Part 2 Report: Micro-Service Implementation for Trail Application 1. Introduction

This report provides an overview of the design and implementation of a micro-service created for the Trail Application, expanding on the foundation laid in CW1. It covers how the microservice was designed, implemented, and tested while addressing critical aspects like data privacy, security, and integrity.

You'll find UML diagrams, an evolved ERD, SQL implementations under the CW2 schema, and an evaluation of the micro-service's functionality. The report concludes with reflections on the work done and areas for improvement.

Links:

- GitHub Repository

---

2. Background

The Trail Application allows users to manage trails, activities, and related locations for outdoor adventures. This micro-service was developed as an independent module to handle core operations such as adding, reading, updating, and deleting trails. It also introduces logging functionality to ensure transparency and traceability.

The service connects to the Trail Application through REST APIs, providing a modular approach that simplifies both scalability and maintainability.

---

3. Design

3.1 UML Diagram

The UML diagram outlines the key components of the micro-service, including the TrailController, TrailService, and Database interactions. It visually represents how requests flow from API endpoints to database operations.

3.2 Logical ERD

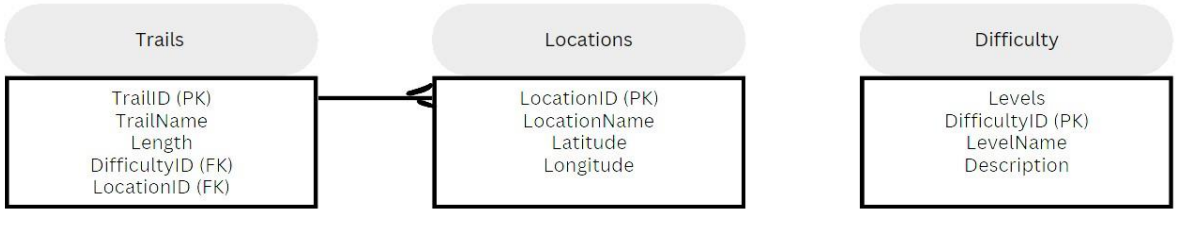
Building on CW1, the ERD for CW2 has been refined and includes the CW2.Trails table, relationships with TrailActivities, and the CW2.TrailAdditionLog.

Key Enhancements:

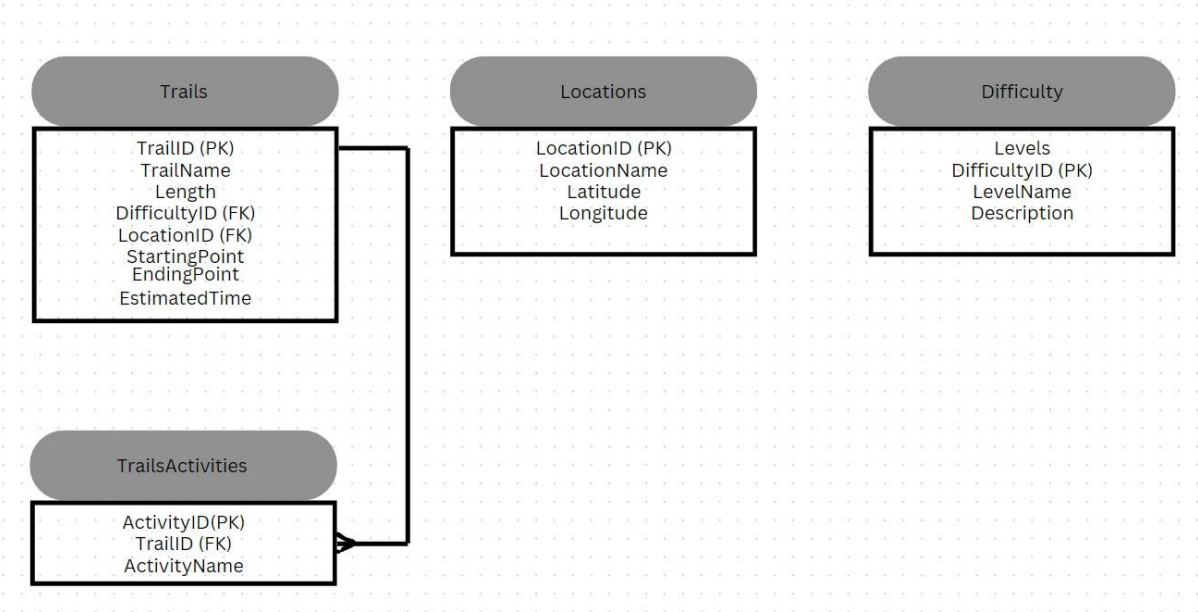
- Tables and columns were renamed for clarity and consistency.
- Constraints were added to strengthen data integrity.
- The TrailAdditionLog table was introduced to capture details like who added trails and when.

3.3 Physical ERD

Key Enhancements:



- Tables and Columns Renaming: For consistency and clarity, some tables and columns were renamed to better reflect their purposes and relationships.
- Constraints: Data integrity was strengthened by adding foreign key constraints, ensuring referential integrity between the tables.
- New Table - TrailAdditionLog: The TrailAdditionLog table was introduced to keep track of who added which trail and when. This table is linked to the Trails table and helps capture audit information regarding trail additions**.**



4. Legal, Social, Ethical, and Professional (LSEP)

4.1 Privacy

- No sensitive user information is logged or exposed, ensuring compliance with data privacy standards.
- Logging is limited to non-sensitive data such as trail IDs and usernames.

4.2 Integrity

- Strong foreign key constraints and transactional consistency ensure data integrity.
- CRUD operations follow strict validation checks to prevent data corruption.

4.3 Security

- The micro-service uses parameterized queries to prevent SQL injection attacks.
- Authentication mechanisms are implemented to secure API endpoints.

4.4 Preservation

- The TrailAdditionLog acts as an audit trail, preserving critical information about trail additions for accountability.

---

5. Implementation

5.1 SQL Reimplementation under CW2

The SQL queries from CW1 were reimplemented and improved under the CW2 schema. The procedures handle inserting, updating, deleting, and reading data effectively while maintaining database integrity.

Example of Insert Procedure:

```
CREATE PROCEDURE CW2.InsertTrail
    @TrailName VARCHAR(100),
    @LocationID INT,
    @Length DECIMAL(5, 2),
    @DifficultyLevelID INT,
    @StartingPoint VARCHAR(100),
    @EndingPoint VARCHAR(100),
    @EstimatedTime VARCHAR(50)
AS
BEGIN
    INSERT INTO CW2.Trails (TrailName, LocationID, Length, DifficultyLevelID, StartingPoint, EndingPoint, EstimatedTime)
    VALUES (@TrailName, @LocationID, @Length, @DifficultyLevelID, @StartingPoint, @EndingPoint, @EstimatedTime);
END;
```

6. Evaluation

6.1 Testing

- Unit and integration tests were performed on all CRUD operations and trigger functionality.
- Endpoints were tested using tools like Postman, verifying data consistency and expected responses.

6.2 Results

- The trigger successfully logged new trail additions to the CW2.TrailAdditionLog table.
- All CRUD operations worked seamlessly with the CW2 schema.

6.3 Areas for Further Work

- Implementing additional logging for updates and deletions.
- Enhancing API response formats with detailed error handling.

Conclusion

This report details the design, implementation, and evaluation of the micro-service for the Trail Application. The GitHub repository and hosted service demonstrate the practical application and functionality of the solution. Future work will focus on enhancing the logging and extending the micro-service capabilities.