**1-As a student, I can record online an SQL evaluation, so that the trainer will correct it semi-automatically.**

@startuml

== run evaluation ==

participant Browser

participant JS

participant Server

participant DB

Browser -> Server: POST evaluation-{evalId}

Server -> DB: INSERT INTO sheet ...

Server -> DB: SELECT ... FROM sql\_question WHERE ...

Server -> Browser: list of questions

JS -> Server: PUT answer-from-{userId}-on-{questionId}-in-{evalId}

Server -> DB: execute the query in the related DB

Server -> DB: update table sql\_skills.sql\_answer with the result\n(requires a trigger on sheet to create the sql\_answer)

Server -> JS: 204 if all right

JS -> Browser: "ok"

**2- As a trainer, I can list my evaluations recently passed by the students, so that I can see or validate their notes.**

@startuml

== list evaluation ==

participant Browser

participant JS

participant Server

participant DB

Browser -> Server: GET all evaluation-{evalId}

Server -> DB: SELECT-{group\_id}-{scheduled\_at}-{ending\_at}-{completed\_at} FROM sql\_evalution WHERE-{trainer\_id}

Server -> Browser: list trainer evaluations

== mark completed ==

JS -> Server: PUT-{validated\_at}-from-evalution-{evalId}

Server -> DB: execute the query in the related DB

Server -> DB: required trigger to check all copies are corrected or not

Server -> JS: 204 ok.

== display list of question and notes for corrected copy ==

Browser -> Server: GET list of question

Server -> DB: SELECT {evalution\_id}-{question\_id}-{user\_id} FROM evalution-sql\_quiz\_question-trainee WHERE user\_id=trainee\_id

Server -> Browser: corrected copies witih all list of question

== display the average note ==

JS -> Server: GET average notes-{completed\_at}

Server -> DB: query to check coppies are completed or not

Server -> DB: SELECT completed\_at FROM evalution WHERE ...(required trigger to ckeck copies are corrected or not

Server -> JS: calculated average notes

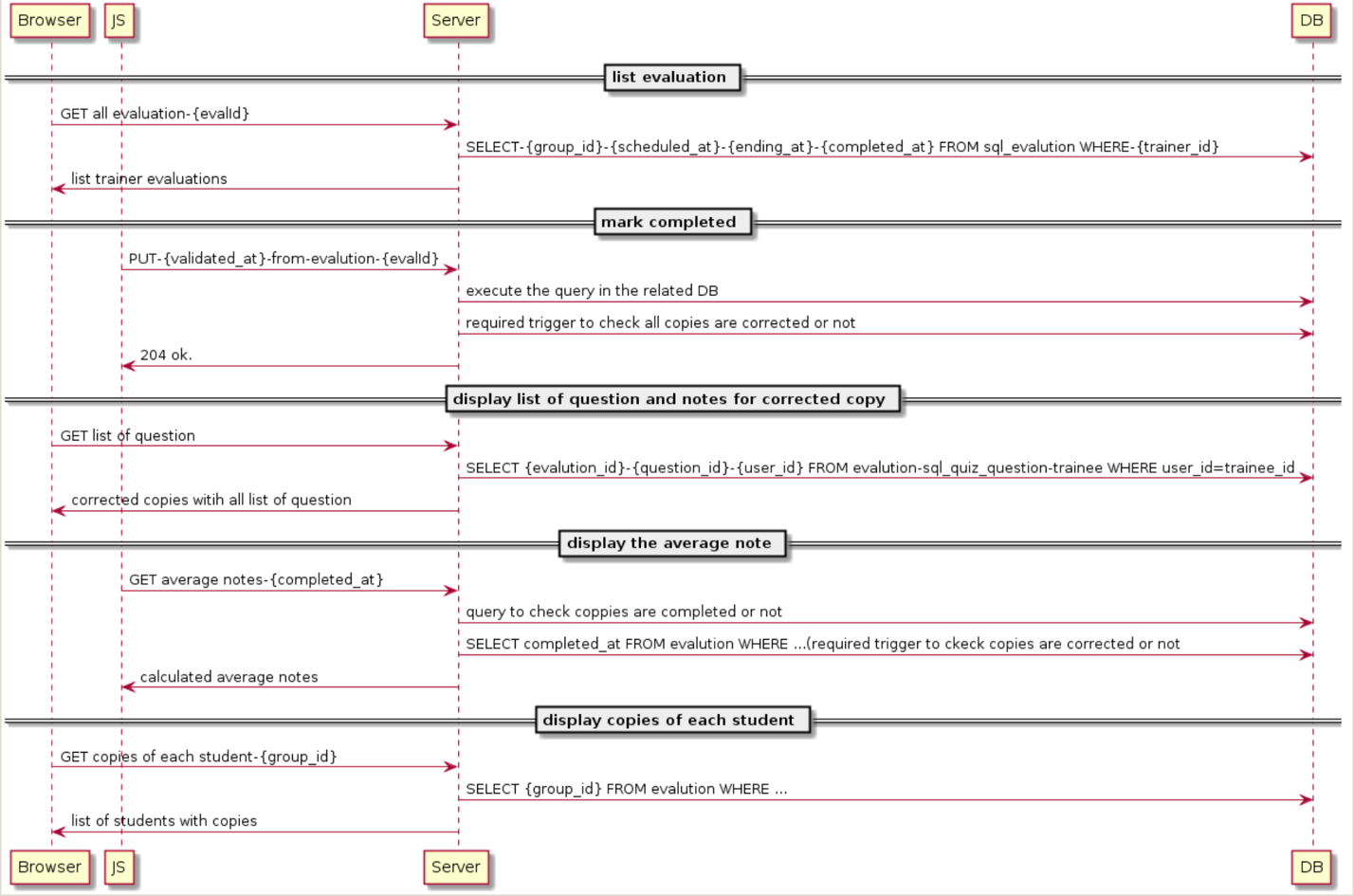
== display copies of each student ==

Brower -> Server: GET copies of each student-{group\_id}

Server -> DB: SELECT {group\_id} FROM evalution WHERE ...

Server -> Brower: list of students with copies

@enduml



**3. As a trainer, I can correct online a student evaluation, so that I am helped by the application.**

@startuml

== 3. As a trainer, I can correct online a student evaluation, so that I am helped by the application. ==

== 3.1 The complete set of questions is displayed ==

participant Browser

participant JS

participant Server

participant DB

Browser -> Server: GET all questions{question\_id}

Server -> DB: SELECT-{\*} FROM sql\_question WHERE-{trainer\_id}-{question\_id}

Server -> Browser: list trainer questions

== 3.4 If the class evaluation is not completed, I can validate or invalidate any question in the student copy, or I can delay its correction. ==

JS -> Server: GET copy of evaluation

Server -> DB: SELECT {\*} FROM {sql\_evaluation}...

DB -> Server: GET results of query

Server -> Browser: Display query results

JS -> Server: PUT-{validated\_at}-from-evalution-{evalId}

Server -> DB: execute the query in the related DB

Server -> DB: required trigger to check all copies are corrected or not

Server -> JS: 204 ok or transaction complete

== 3.5 When I validate or invalidate a question, or delay its correction, the change is immediately recorded (if I reload the page, its state remains the same). ==

JS -> Server: PUT a question status as {validate}-{invalidate}-{delay}

Server -> DB: UPDATE sql\_quiz\_question WHERE {quiz\_id}-{question\_id}; SELECT FROM {sql\_evaluation} WHERE {group\_id}-{trainer\_id}-{quiz\_id}..

DB -> Server: GET results of the query

Server-> Browser: display question status

Server -> JS: 204 or ok

== 3.6 The application shows if an answer is considered correct by the application and by myself ==

JS -> Server: GET correct answer

Server -> DB: SELECT answer FROM sql\_answer WHERE {question\_id}-{trainee\_id}-{evaluation\_id}

DB-> Server: return resutls of query

Server -> Browser: display correct answer

Server -> JS: 204 or ok

== 3.8 The application displays the total student note, according to the correctness rule above ==

JS -> Server: GET total student note according to the correctness

Server -> DB: SELECT SUM(give\_correct\_result) FROM sql\_answer WHERE {trainee\_id}+{evaluation}

DB-> Server: return results to server

Server -> JS: calculated total notes

JS -> Browser: Display total notes computed

== 3.9 The student copy is marked as completed when and only when I have completed all of its questions ==

JS -> Server: PUT-{validated\_at}-from-evalution-{evalId}

Server -> DB: execute the query in the related DB

Server -> DB: execute the required trigger to check if all copies are corrected or not

Server -> JS: return 204 if ok else the necessary error code

== 3.10 A click on a question displays the page correcting this question, all students included (see next feature) ==

Broswer -> Server: GET copies of questions for each student-{group\_id}{evaluation\_id}{question\_id}{trainee\_id}

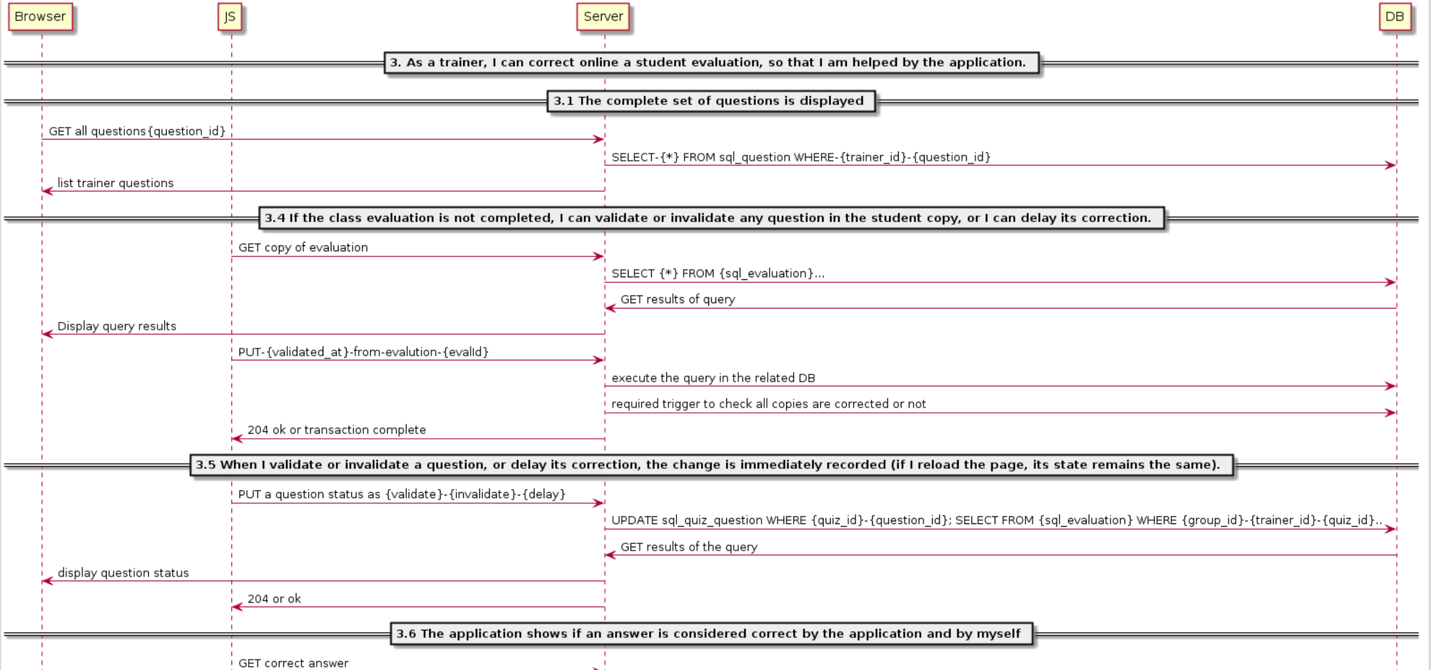
Server -> DB: SELECT {group\_id}+{evaluation\_id}+{question\_id}+{trainee\_id} FROM evalution WHERE ...

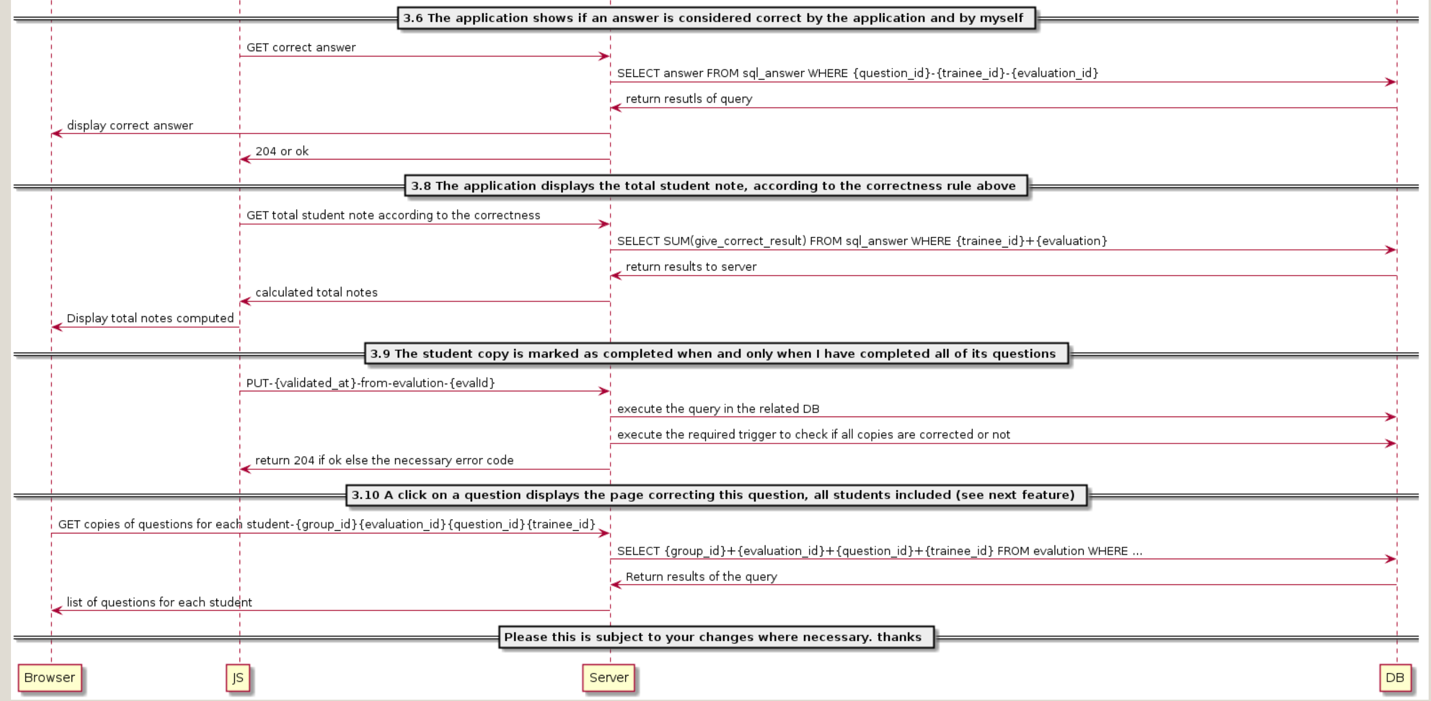
DB -> Server: Return results of the query

Server -> Browser: list of questions for each student

== Please this is subject to your changes where necessary. thanks ==

@enduml





**4-As a trainer, I can correct online a question, all students included, in order to be equitable for all students and concentrated on the question.**

@startuml

== sign up ==

participant Browser

participant JS

participant Server

participant DB

== All the Questions ===

Browser -> Server: GET all the questions-{ques\_id}

Server -> DB: SELECT-{ques\_id}-{ques\_name}-{ques\_type} from question

DB -> Server : Fetches the list of questions

Server -> Browser: Displays the list of questions

== Answers for the Question ===

Browser -> Server: GET all the answers-{ques\_id}

Server -> DB: SELECT-{ques\_id}-{valid\_answer} from answers

DB -> Server : Fetches the list of valid Answers

Server -> JS: Displays list of valid answers for the question

== Sheet and student mapping ==

Browser -> Server: GET sheets for the student-{stud\_id}

Server -> DB: SELECT-{sheet\_id}-{stud\_id}-{result} from sheet where stud\_id={stud\_id}

DB -> Server : Fetches the sheet with name answer and result.

Server -> Browser: Displays the sheet with name answer and result.

== Validate or Invalidate ==

JS -> Server: PUT sheet {sheet\_id}-{question\_id}-{valid\_flag=SET/RESET}

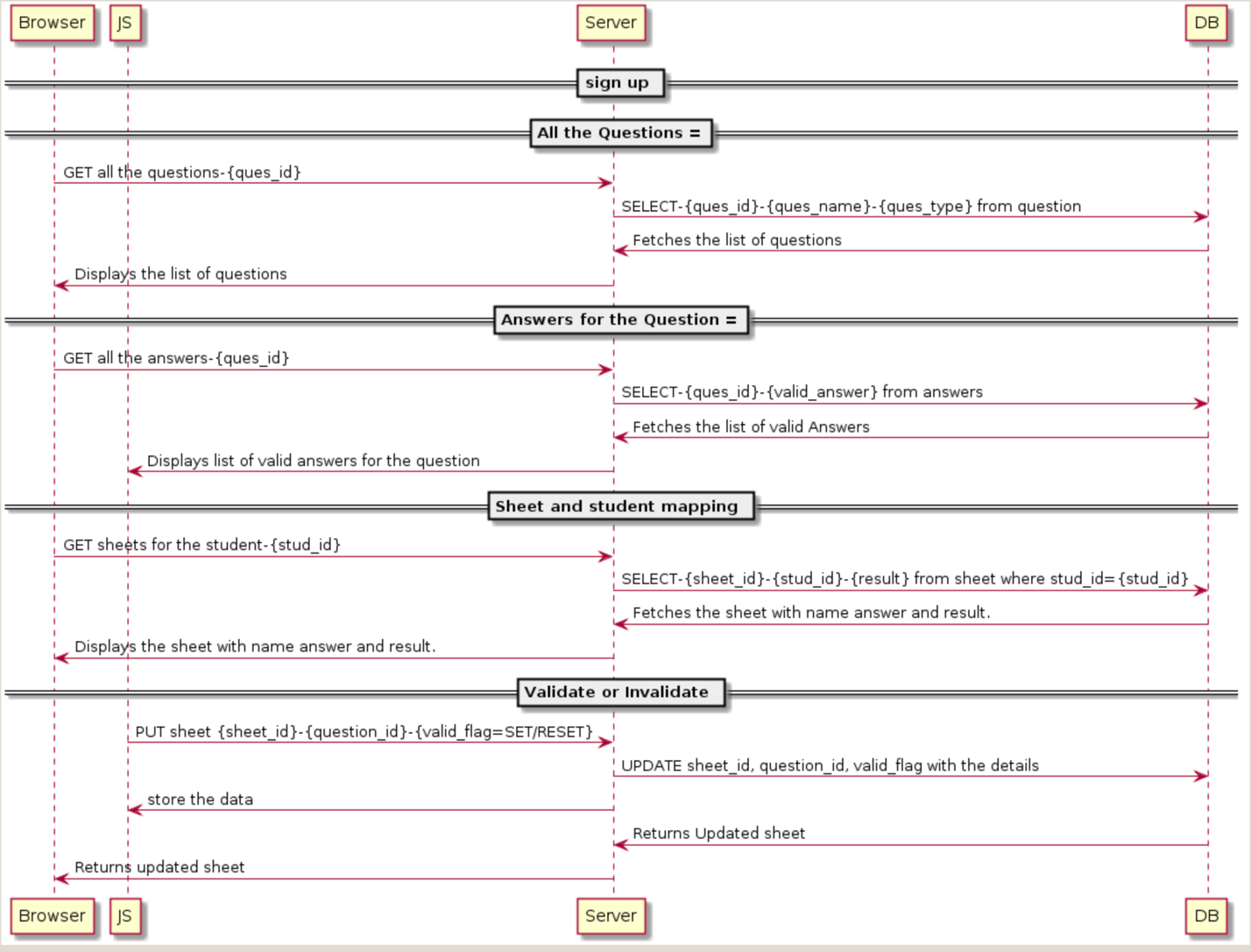
Server -> DB: UPDATE sheet\_id, question\_id, valid\_flag with the details

Server -> JS: store the data

DB -> Server: Returns Updated sheet

Server -> Browser: Returns updated sheet

@enduml



**5- As a student, I can ask to join a class, so that I can access its evaluations.**

@startuml

== sign up ==

participant Browser

participant JS

participant Server

participant DB

Browser -> server: PUT /signup

Server -> DB: INSERT INTO STUDENTS

Server -> JS: 200 ok

== student login ==

Browser -> server: POST /login

Server -> DB: SELECT \* FROM STUDENTS WHERE ID=? AND PASSWORD=?

Server -> JS: 200 ok

JS -> Browser: give access

== request to join a class ==

Browser -> Server: PUT /joining

Server -> DB: INSERT INTO joiningrequests a student id

Server -> Browser: 203 Created request

== send an e-mail to the student with token ==

Server -> DB: SELECT token, student\_id,student\_email FROM joiningrequests,student tables WHERE ...

DB -> Server: Token and other information

Server -> Browser: send student a email with token

== student accepts the request ==

Browser -> Server: POST /acceptingrequest

Server -> DB: update table tokenAcceptance

Server -> DB: check the variable TRUE/FALSE

Server -> JS: 200/400

JS -> Browser: "ok"

== student is able to check his classes ==

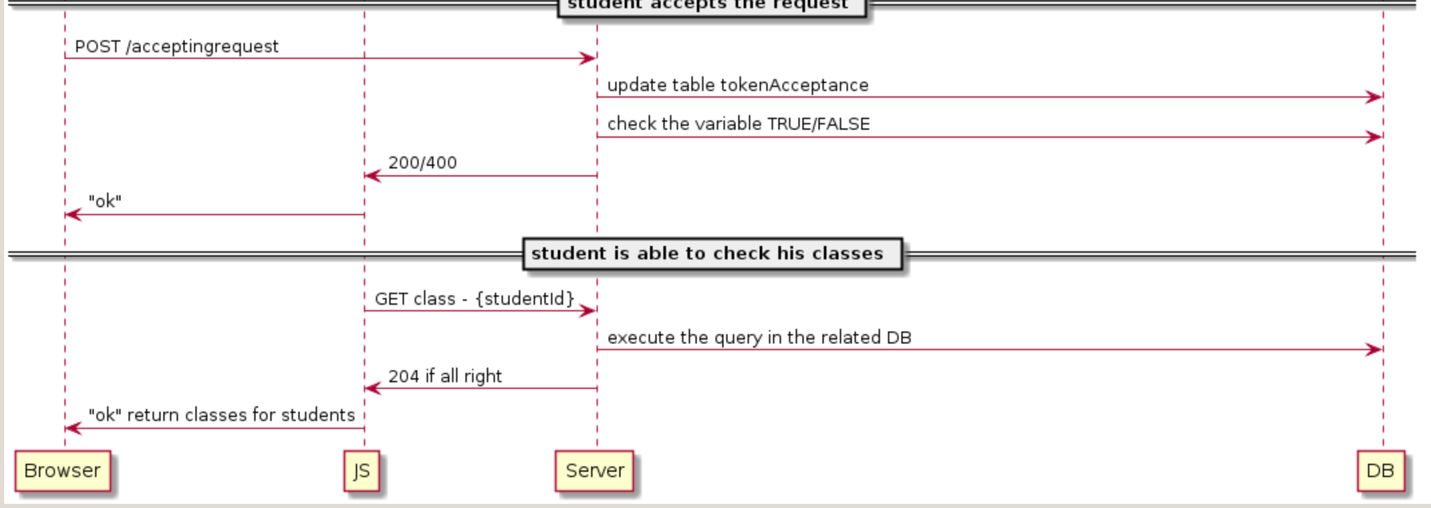
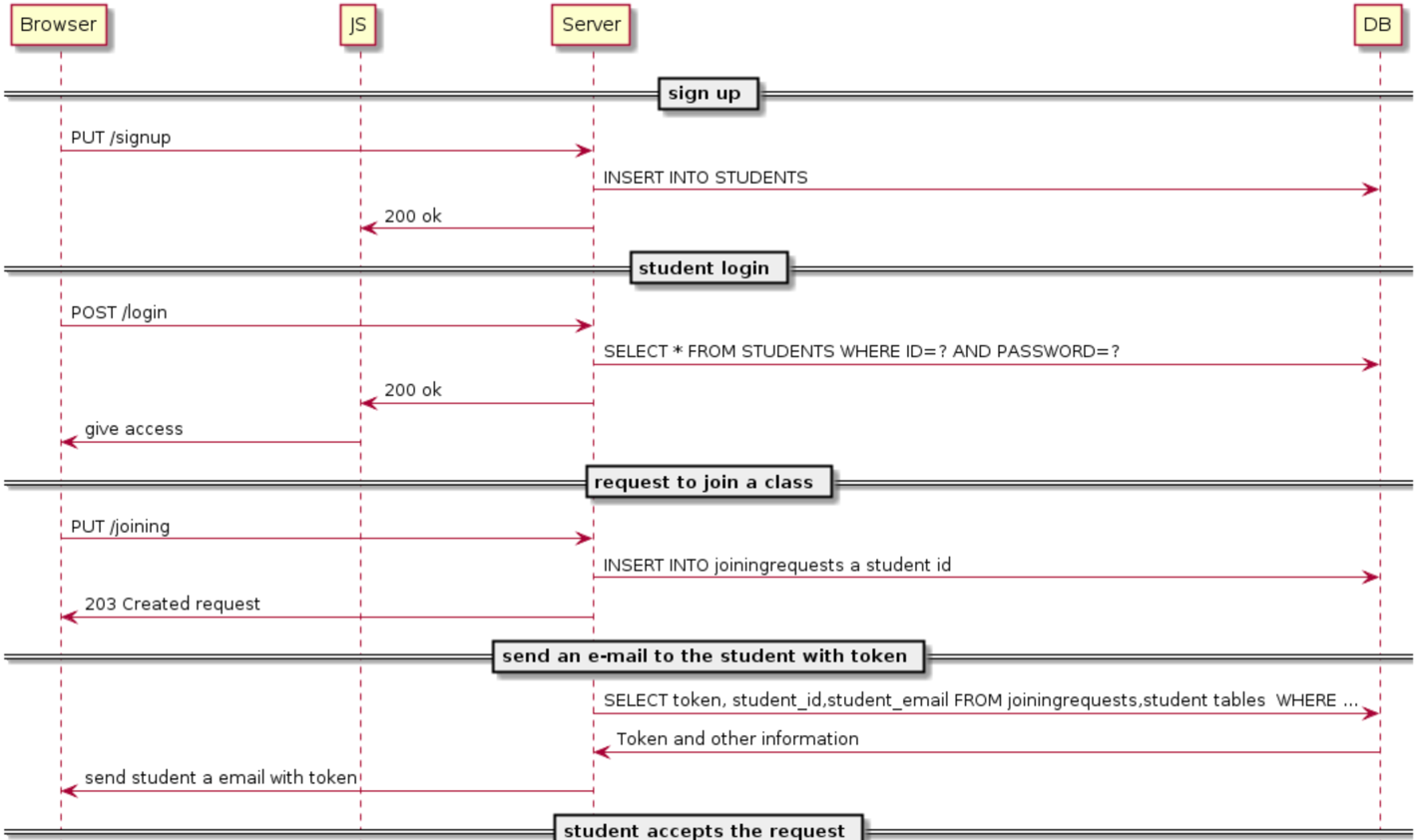
JS -> Server: GET class - {studentId}

Server -> DB: execute the query in the related DB

Server -> JS: 204 if all right

JS -> Browser: "ok" return classes for students

@enduml



**6- . As a trainer, I can validate for a class the requests to be a member of it, so that they are controlled.**

@startuml

== Get list of validated members ==

participant Browser

participant JS

participant Server

participant DB

Browser -> Server: GET list of validated members-{memberid as user\_id}

Server -> DB: SELECT {user\_id}-{first\_name}-{last\_name}-{email}-{validated\_at} FROM user WHERE ...

DB -> Server: Fetches the list of members

Server -> Browser: Displays list of members

== validated, invalidated and pending requests have an associated style ==

JS -> Server: GET colours at user-{user\_id}

Server -> DB: execute SELECT \* FROM user WHERE ...

DB -> Server: show all members validated, invalidated, or pending(requires trigger to check status)

Server -> JS: list of members with colours

== validate or invalidate or postpone request to join group ==

Browser -> Server: POST user-{user\_id}

Server -> DB: SELECT {user-id}-{validation\_at} FROM user WHERE ...

DB -> Browser: Action performed

Server -> Browser: Action has been completed

== Validation recorded ==

JS -> Server: PUT user-{validation\_at}

Server -> DB: UPDATE user {user\_id}-{validation\_at} with system time ...

DB -> Browser: {validation\_at} show time

Server -> Browser: Shows the validated time

== close class ==

JS -> Server: DELETE user

Server -> DB: execute query to check class is open or not

DB -> Server: Returns time status

Server -> JS: Display message relatively

@enduml

