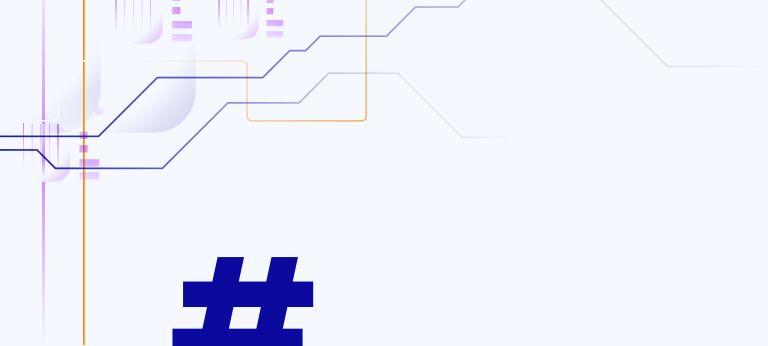




# **Week 3**

# **Backend Workshop**

Hack4Impact Junior Development Team



**#**

**Where We Left Off**

# Hack4Impact Backend Slack Channel



# CRUD Operations

- Four basic operations a software application should be able to perform
- CREATE, READ, UPDATE, DELETE
- Necessary to implement a persistent storage application

# CRUD Operations

- Create collection schema
  - New folder 'models' > new file 'Todo.js'
- Require model in index.js

# #1 Create

- Change POST method so when we click the add button our app inserts data into the database

The screenshot displays the MongoDB Atlas web interface for a project named 'todo-app'. The top navigation bar includes tabs for Overview, Real Time, Metrics, Collections (which is the active tab), Search, Profiler, Performance Advisor, Online Archive, and Cmd Line Tools. On the right, it shows the version as 6.0.10 and the region as AWS N. Virginia (us-east-1). Below the navigation bar, it indicates 'DATABASES: 10' and 'COLLECTIONS: 24'. On the left sidebar, there is a '+ Create Database' button and a search bar for namespaces. A list of sample namespaces is shown, including 'sample\_airbnb', 'sample\_analytics', 'sample\_geospatial', 'sample\_guides', 'sample\_mflix', 'sample\_restaurants', 'sample\_supplies', 'sample\_training', 'sample\_weatherdata', and a 'test' database which contains a 'todos' collection. The main panel shows the 'test.todos' collection with statistics: STORAGE SIZE: 24KB, LOGICAL DATA SIZE: 65B, TOTAL DOCUMENTS: 1, and INDEXES TOTAL SIZE: 24KB. It has tabs for Find (selected), Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. An 'INSERT DOCUMENT' button is present. A filter bar allows for querying with the text 'Type a query: { field: 'value' }'. Below, the 'QUERY RESULTS: 1-1 OF 1' section shows a single document: 

```
{
  "_id": ObjectId('6508ff6927db505b511f52f1'),
  "content": "task 1",
  "date": "2023-09-19T01:54:49.927+00:00",
  "__v": 0
}
```

## #2 Read

- Read data from the database
- To make the items visible in the app
- Change GET method and todo.ejs

## #3 Update

- Create a new file 'todoEdit.ejs' in 'views' folder
- We will create a new view where the user edits a task
- Create UPDATE method in 'index.js'



## #4 Delete

- User can delete a task
- Use the method findByIdAndRemove