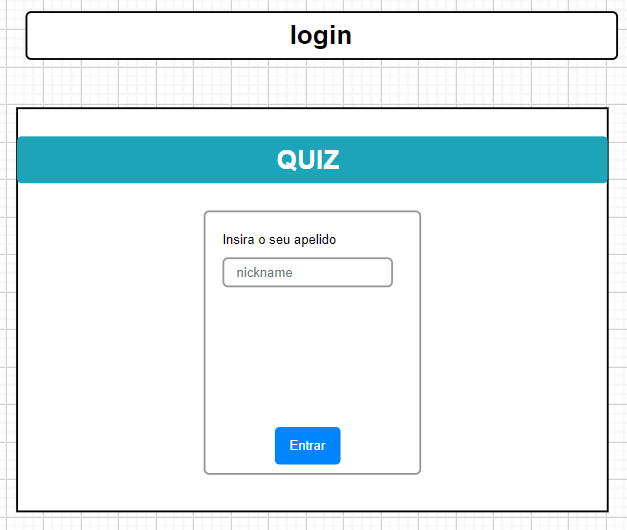
**CONTRATO DE COMUNICAÇÃO ENTRE FRONT E BACK**

[ENTRAR NA TELA INICIAL]

O front deverá iniciar conexão com o socket

(emitir evento) ‘new-visitor’;

io.to(socket.id).emit('new-visitor’,{});

(receber evento) ‘visitor-id’

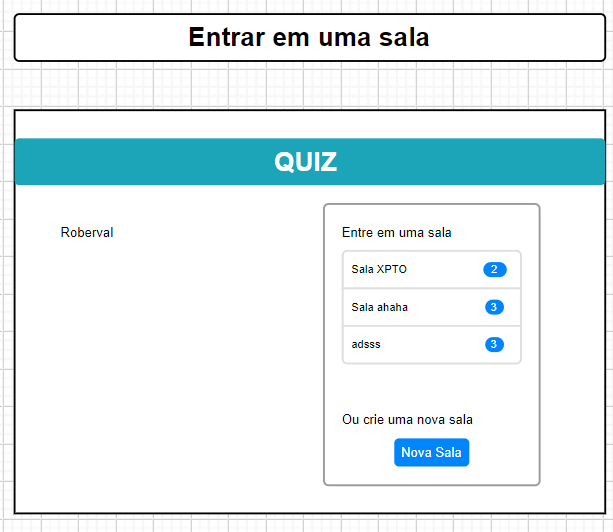
{playerId: 00000}

[BOTÃO ENTRAR]

(POST) “/create-player”

- Request: {“playerId”: 0, “nickname”:”string”}

- Response: statusCode 201



[ENTRAR NA TELA]

(GET) “/list-rooms”

- Response:

{ "rooms":[{

"roomId":0,

"roomName":"string",

"players":[{

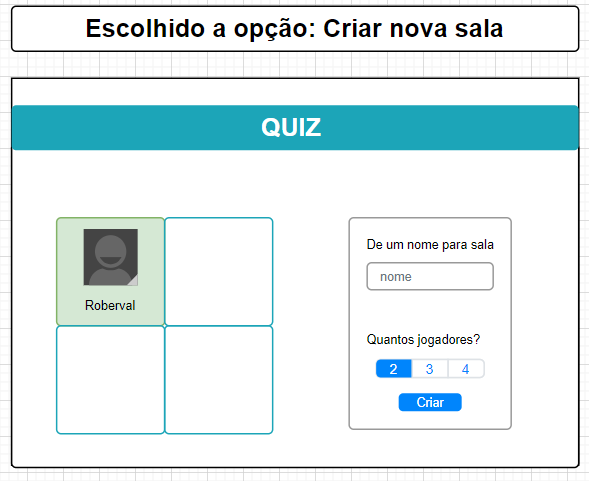
"playerId":2,

"playerName":"string"}],

"maxPlayers":4,

“socketUp“: false }]}

[BOTÃO CRIAR]

(POST) “/create-room”

- Request: {“roomName”:”string”, “playerId”:0, “maxPlayers”:2}

- Response:

{ "roomId":0,

"roomName":"string",

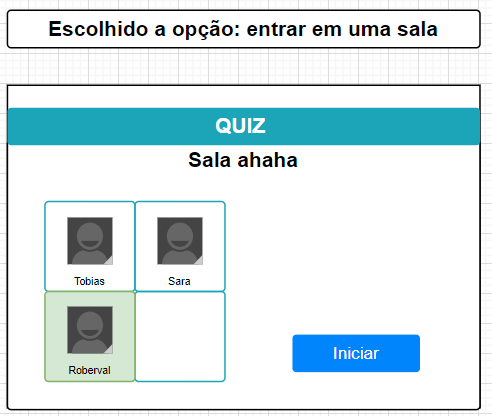
"players":[{

"playerId":2,

"playerName":"string"}],

"maxPlayers":4 }

(emitir evento) ‘create-room’ {roomId};



[CLICAR EM SALA DISPONÍVEL – 2ª tela]

(emitir evento) ‘join-room’ {roomId:00000}

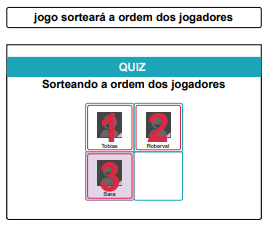
Caso algum usuário tente entrar em uma sala que já foi totalmente preenchida, será enviado o evento:

(receber evento) ‘ error’ {code: 409, message: “A sala atingiu a quantidade maxima de jogadores”}

[CLICAR NO BOTÃO INICIAR]

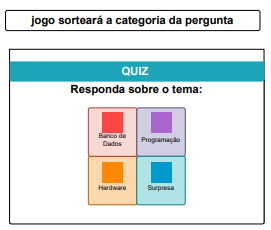
(emitir evento) ‘start’ {roomId:00000}

(receber evento) ‘questions-category’ {“questionsCtegory”:[{“categoryId”:1, “description”:”Hardware”}, {“categoryId”:2, “description”:”Programacao”}, {..}]}



Conforme o novo fluxo do diagrama, a ordem dos jogadores será por ordem de entrada na sala, por tanto nenhuma ação do back será necessária.

Essa informação já está contida no ultimo payload (propriedade: players[])

Ao clicar no botão para iniciar partida, o front recebeu o evento ‘question-categories’ com as categorias e nesse ponto, deve:

(emitir evento) ‘new-round’ {“roomId”:0, “categoryId”:0}

Esse evento deverá ser emitido a cada início de rodada, deste modo, o front deve esperar um dos seguintes eventos:

1. (receber evento) ‘game-over’ {*payload (****maquina de estado****) descrito no fim desse arquivo*}

socket.on(‘game-over', () => {

//chamar tela de ranking

});

1. (receber evento) ‘ok’ {*payload (****maquina de estado****) descrito no fim desse arquivo*}

*socket.on(‘ok’, () => {*

*io.to(socket.id).emit(‘new-question’, {“roomId:0, “categoryId”:0});*

*});*

Caso receba o evento ‘ok’, o front deve:

(emitir evento) ‘new-question’ {“roomId:0, “categoryId”:0}

(receber o evento) ‘new-question’ {*payload (****maquina de estado****) descrito no fim desse arquivo*}

Caso receba o evento ‘game-over’, significa que o jogo acabou e deve ser exibido o ranking dos jogadores.

[AO RENDERIZAR A TELA COM A PERGUNTA E DISPARAR O CRONOMETRO PARA APOSTA]

(Emitir evento) ‘bet-timer’ {“roomId”:0}

(Receber evento) ‘timer-running’ {timer:05)

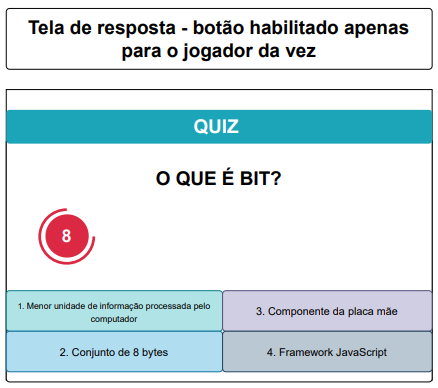
O back emitirá esse evento a cada segundo ate que o tempo para responder a pergunta se esgote

[CLICAR NO VALOR DA APOSTA OU AO ESGOTAR O TEMPO]

(Emitir evento) ‘bet’ {“roomId”:0, “bet”:2, “upVote”: true}

bet = valor da aposta

upVote = apostar que o jogador da vez irá acertar (retorna boolean) – caso seja o jogador da vez, sempre retornar true



[AO RENDERIZAR A TELA COM AS ALTERNATIVAS]

(Emitir evento) ‘question-timer’ {“roomId”:0}

(Receber evento) ‘timer-running’ {“timer”:10)

O back emitirá esse evento a cada segundo ate que o tempo para responder a pergunta se esgote

[CLICAR EM UMA ALTERNATIVA]

(Emitir evento) ‘answer’ {“roomId:0, “questionId’: 25, “answer”:1, “timer”:8}

Timer = tempo restante no momento em que o jogador clicou na resposta

(Receber evento) ‘round-results’ *{ Payload room completo - descrito no fim desse arquivo*}

Na propriedade ‘correctAnswer’, será retornado a alternativa correta, dessa forma o front poderá informar qual a alternativa correta e saberá se o jogador acertou ou errou.

[CASO O TEMPO SE ESGOTE E O JOGADOR NÃO CLIQUE EM UMA ALTERNATIVA]

(Emitir evento) ‘time-over {“roomId:0, “questionId’: 25}

(Receber evento) ‘round-results’ *{ Payload room completo - descrito no fim desse arquivo*}

Na propriedade ‘correctAnswer’, será retornado a alternativa correta, dessa forma o front poderá informar qual a alternativa correta e saberá se o jogador acertou ou errou.

Exemplo do Payload room completo:

{

"roomId":0,

"roomName":"kldjfdfdçlf33ldmld399re0jfpwok-",

"question":{

"questionId":123,

"category":"variada",

"question":"biru biru du biruleibi",

"a":"sim",

"b":"nao",

"c":"talvez",

"d":"com certeza"

},

"players":[

{

"playerId":2,

"playerName":"roberval",

"score":0,

"coins":50,

"bet":0,

"upVote":true,

"isPtresent":true

},

{

"playerId":3,

"playerName":"tobias",

"score":0,

"coins":50,

"bet":0,

"upVote":true,

"isPtresent":true

}

],

"maxPlayer":4,

"socketUp":true,

"stateMachine":{

"actingPlayerId":2,

"round":0,

"correctAnswer":"a",

"alreadyAsked":[

1,

2,

5,

],

"questionTimer":10,

}

}

Tabelas:

CATEGORY

id

description

QUESTIONS

Id

categoryId

description

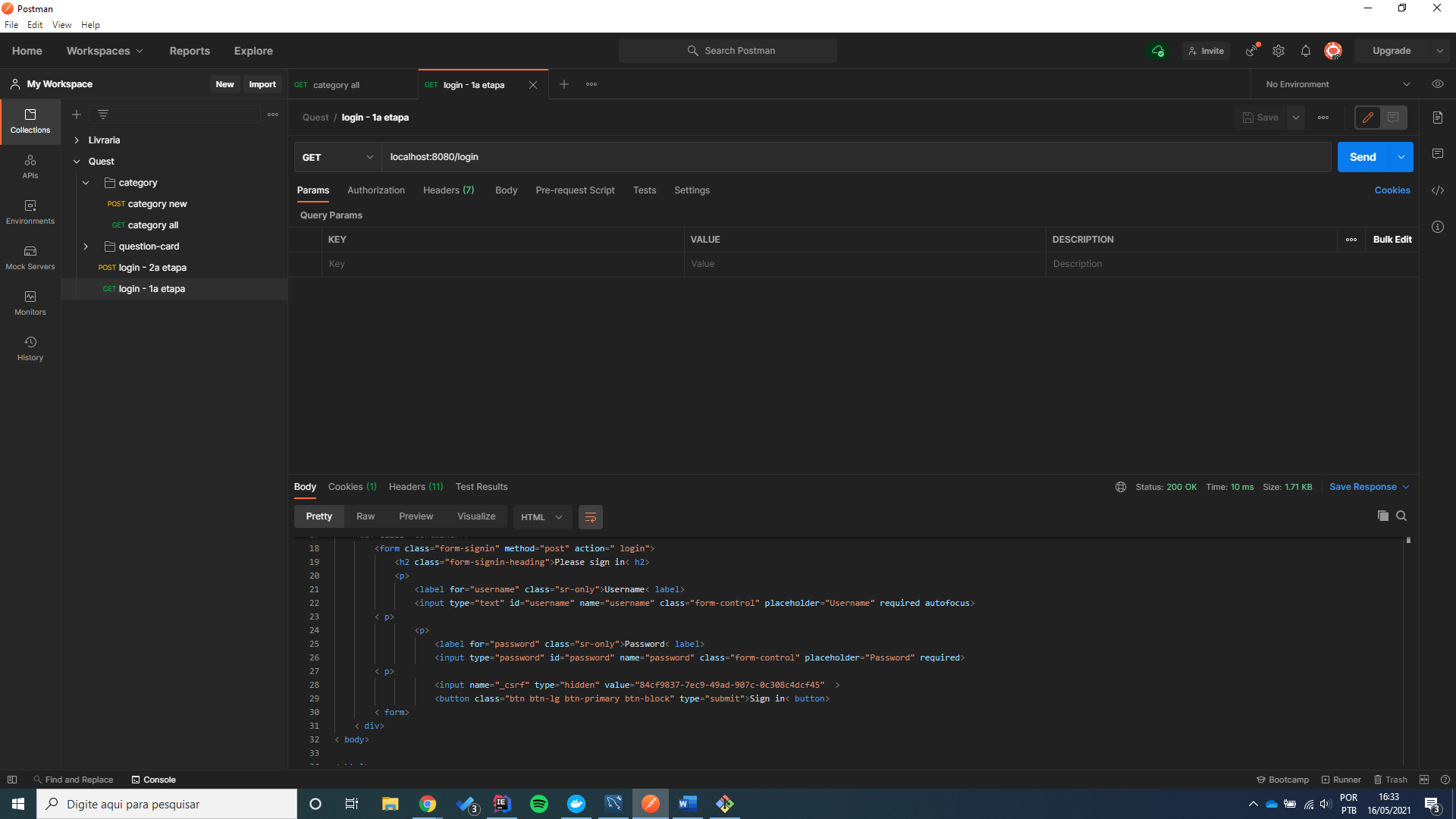
correctAnswer

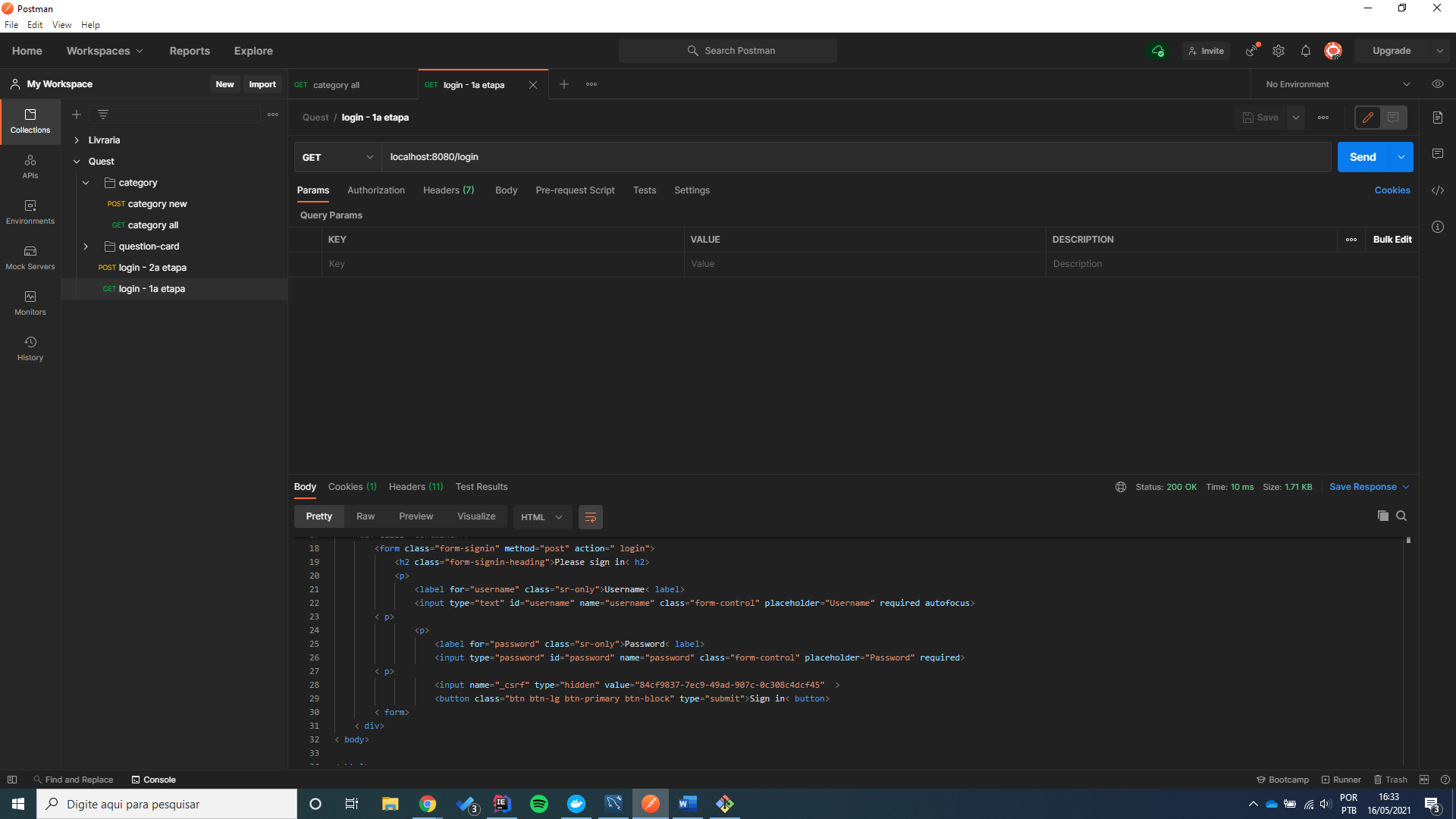
alternativeA

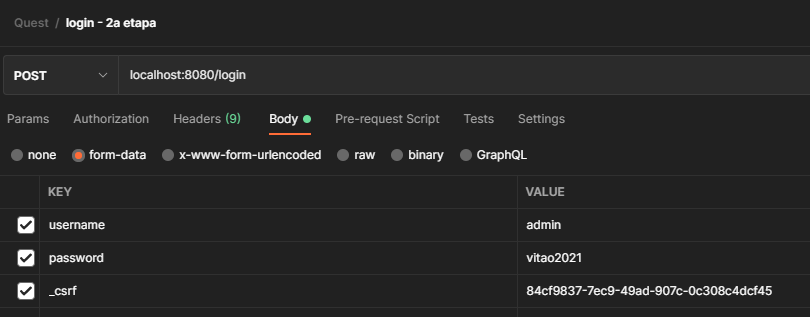
alternativeB

alternativeC

alternativeD







https://www.appsdeveloperblog.com/spring-security-default-username-password-role/#:~:text=Add%20Spring%20Security&text=The%20default%20username%20is%3A%20user,Spring%20Boot%20project%20is%20starting.