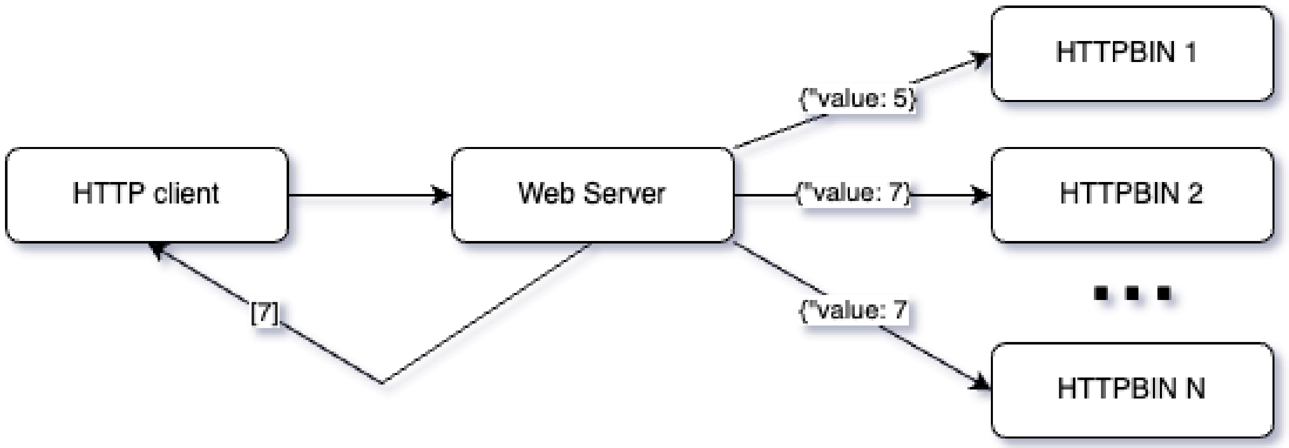
Task: Web Server

Create a web server in **Rust,** that has the following functionalities:

/run Endpoint: When called, this endpoint should send **30 POST requests** to https://httpbin.org/post. Each request should have a JSON body in the format *{"value": **N**}*, where ***N*** is a randomly generated number in the range **[0, 10]**. The server should parse the responses and extract the *json.value* field from each JSON response.

It should then calculate and return a collection containing the numbers that appear more than once (i.e., the most frequent numbers) in ascending order.

Examples:
For the collection: [3, 2, 5, 1, 5, 7, 2, 1], the response should be [1, 2, 5].
For the collection: [5, 7, 7], the response should be [7] (see picture below).



Database Integration: The server should save each request and its corresponding generated value from https://httpbin.org/post into a **MySQL** database. Each stored request should have a unique identifier and the generated value.

CRUD API: Implement a **REST-like CRUD API** for managing the stored data in the **MySQL** database. The **API** should include endpoints for **creating**, **reading**, **updating**, and **deleting** records. The data should include both request identifiers and the corresponding generated values.

User Authentication: Create a basic authentication system where users can authenticate with a username and password before using the **CRUD API**.
See: Basic Auth Wiki.

Preferable technologies to use:

Runner: tokio

Web-framework: actix-web v4

DB: sqlx, sea-orm

HTTP library: reqwest

Logging: env_logger

Serializing / deserializing: serde_json

Requirements:

Develop the web server, including the functionalities, described above.

Provide proper error handling for HTTP requests, database interactions, and authentication.

Create a test suite that includes testing the /run endpoint and authentication. Feel free to use any testing libraries you want (httpmock etc.).

Provide a testing-friendly environment (docker + "how to run" readme is more than enough).

How to earn extra points:

Write clean and readable code.

Pay attention to the speed of the “/run” endpoint.

Isolate domain logic from external factors, like requests handling. Some guidelines from popular software design concepts (hexagonal architecture, DDD, etc.) are appropriate here.

Having both unit and integration tests will be a plus.