**Quiz – Manager (PHP Application):**

**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 View

Participant Controller

Participant Model

Participant DBMS

Browser -> Controller: GET evaluation-{id}

Controller -> Model: Evaluation::get($id)

Model -> DBMS: SELECT ... FROM evaluation JOIN sql\_quiz JOIN quiz\_db

DBMS -> Model: rows

Controller -> View: evaluation\_view.php must send the trainee\_id (taken in the session)

View -> Browser: diagram, title, scheduled\_at, ending\_at + start button

== start ==

Browser -> Controller: POST sheet-by-{traineeId}-on{evalId}

Controller -> Model: Sheet::update($traineeId, $evalId)

Model -> DBMS: UPDATE sheet ...

Controller -> Model: ...getQuestions($quizId)

Controller -> View: sheet-student.php

View -> Browser: list of questions with form

== answer ==

JS -> Controller: PUT answer-by-{traineeId}-on-{questionId}-for-{evalId}

Controller -> Model: SqlAnswer::update($traineeId, $questionId, $evalId, $answer)

Model -> DBMS: UPDATE sql\_answer SET ... WHERE ...

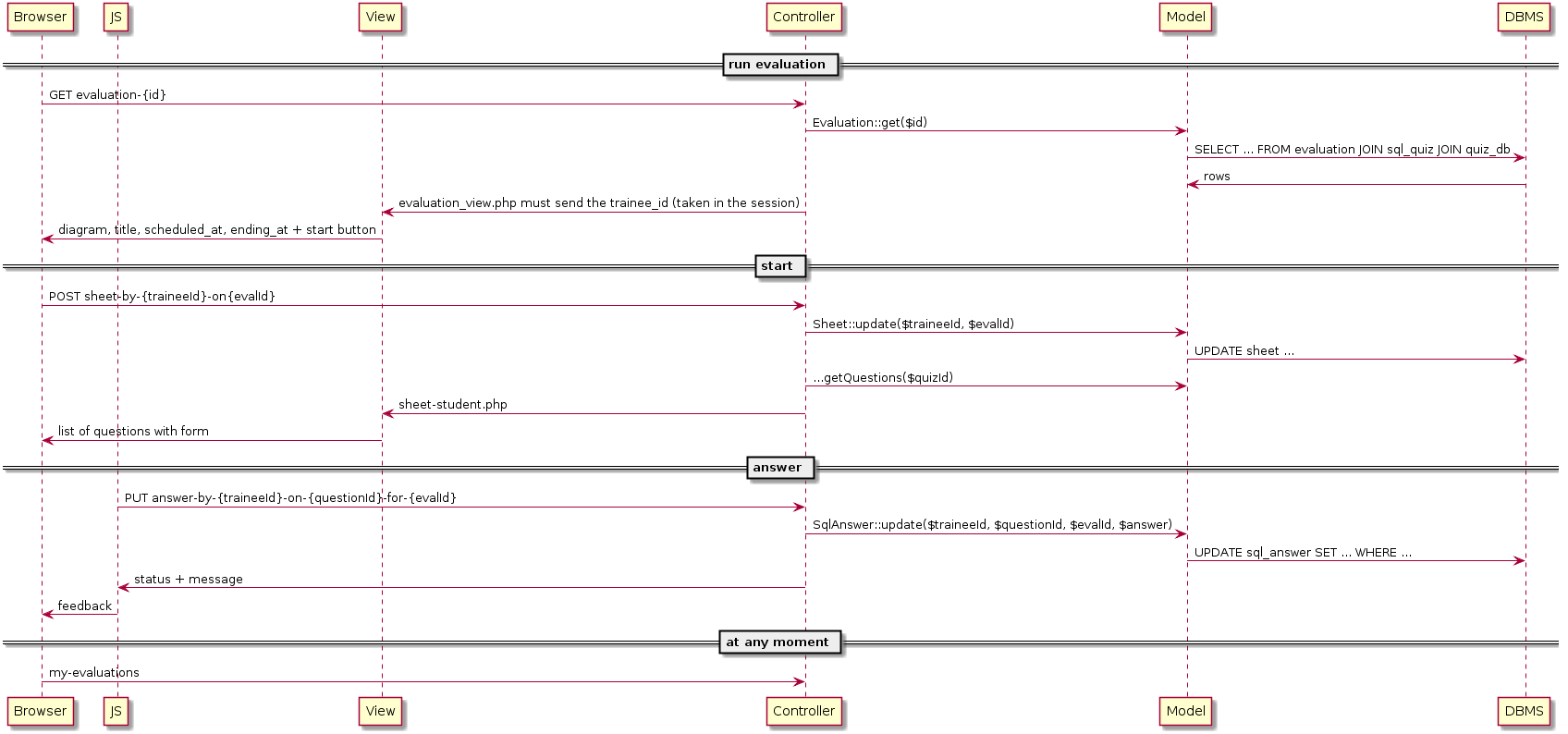
Controller -> JS: status + message

JS -> Browser: feedback

== at any moment ==

Browser -> Controller: my-evaluations

@enduml



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

participant Browser  
participant JS  
participant View  
participant Controller  
participant Model  
participant DBMS

View -> Controller: GET all evaluation-{evalId}  
Model -> DBMS: SELECT-{group\_id}-{scheduled\_at}-{ending\_at}-{completed\_at} FROM sql\_evalution WHERE-{trainer\_id}  
Model -> View: list trainer evaluations

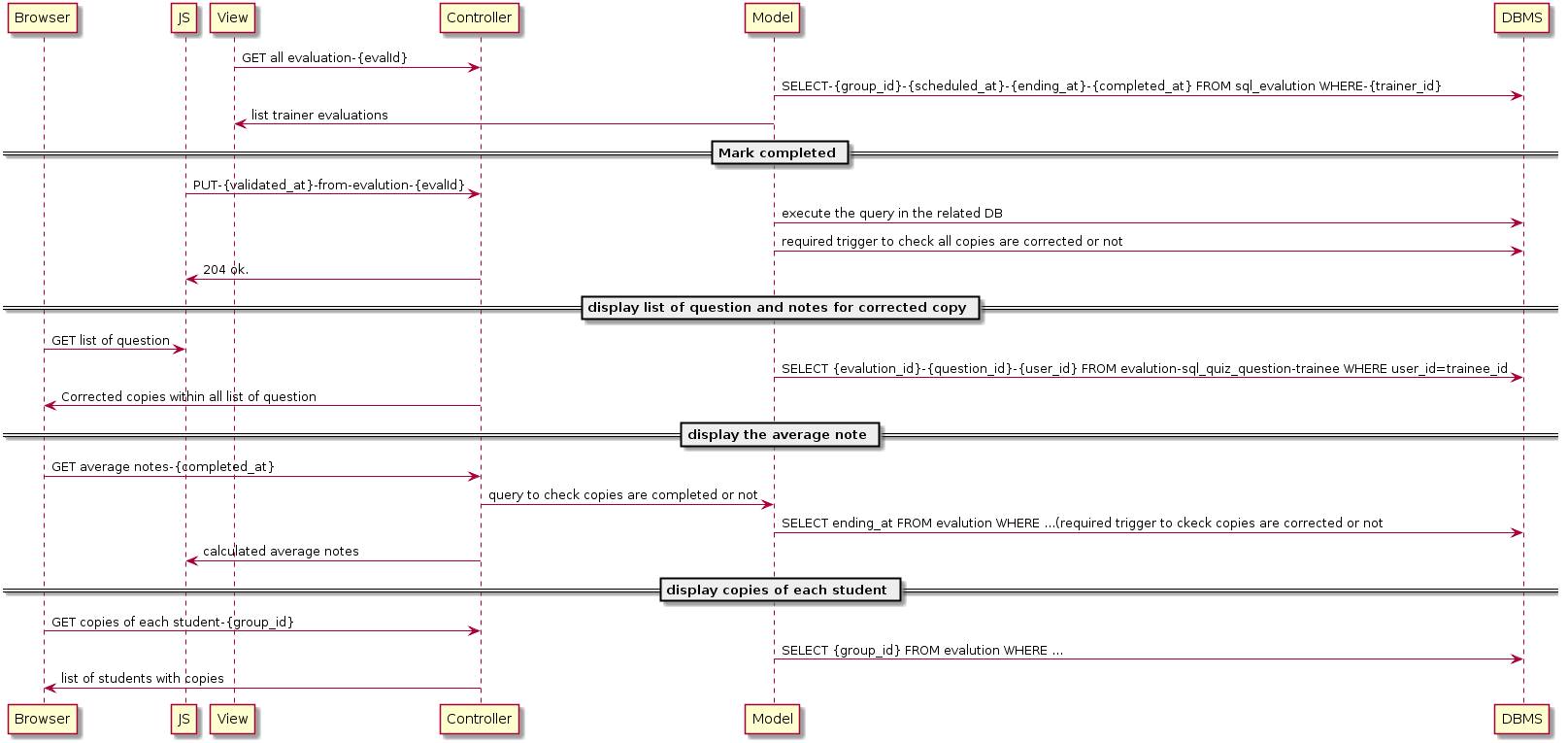
== Mark completed ==  
JS -> Controller: PUT-{validated\_at}-from-evalution-{evalId}  
Model -> DBMS: execute the query in the related DB  
Model -> DBMS: required trigger to check all copies are corrected or not  
Controller -> JS: 204 ok.

== display list of question and notes for corrected copy ==  
Browser -> JS: GET list of question  
Model -> DBMS: SELECT {evalution\_id}-{question\_id}-{user\_id} FROM evalution-sql\_quiz\_question-trainee WHERE user\_id=trainee\_id  
Controller -> Browser: Corrected copies within all list of question

== display the average note ==  
Browser -> Controller: GET average notes-{completed\_at}  
Controller -> Model: query to check copies are completed or not  
Model -> DBMS: SELECT ending\_at FROM evalution WHERE ...(required trigger to ckeck copies are corrected or not  
Controller -> JS: calculated average notes

== display copies of each student ==  
Browser -> Controller: GET copies of each student-{group\_id}  
Model -> DBMS: SELECT {group\_id} FROM evalution WHERE ...  
Controller -> Browser: 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.**

== The complete set of questions is displayed ==

participant Browser  
participant JS  
participant View  
participant Controller  
participant Model  
participant DBMS

View -> Controller: GET all questions{question\_id}

Model -> DBMS: SELECT-{\*} FROM sql\_question WHERE-{trainer\_id}-{question\_id}

Controller -> Browser: list trainer questions

== 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 -> Controller: GET copy of evaluation

Model -> DBMS: SELECT {\*} FROM {evaluation}...

DBMS -> Controller: GET results of query

Controller -> View: Display query results

JS -> Controller: PUT-{corrected\_at}-from-evalution-{evalId}

Model -> DBMS: execute the query in the related DB

Model -> DBMS: required trigger to check all copies are corrected or not

Controller -> JS: 204 ok or transaction complete

== 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 -> Controller: PUT a question status as {validate}-{invalidate}-{delay}

Model -> DBMS: UPDATE sql\_quiz\_question WHERE {quiz\_id}-{question\_id}; SELECT FROM { evaluation} WHERE {group\_id}-{trainer\_id}-{quiz\_id}..

DBMS -> Model: GET results of the query

Controller -> Browser: display question status

Controller -> JS: 204 or ok

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

JS -> Controller: GET correct answer

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

DBMS -> Controller: Return results of the query

Controller -> Browser: Displaying the correct answer

Controller -> JS: 204 or ok

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

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

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

DBMS -> Controller: return results to server

Controller -> JS: Calculated the Total Notes

JS -> Browser: Displaying the total notes computed

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

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

Model -> DBMS: Execute the query in the related DB

Model -> DBMS: Execute the required trigger to check if all copies are corrected or not

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

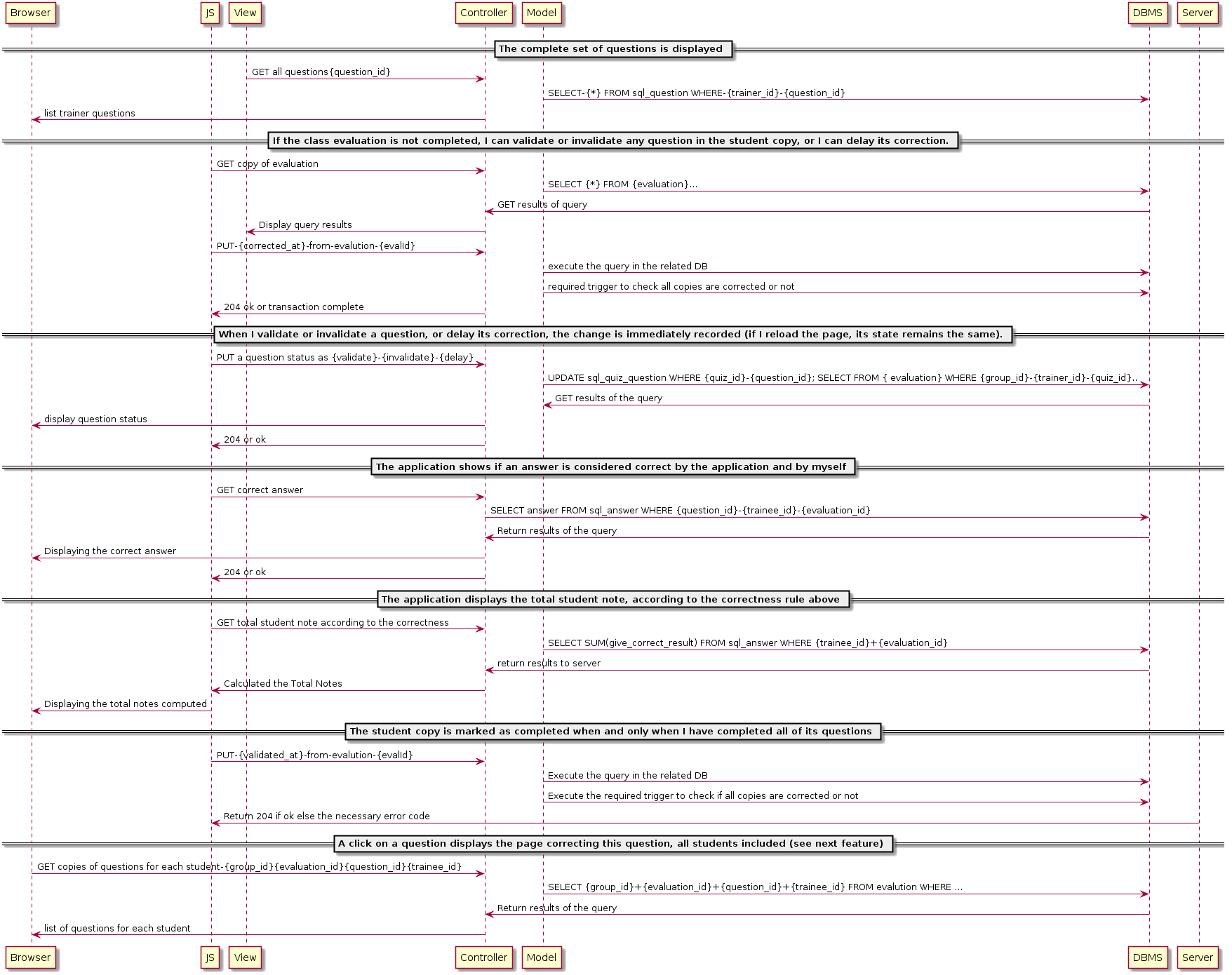
== A click on a question displays the page correcting this question, all students included (see next feature) ==

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

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

DBMS -> Controller: Return results of the query

Controller -> Browser: list of questions for each student



**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

participant Browser

participant JS

participant View

participant Controller

participant Model

participant DBMS

== All the Questions ==

View -> Controller: GET all the questions-{question\_id}

Model -> DBMS: SELECT-{question\_id}-{question\_name} from question

DBMS -> Model : Fetches the list of questions

Controller -> View: Displays the list of questions

== Answers for the Question ==

View -> Controller: GET all the answers-{question\_id}

Model -> DBMS: SELECT-{question\_id}-{valid\_answer} from answers

DBMS -> Model : Fetches the list of valid Answers

Controller -> View: Displays the list of valid answers for the question

== Sheet and student mapping ==

View -> Controller: GET sheets for the student-{user\_id}

Model -> DBMS: SELECT-{sheet\_id}-{user\_id}-{result} from sheet where user\_id={user\_id}

DBMS -> Model : Fetches the sheet with answer and results.

Controller -> View: Displays the sheet with answer and results.

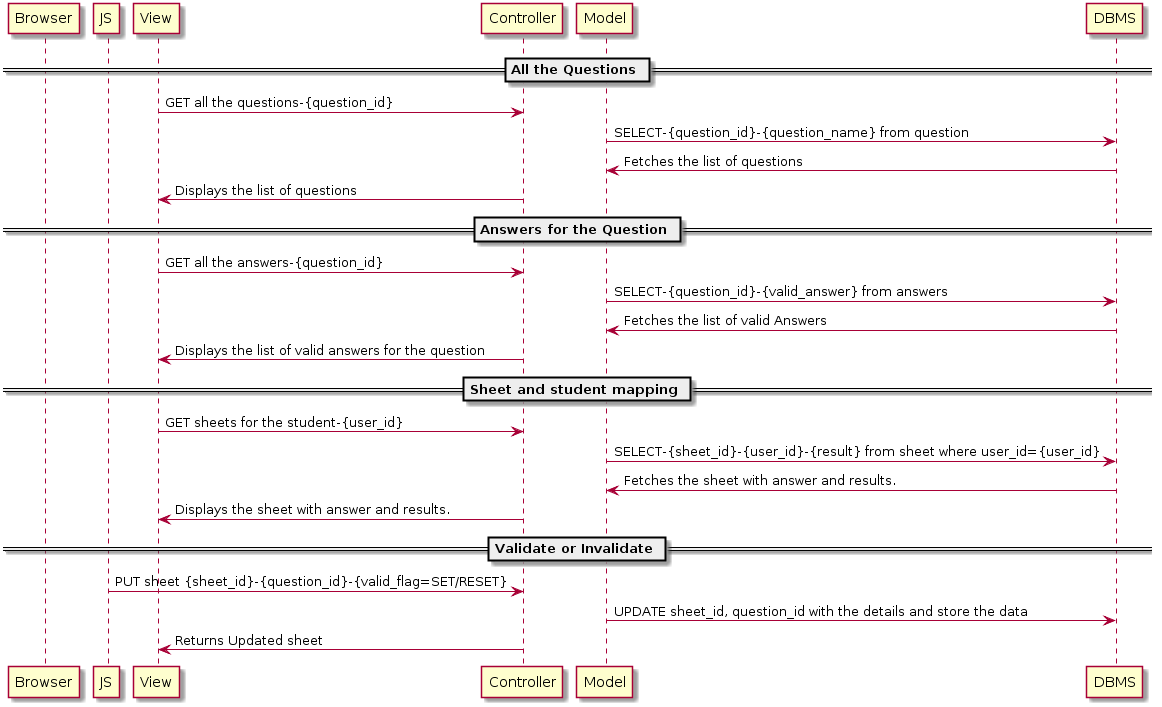
== Validate or Invalidate ==

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

Model -> DBMS: UPDATE sheet\_id, question\_id with the details and store the data

Controller -> View: Returns Updated sheet

@enduml



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

@startuml

participant Browser

participant JS

participant View

participant Controller

participant Model

participant DBMS

== Signup Users ==

Browser -> Controller: PUT / Signup

Controller -> Model: User Details (Email, Password, First Name, Last Name)

Model -> DBMS: INSERT into user

Controller -> JS: 200 ok

== Student login ==

Browser -> Controller: POST /login

Controller -> JS: 200 ok

Controller -> DBMS: SELECT \* FROM user WHERE email=? AND pwd=?

JS -> View: give access

== Requesting to join a class ==

Browser -> Controller: PUT / joining

Model -> DBMS: INSERT INTO group\_member a user\_id

Controller -> Browser: 203 Created request

== Student accepts the request ==

Browser -> Controller: POST / Accepting request

Model -> DBMS: update the user table

Controller -> DBMS: check the variable TRUE/FALSE

Controller -> JS: 200/400

JS -> Browser: "ok"

== Student is able to check his classes ==

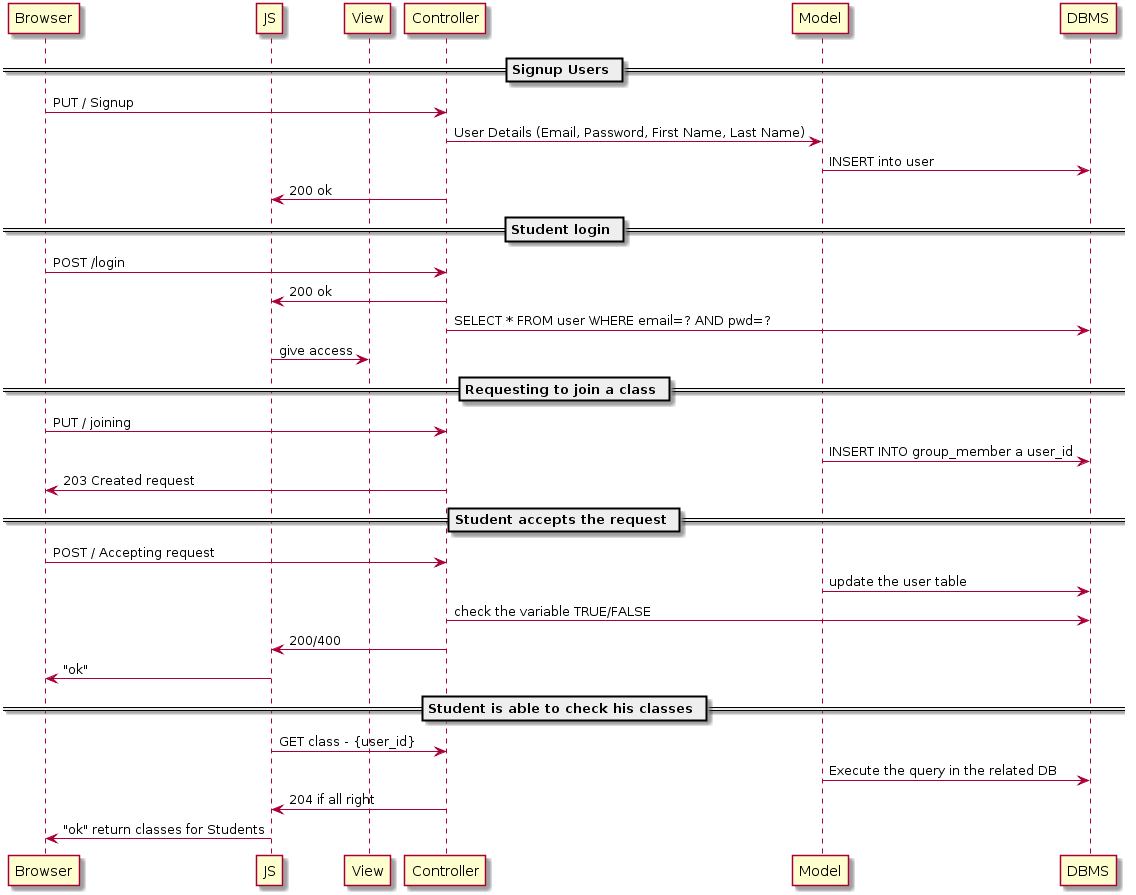
JS -> Controller: GET class - {user\_id}

Model -> DBMS: Execute the query in the related DB

Controller -> 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 View

participant Controller

participant Model

participant DBMS

View -> Controller: GET list of validated members-{user\_id}

Model -> DBMS: SELECT {user\_id}-{first\_name}-{name}-{email}-{validated\_at} FROM user WHERE ...

DBMS -> Model: Fetches the list of members

Controller -> View: Displays list of members

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

JS -> Controller: GET Status (Validation) at user-{user\_id}

Model -> DBMS: execute SELECT \* FROM user WHERE ...

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

Controller -> JS: list of members with Status (Validation)

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

View -> Controller: POST user-{user\_id}

Model -> DBMS: SELECT {user-id}-{validation\_at} FROM user WHERE ...

== Validation recorded ==

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

Model -> DBMS: UPDATE user {user\_id}-{validation\_at} with system time ...

Model -> Controller: Retrieve the {validation\_at} show time

Controller -> Browser: Showing the validated time

== Close class ==

JS -> Controller: Deleting the User

Model -> DBMS: Execute query to check class is open or not

Controller -> JS: Display message relatively

@enduml

