**Service Layer**

**03.21.2024**

**Mary Jines**

**SWDV 691 Capstone**

**Maryville University**

**Overview**

The service layer of the Brazilian Jiu Jitsu Journal will be designed to ensure efficient communication between the user interface and the PostgreSQL database. The service layer will be implemented using Django’s Models. The following three layers will compose the backend of the application:

1. URL Dispatcher (Routes): Django utilizes “URL Dispatcher” to define the possible endpoints of the application and directs HTTP request to the right views based on the URL.
2. Views: Django uses “Views”. The routes mentioned above point to the correct views to handle the HTTP requests also mentioned above. Responses are also handled by the views and contain the business logic for interacting with the database.
3. Models: Finally, models will define the structure of the database tables and map to and interact with the database tables.

\*Notes: I am using a placeholder URL for now. Also, in some of the requests you’ll see the Cookie header with a session ID. Realistically, the session ID in cookie form is handled by the browser, but I included it for my own clarity.

**Service Endpoints**

**User Registration**

Method: POST

URL: https://brazilianjiujitsujournal.com/api/auth/register

Purpose: This is the service endpoint for when a new user goes to create a new account through the “Create Account” selection on the navbar.

Sample Request:

POST /api/auth/register HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

{

"username": "newUser",

"email": "newUser@example.com",

"password": "password123"

}

Success Response:

HTTP/1.1 201 Created

Content-Type: application/json

{

"message": "Account created successfully"

}

Error Response:

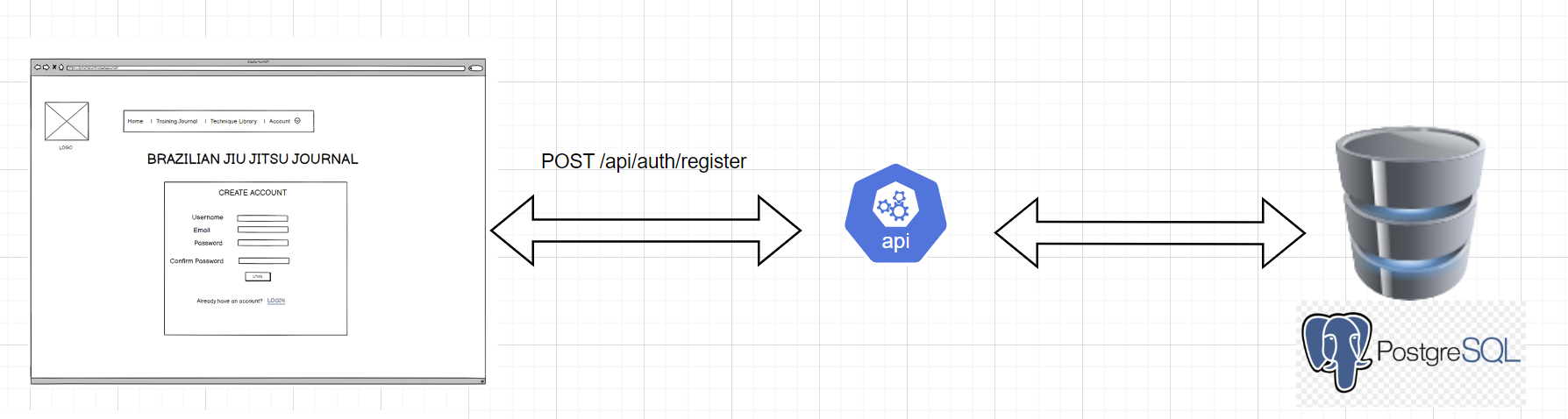
HTTP/1.1 400 Bad Request

Content-Type: application/json

{

"error": "Email already exists"

}



**User Login**

Method: POST

URL: https://brazilianjiujitsujournal.com/api/auth/login

Purpose: This will authenticate the user when they log in, and return a token or a session id depending on if I choose to use the default Django session IDs or token-based authentication.

Sample Request:

POST /api/auth/login HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

{

"username": "user@example.com",

"password": "password123"

}

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

Set-Cookie: sessionid=abc123; HttpOnly; Path=/

{

"message": "Login successful"

}

Error Response:

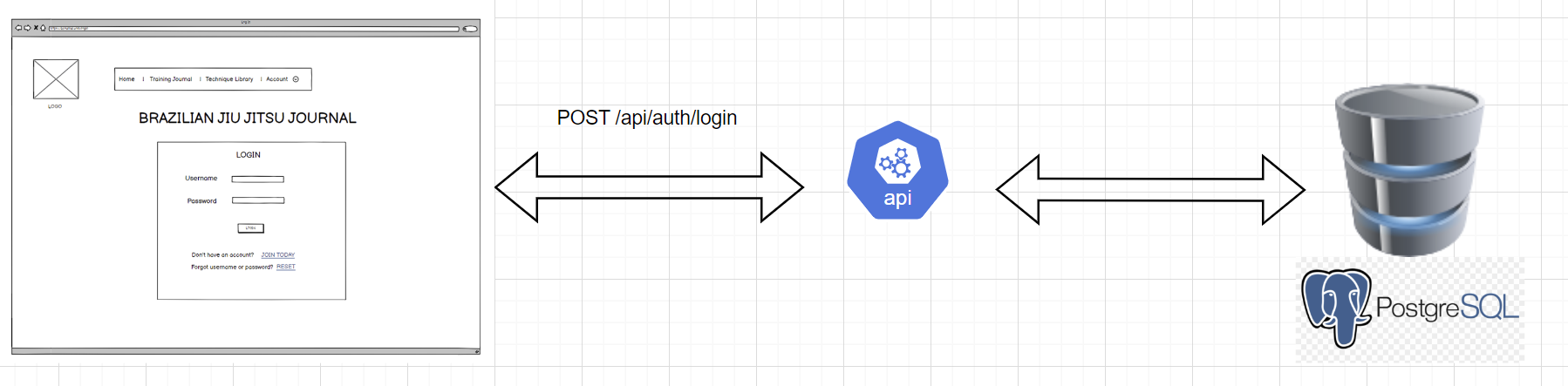
HTTP/1.1 401 Unauthorized

Content-Type: application/json

{

"error": "Invalid username or password"

}



**Training Journal**

Method: GET

URL: https://brazilianjiujitsujournal.com/api/journal/entries

Purpose: This will retrieve all of the journal entries for a particular user to display on the user’s Training Journal page.

Sample Request:

GET /api/journal/journal HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"entries": [

{

"date": "2024-03-20",

"duration": "2 hours",

"notes": "Practiced arm bars",

"techniques": ["Arm Bar"]

},

...

]

}

Error Response:

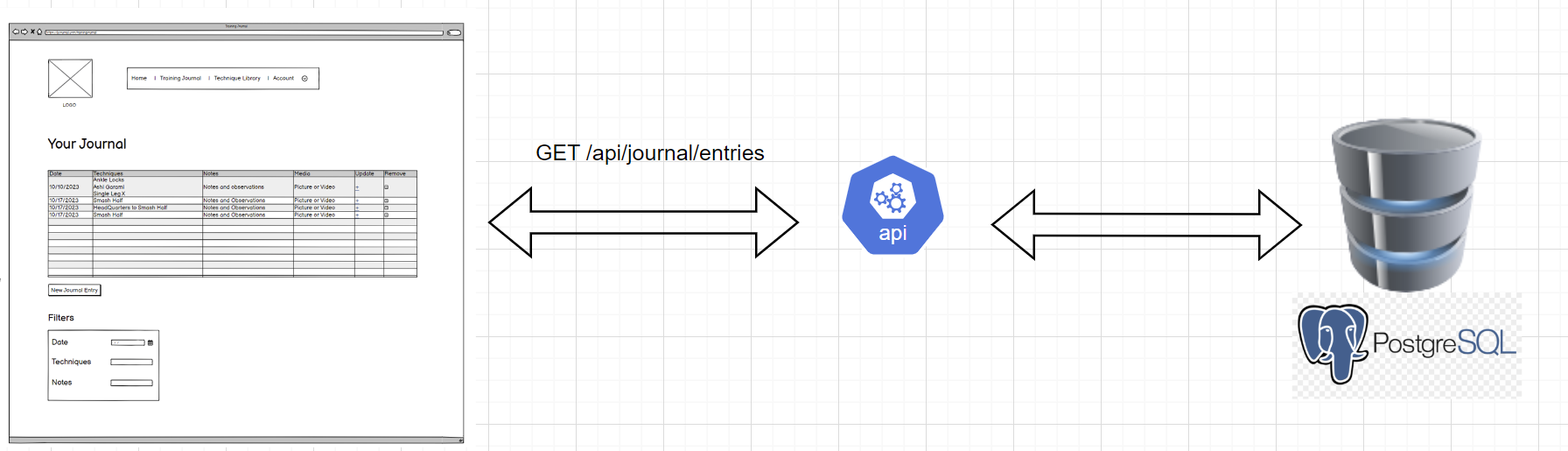
HTTP/1.1 403 Forbidden

Content-Type: application/json

{

"error": "Unauthorized access"

}



Method: POST

URL: https://brazilianjiujitsujournal.com/api/journal/entry

Purpose: When a user clicks “New Entry” on their training journal page, this is the endpoint that will be reached that will handle the creation of a new journal entry.

Sample Request:

POST /api/journal/entry HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

Cookie: sessionid=abc123

{

"date": "2024-03-22",

"duration": "2 hours",

"notes": "Focused on leg locks",

"techniques": ["Heel Hook", "Kneebar"]

}

Success Response:

HTTP/1.1 201 Created

Content-Type: application/json

{

"message": "Entry added successfully"

}

Error Response:

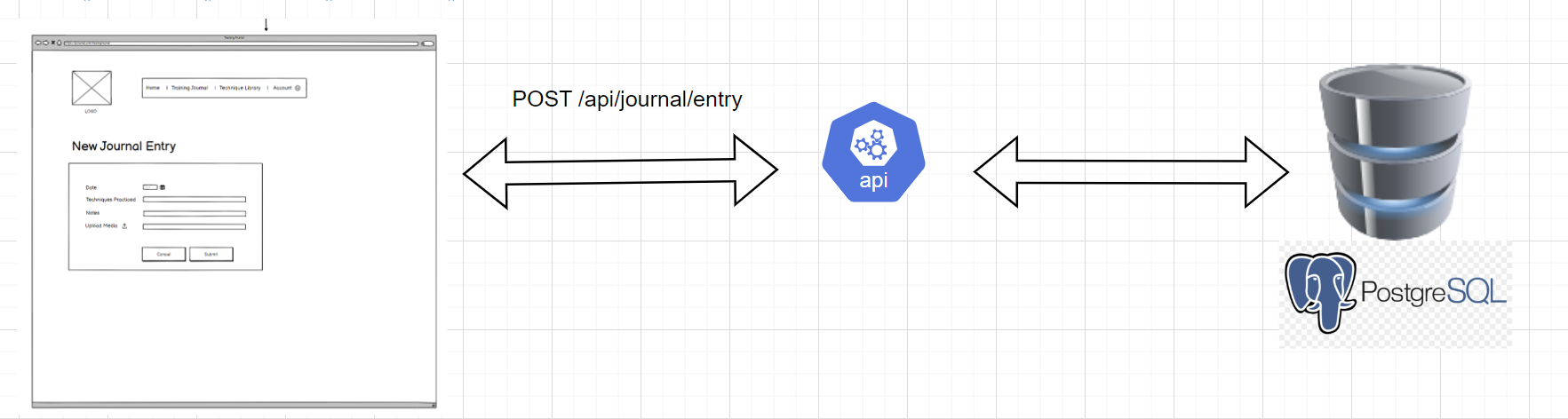
HTTP/1.1 400 Bad Request

Content-Type: application/json

{

"error": "Missing required fields"

}



Method: PUT

URL: <https://brazilianjiujitsujournal.com/api/journal/entries/update>

Purpose: If a user selects the update option on a journal entry, this is the endpoint that will handle updating that particular entry.

Sample Request:

PUT /api/journal/update/{session\_id} HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

Cookie: sessionid=abc123

{

"date": "2024-03-23",

"duration": "1.5 hours",

"notes": "Improved guard passing",

"techniques": ["Guard Pass", "Side Control"]

}

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"message": "Entry updated successfully"

}

Error Response:

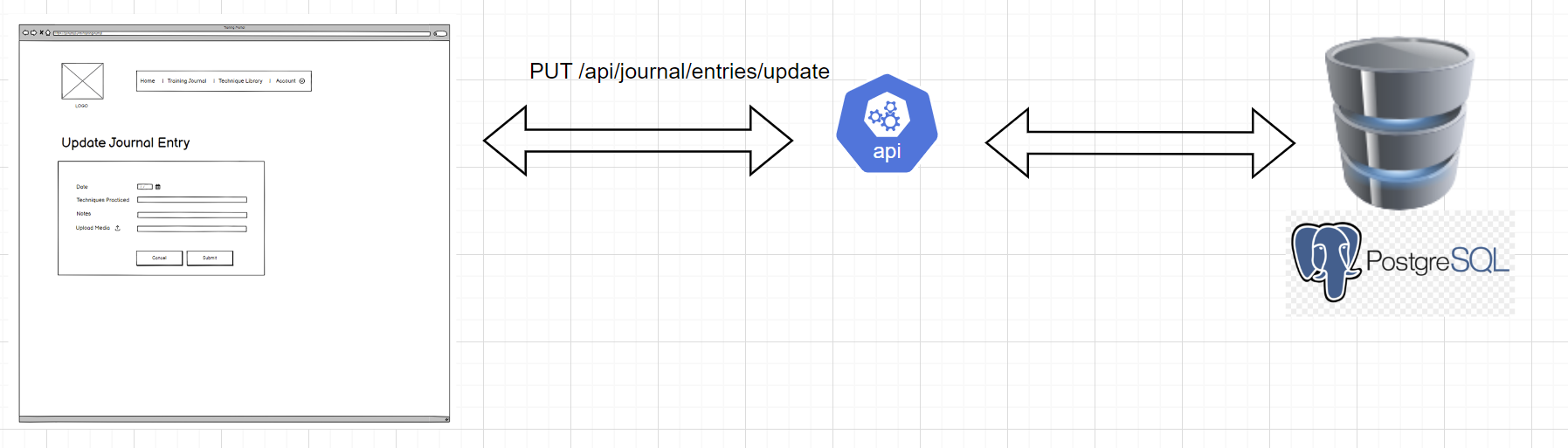
HTTP/1.1 404 Not Found

Content-Type: application/json

{

"error": "Entry not found or unauthorized"

}



Method: GET

URL: https://brazilianjiujitsujournal.com/api/journal/search?query=[searchterm]

Purpose: This endpoint serves to handle searches a user might perform on the entries in their Training Journal.

Sample Request:

GET /api/journal/search?query=armbar HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"entries": [

{

"date": "2024-03-22",

"duration": "2 hours",

"notes": "Focused on mount",

"techniques": ["armbar", "Head and arm choke"]

},

...

]

}

Error Response:

HTTP/1.1 200 OK

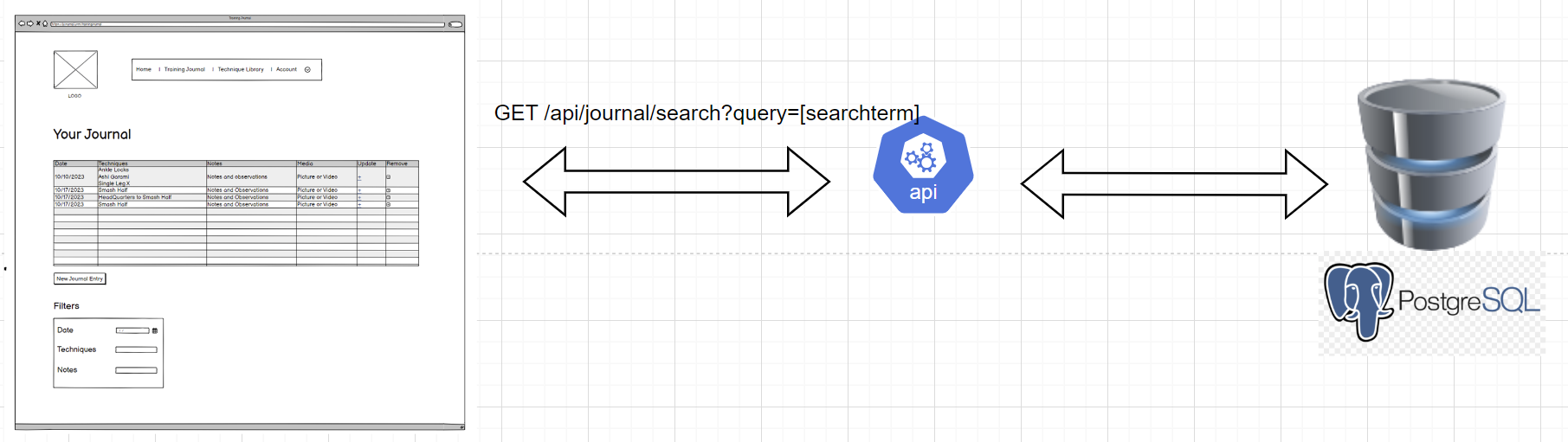
Content-Type: application/json

{

"message": "No journal entries found matching the criteria",

"results": [] // Empty results indicating no matches found

}



Method: DELETE

URL: https://brazilianjiujitsujournal.com/api/journal/entries/{entry\_id}

Purpose: This will handle the deletion of specific journal entries.

Sample Request:

DELETE /api/journal/entry/{entry\_id} HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"message": "Entry deleted successfully"

}

Error Response:

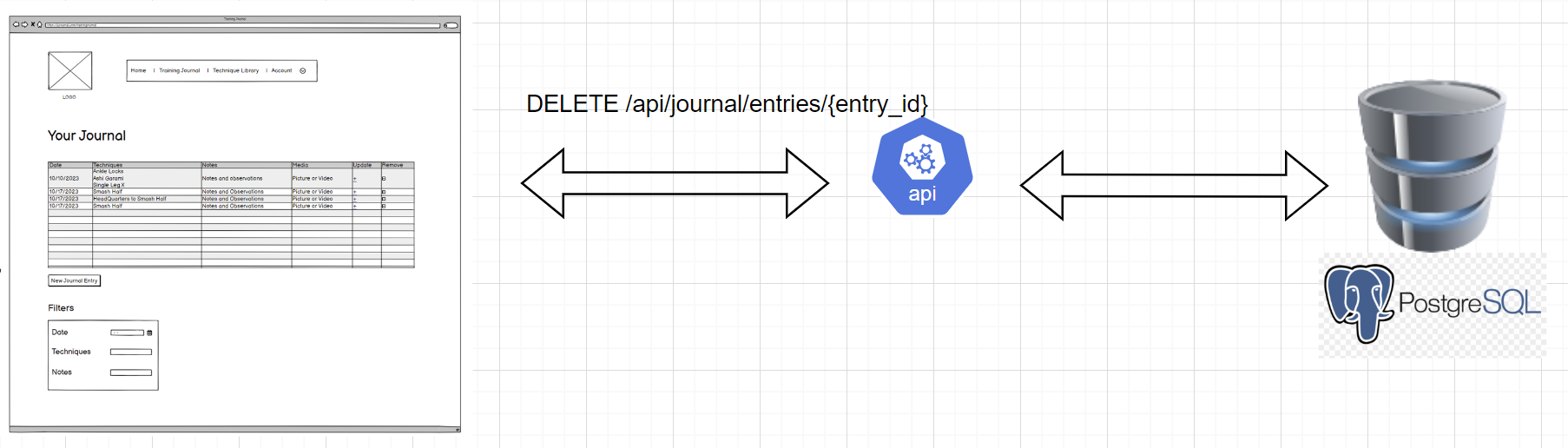
HTTP/1.1 404 Not Found

Content-Type: application/json

{

"error": "Entry not found or unauthorized"

}



**Technique Library**

Method: GET

URL: https://brazilianjiujitsujournal.com/api/technique/library

Purpose: This will retrieve all of the techniques for a particular user to display on the user’s Technique Library page.

Sample Request:

GET /api/technique/library HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"techniques": [

{

"name": "Arm Bar",

"type": "Submission",

"description": "An armlock that hyperextends the elbow"

},

...

]

}

Error Response:

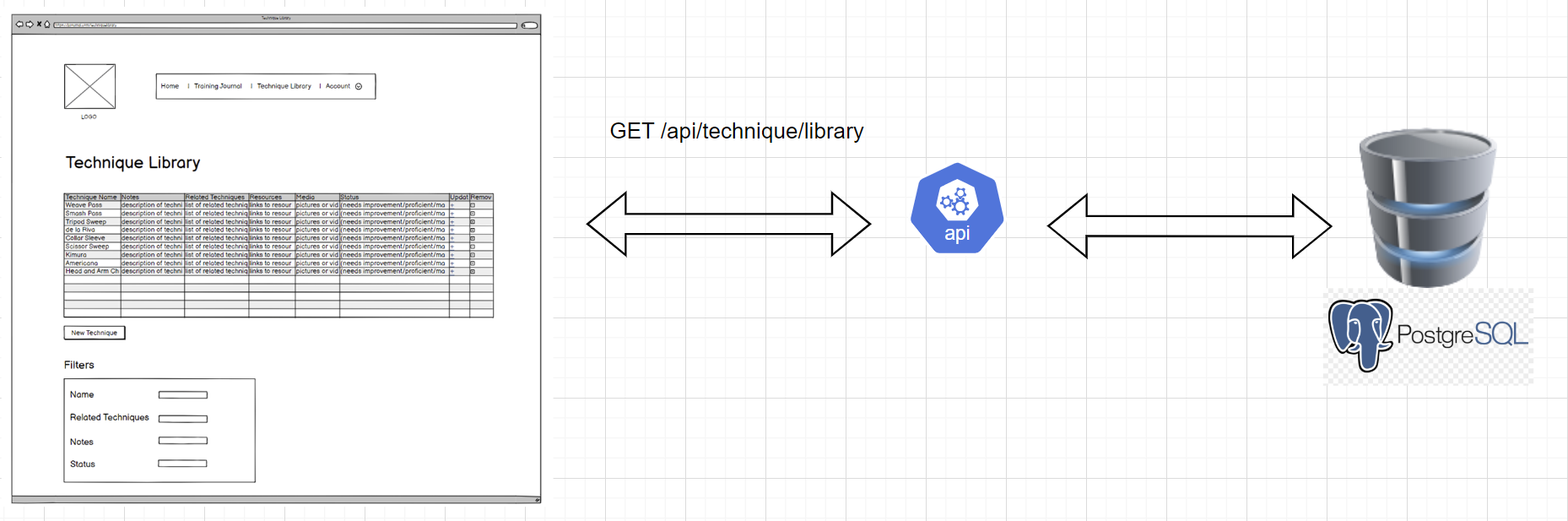
HTTP/1.1 403 Forbidden

Content-Type: application/json

{

"error": "Unauthorized access"

}



Method: POST

URL: https://brazilianjiujitsujournal.com/api/technique/entry

Purpose: When a user clicks “New Technique” on their Technique Library page, this is the endpoint that will be reached that will handle the creation of a new technique entry.

Sample Request:

POST /api/technique/entry HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

Cookie: sessionid=abc123

{

"name": "Kimura",

"type": "Submission",

"description": "A shoulder lock that applies pressure to the upper arm"

}

Success Response:

HTTP/1.1 201 Created

Content-Type: application/json

{

"message": "Technique added successfully"

}

Error Response:

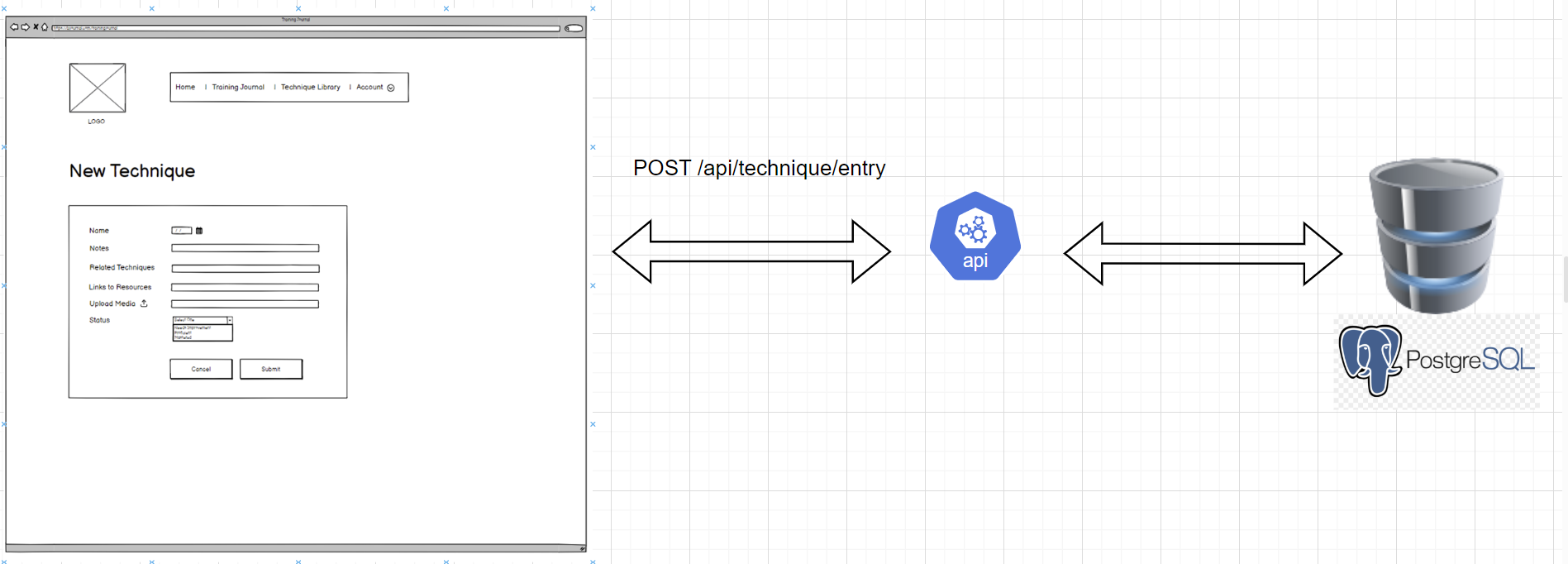
HTTP/1.1 400 Bad Request

Content-Type: application/json

{

"error": "Missing required fields"

}



Method: PUT

URL: https://brazilianjiujitsujournal.com/api/technique/update/{technique\_id}

Purpose: If a user selects the update option on a technique, this is the endpoint that will handle updating the details of that particular technique.

Sample Request:

PUT /api/technique/update/{technique\_id} HTTP/1.1

Host: brazilianjiujitsujournal.com

Content-Type: application/json

Cookie: sessionid=abc123

{

"name": "Updated Kimura",

"type": "Submission",

"description": "Updated description of the technique"

}

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"message": "Technique updated successfully"

}

Error Response:

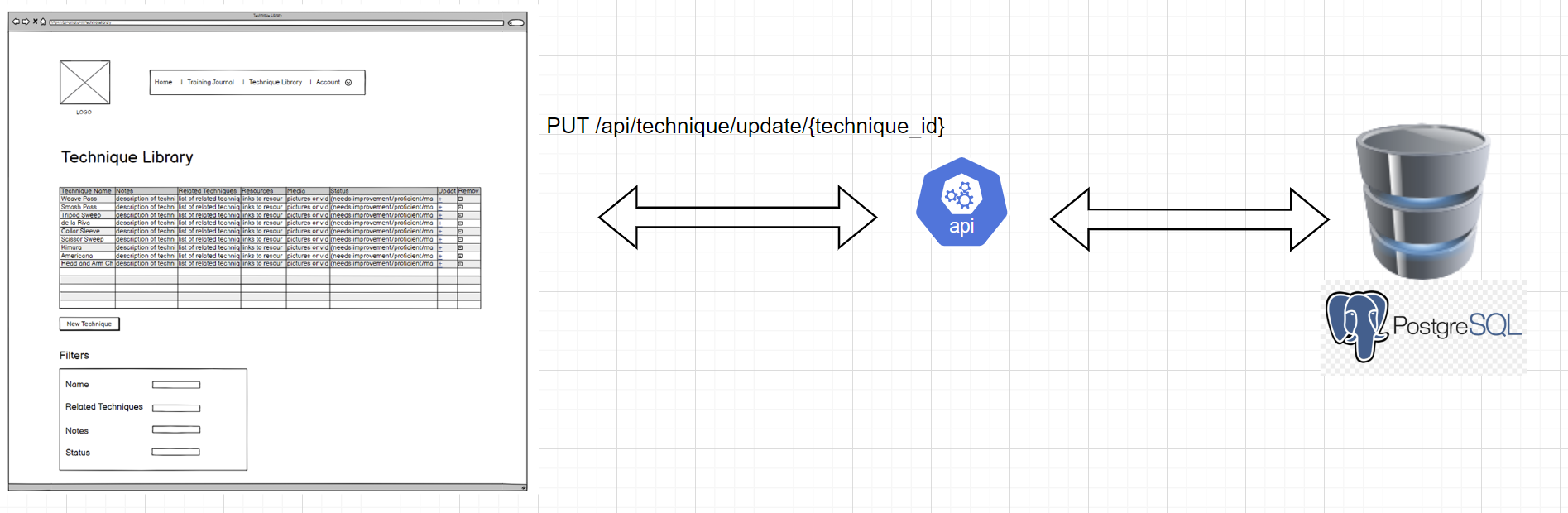
HTTP/1.1 404 Not Found

Content-Type: application/json

{

"error": "Technique not found or unauthorized"

}



Method: GET

URL: https://brazilianjiujitsujournal.com/api/ technique/library?query=[searchterm]

Purpose: This endpoint serves to handle searches a user might perform on the entries in their Technique Library.

Sample Request:

GET /api/technique/library?query=Kimura HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"results": [

{

"name": "Kimura",

"type": "Submission",

"description": "A shoulder lock that applies pressure to the upper arm."

},

... // Additional results here

]

}

Error Response:

HTTP/1.1 200 OK

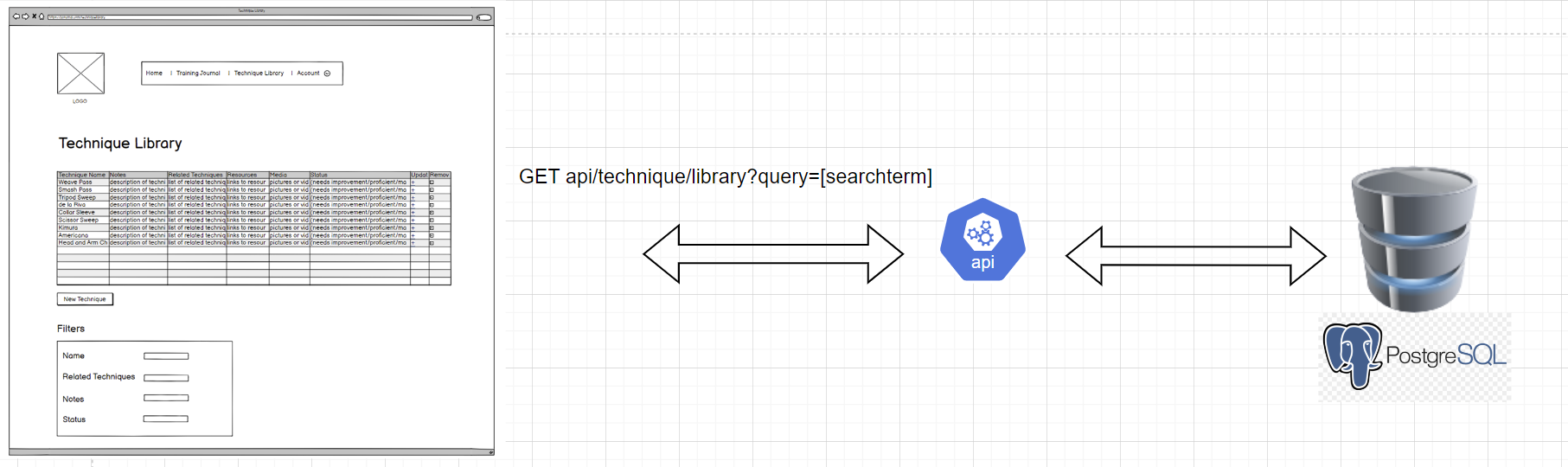
Content-Type: application/json

{

"error": "No techniques found matching the criteria",

"results": [] // Empty results indicating no matches found

}



Method: DELETE

URL: https://brazilianjiujitsujournal.com/api/technique/library/{technique\_id}

Purpose: This will handle the deletion of specific technique entries.

Sample Request:

DELETE /api/technique/{technique\_id} HTTP/1.1

Host: brazilianjiujitsujournal.com

Cookie: sessionid=abc123

Success Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"message": "Technique deleted successfully"

}

Error Response:

HTTP/1.1 404 Not Found

Content-Type: application/json

{

"error": "Technique not found or unauthorized"

}

