(function (evt) {
        clkBtn = evt.target.id
    })
    $("#disconnect").hide().click(function () {
        disconnect()
    })
})
```

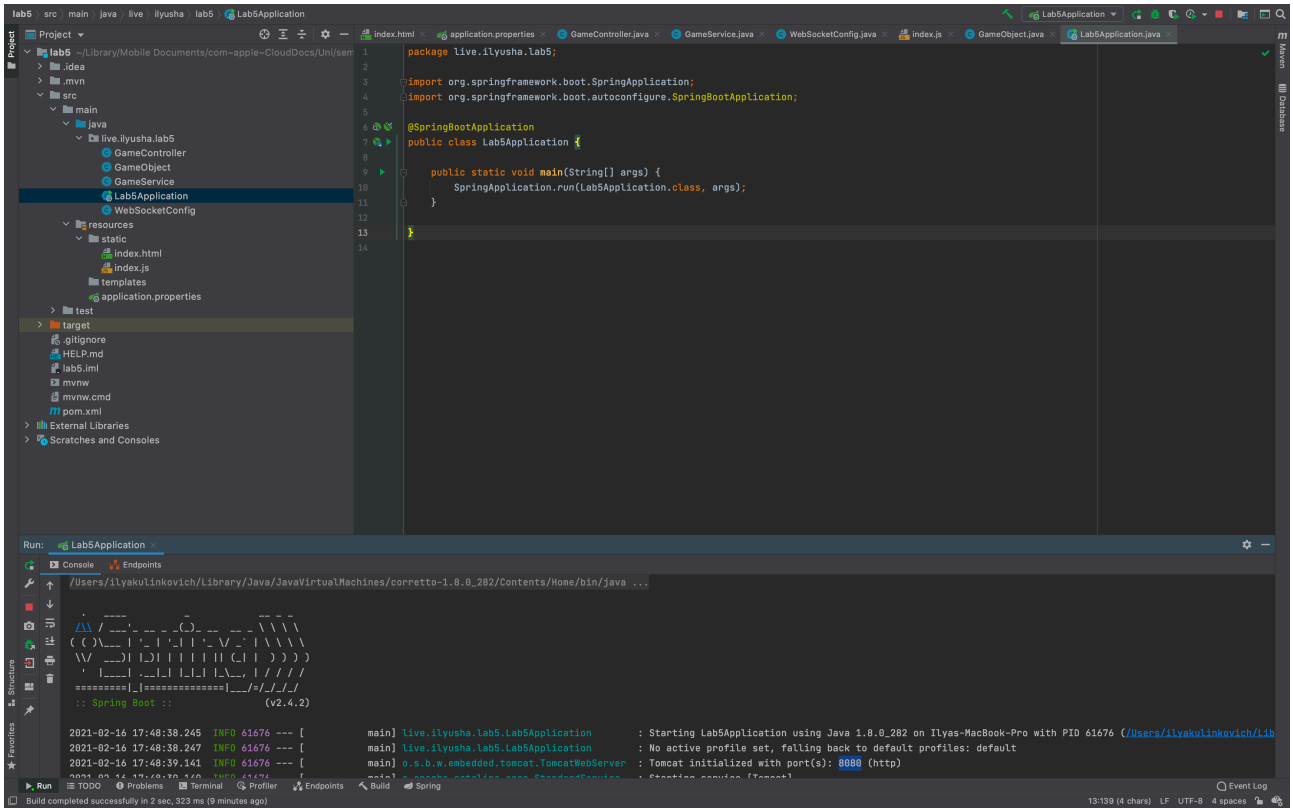
Спецификация ввода

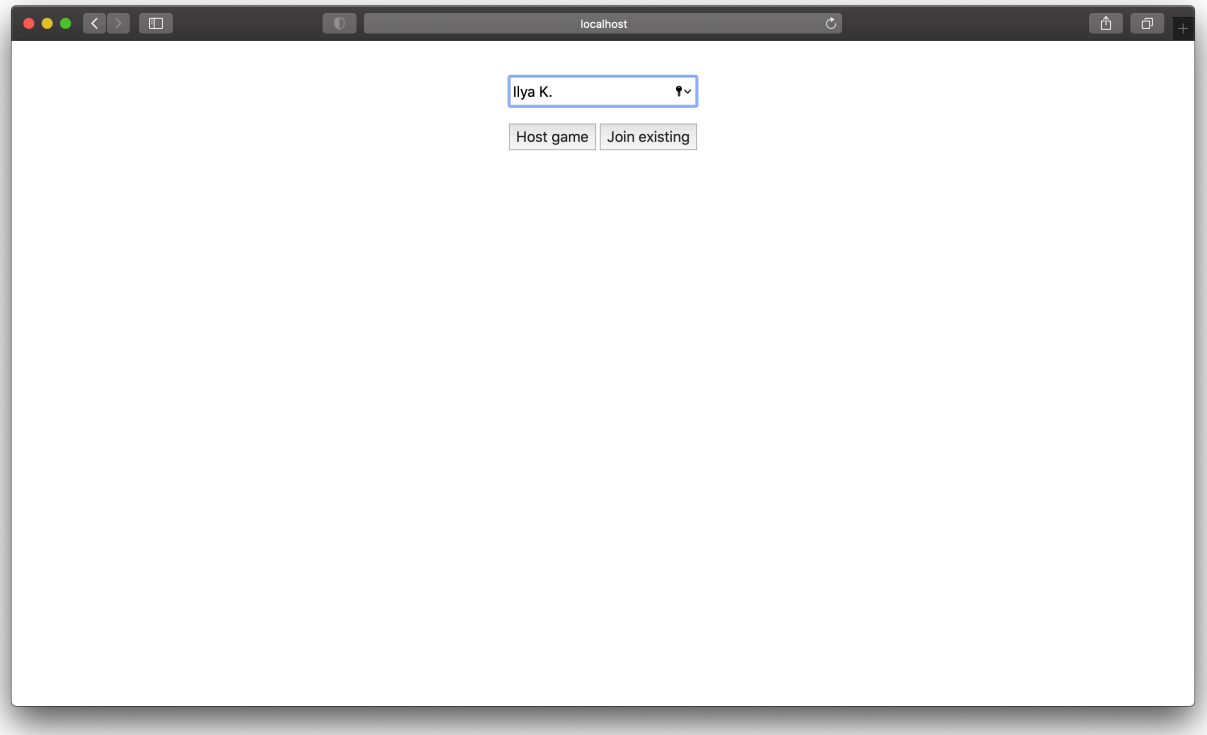
При запуске программа не запрашивает каких-либо данных. В процессе работы клиент может запрашивать данные идентифицирующие пользователя (имя) и его действия в игре.

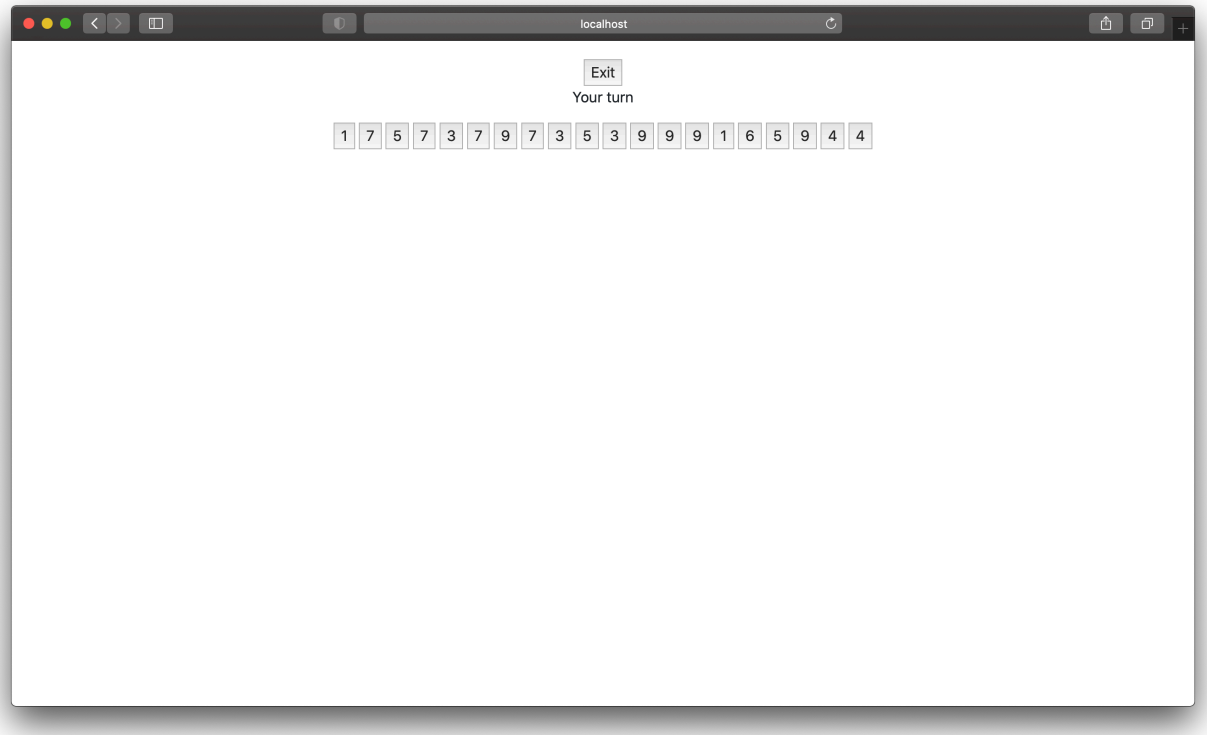
Спецификация вывода

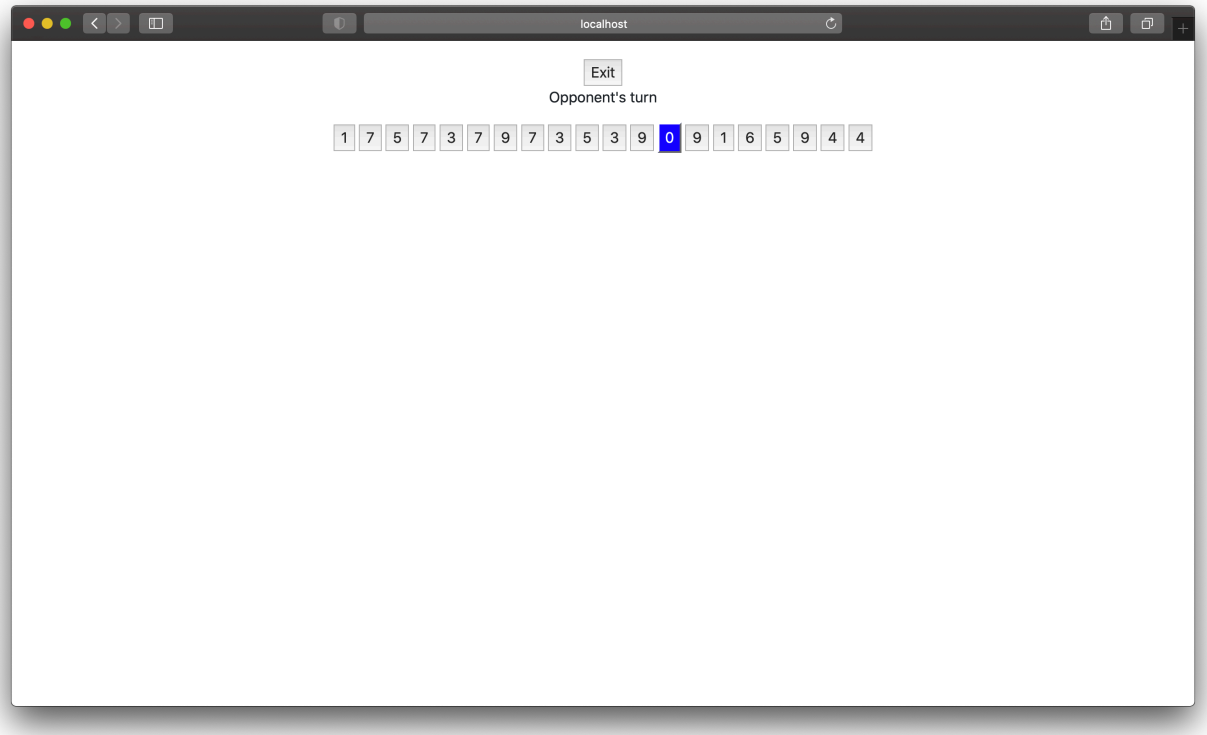
Клиент — веб-приложение по адресу localhost:8080.
Сервер — логи работы веб-сервера сообщающие о подключениях и статусе работы.

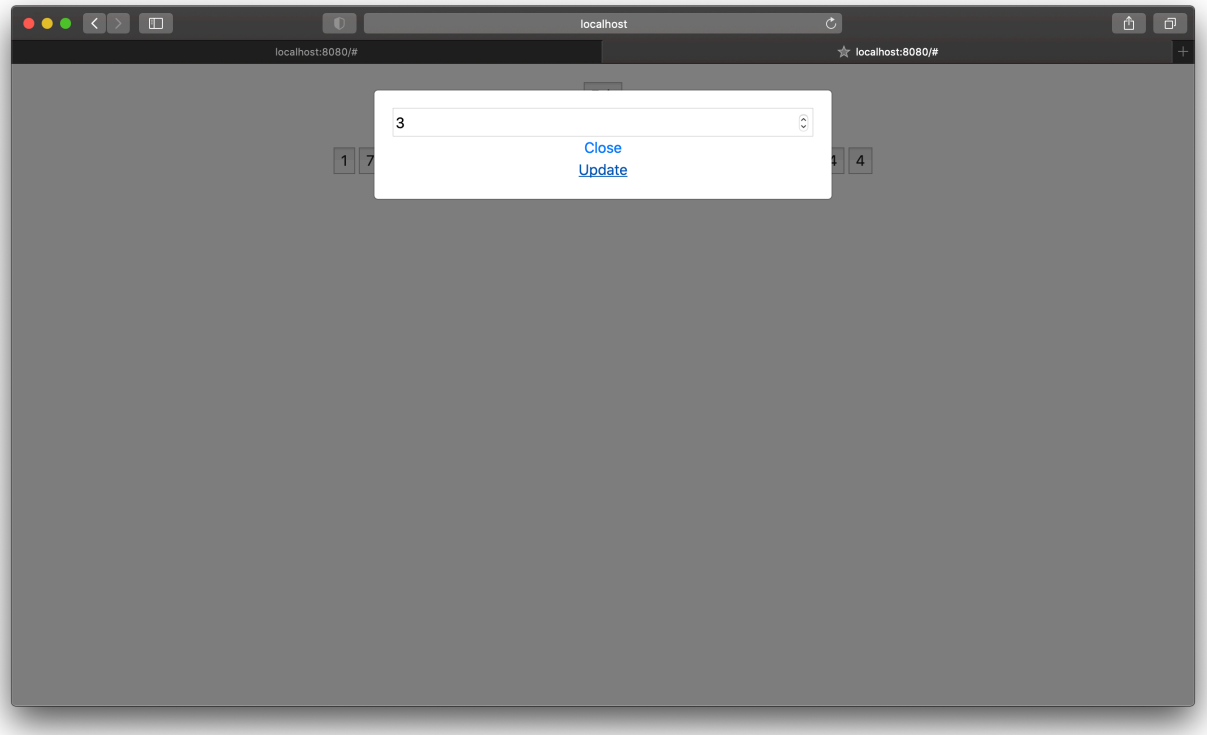
Рисунки с результатами работы программы




localhost
Ilya K.
Host game
Join existing


localhost
Exit
Your turn
1 7 5 7 3 7 9 7 3 5 3 9 9 9 1 6 5 9 4 4





Вывод

В данной лабораторной работе я освоил приемы разработки клиент-серверных веб-приложений на Java с использованием сокетов.