Event Organizer Technical Report

Jasser saad

January 21, 2024

Introduction

- Welcome to the Event Organizer Technical Report presentation.
- The Event Organizer project is designed to streamline the process of managing and coordinating events through a powerful and efficient backend system.
- This technical report provides insights into the project's architecture, technologies used, functionalities, and security considerations.

Technologies Used

- **Backend Framework:** ExpressJS and NodeJS
- **Database:** MongoDB (NoSQL)
- **Authentication:** JSON Web Tokens (JWT)
- **API Testing:** Postman

Backend Framework

- The backend relies on the robust and scalable ExpressJS and NodeJS framework.
- ExpressJS simplifies the process of building a web server, making it an ideal choice for developing RESTful APIs.
- Key features include routing, middleware, template engine, and extensibility.

NodeJS

- NodeJS is the runtime that enables server-side JavaScript execution.
- Asynchronous I/O, Package Management (NPM), Cross-Platform Compatibility, and Scalability are key aspects of NodeJS.

Database

- The Event Organizer project relies on MongoDB, a NoSQL database, to manage event-related data.
- MongoDB's NoSQL architecture offers flexibility, scalability, and efficient querying for dynamic event information.

NoSQL and MongoDB

- MongoDB's schema-less approach is advantageous for handling dynamic event data.
- Flexibility, scalability, and efficient querying make MongoDB an ideal choice for the Event Organizer project.

Authentication

- To safeguard system functionalities, secure authentication mechanisms have been implemented, leveraging technologies such as JSON Web Tokens (JWT).

API Testing

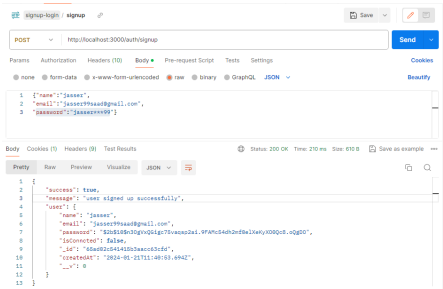
- Postman, a comprehensive API testing tool, is employed to validate and evaluate the effectiveness of the API.

Functionalities

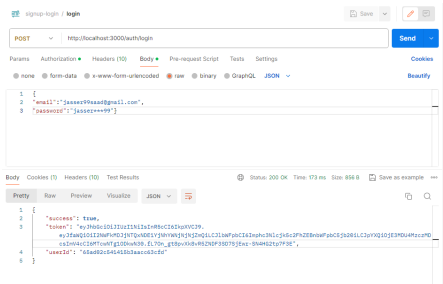
- **Authentication:** Robust authentication mechanism using JSON Web Tokens (JWT).
- **Event Management:** CRUD operations for managing events.
- **Viewing Events:** API endpoints to view a list of available events and detailed information about a specific event.

Authentication

- **Endpoint: POST /auth/signup:** New users can sign up for the application.

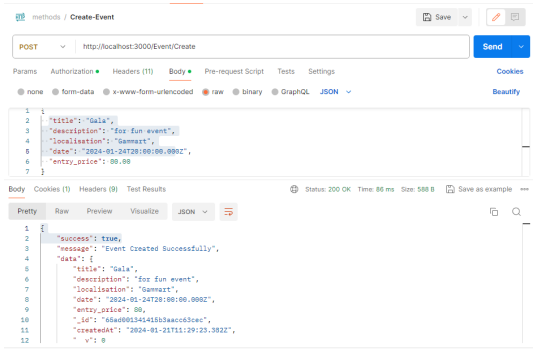


- **Endpoint: POST /auth/login:** Issuance of a JWT token upon successful authentication.

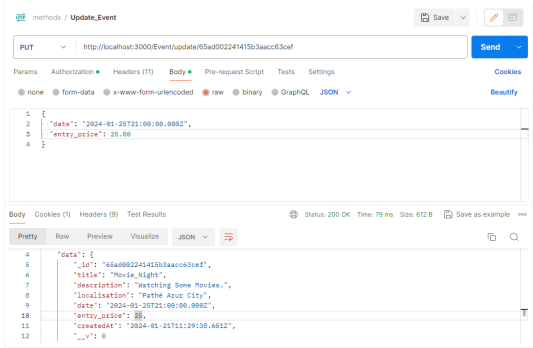


Event Management

- **Creating New Event:** Users can create new events through the API.
- **Endpoint: POST /events/create:** Submitting event details through the API.

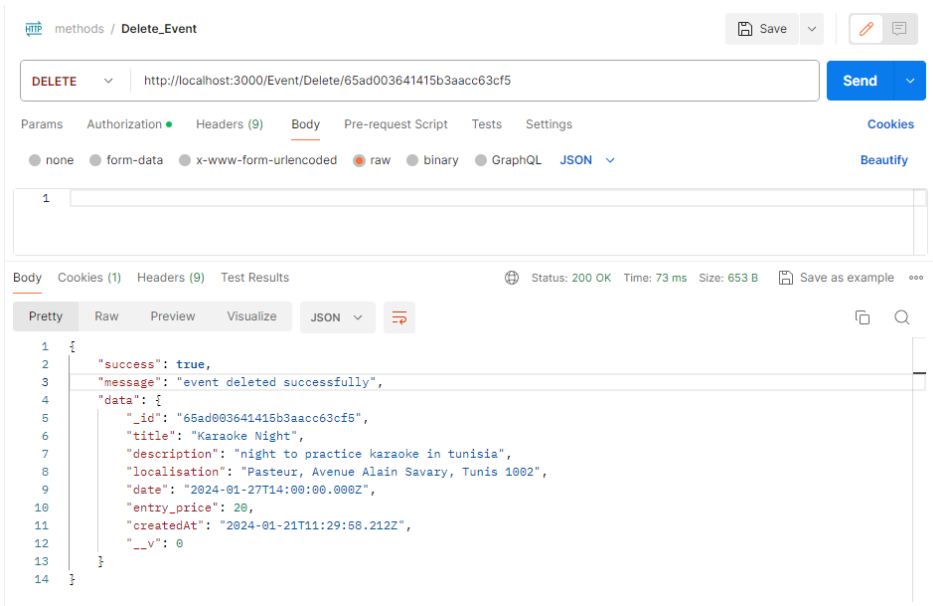


- **Updating Event:** Users can update existing event information.
- **Endpoint: PUT /events/update/:eventId:** Updating event details through the API.



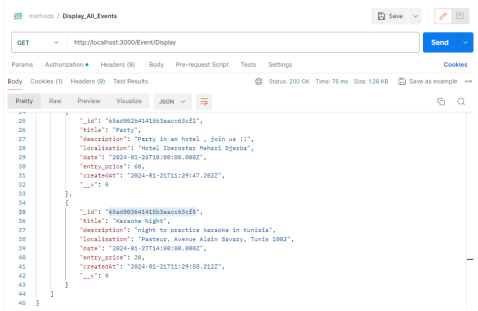
Deleting Event: Users can delete events through the API.

Endpoint: DELETE /events/delete/:eventId: Removing a specified event from the database.

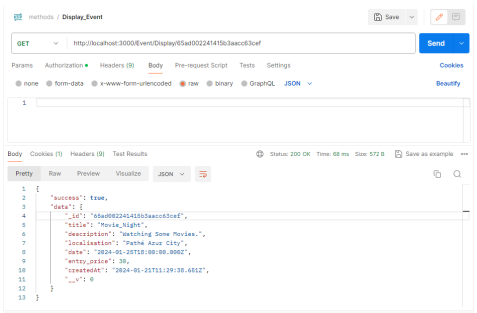


Viewing Events

- **Endpoint: GET /events/display:** Retrieving a list of available events.



- **Endpoint: GET /events/display/:eventId:** Retrieving detailed information about a specific event.



Security Considerations

- **Authentication:** Robust authentication mechanisms, including the use of JWT.
- **Authorization:** Proper authorization checks to control access and actions within the system.

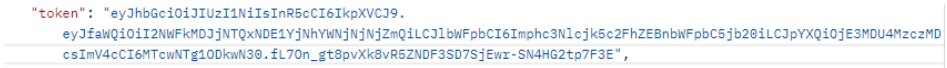


Figure: Token generated

- **Data Encryption:** Encryption of sensitive data to enhance overall system security.

Conclusion

- The Event Organizer project is meticulously designed to provide a seamless and secure experience for managing events.
- The RESTful API architecture ensures scalability, allowing the system to handle varying loads efficiently.
- The chosen technologies, including ExpressJS, NodeJS, and MongoDB, strike a balance between performance and security.
- The use of Postman as a testing tool validates the effectiveness and reliability of the API, ensuring a robust and error-free application.