

COMP2001 CW2 – Report on Microservice Implementation

Introduction:

Within this report, I will be designing and implementing a microservice which can be used to store information about specific trails which users can walk around the country. These details will range from location, estimated time and difficulty, to things such as user ratings. I will be showcasing the full design process, as well as stating any issues which need to be addressed concerning the implementation of this microservice using the LSEP guides which will be explained further later in this report. Additionally, I will be showing the implementation of the microservice, and finally evaluating it, stating what I think went well and what I think I can improve on were I to do this project again.

The GitHub link for my code can be found here: <https://github.com/Mark7567/COMP2001-CW2>

The docker image can be found at: mhardy101/comp2001

Background:

The microservice which I have implemented allows users to view trails which have been created by admins, showing the key information, such as the trail's location, distance and elevation, as well as things such as difficulty and any tags which could relate to the trail. The intent of the microservice is to allow users to see different trails in different areas, so they know whether or not they want to follow that route based on the provided information. The purpose of the overall application is to provide a database of trails across the world, which can be created by admins. Users are able to create an account, which they can then use to find these trails, as well as favourite them and leave comments relating to what they thought about the trail. The trail microservice I have created links in with this as it provides the main database which the trails will be fetched from to then be able to show them to the user. The microservice includes details regarding the trails themselves, as well as any relevant tags that could relate to the trail, such as whether they are dog friendly, or include lots of narrow pathways.

Design:

For the design of this microservice, I created many ERDs to showcase the relationships between the different elements of the microservice, as well as normalising data to be able to group them under relevant headings, giving me a clear indication of what each table should consist of. I also created field definition grids for each table which show which attributes belong to which entity. This was used to create the databases within SQL as it showed where everything needed to be, as well as any validation which was required for each attribute. I will show each of these design elements below.

Normalisation:

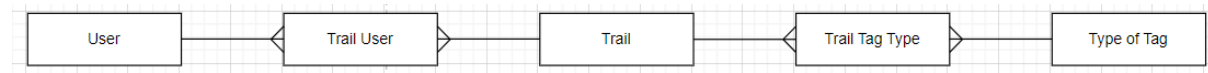
The normalisation used for this project was provided by Martin Read and contains all normalised forms for the data, as well as the relation names to be used for the tables.

Trail Normalisation Summary

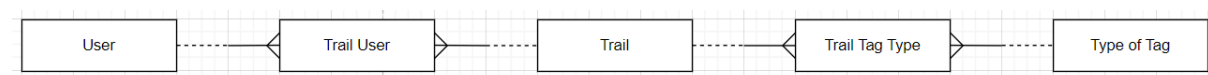
UNF	1NF	2NF	3NF	Relation name
<u>Trail ID</u>	<u>Trail ID</u>	<u>Trail ID</u>	<u>Trail ID</u>	TRAIL
Trail name	Trail name	Trail name	Trail name	
Difficulty	Trail Summary	Trail Summary	Trail Summary	
Location	Trail Description	Trail Description	Trail Description	
Length	Difficulty	Difficulty	Difficulty	
Elevation gain	Location	Location	Location	
Route type	Length	Length	Length	
Trail Summary	Elevation gain	Elevation gain	Elevation gain	
(Trail Feature ID	Route type	Route type	Route type	
Trail Feature)				
Trail Description	<u>Trail ID</u>	<u>Trail ID</u>	<u>Trail ID</u>	TRAIL-FEATURE
	<u>Trail Feature ID</u>	<u>Trail Feature ID</u>	<u>Trail Feature ID</u>	
	Trail Feature	<u>Trail Feature ID</u>	<u>Trail Feature ID</u>	FEATURE
		Trail Feature	Trail Feature	

ERDs:

This is the partial ERD I created for the design. It does not include optionality for the entity, and also includes a trail user entity. While I feel this is necessary as a link entity for the ERD, it is not a table to be created within the database as I believe that would exist under the user microservice and not the trail microservice, making it irrelevant for this project.



This is the final ERD I created for the design of this database. It is very similar to the partial ERD created after the normalisation, however it shows the optionality of each entity. Once again, it includes the trail user link entity as I feel this is necessary for the ERD, but I will not create the table for that link entity as it is not part of this microservice



Field Definition Grids:

Entity Name: Trail

Attribute Names	Descriptions	Data Types	Size (*=size)	Possible Data Values	Optional?	Validation Rules	Key?
Trail ID	Unique identifier for trail	Alphanumeric	6	TR0000 – TR9999	N	Start with TR, End with 4 numbers	Primary Key
Trail Name	Name tied to trail ID	Alphanumeric	128	Any	N		
Trail Summary	Summary tied to trail ID	Alphanumeric	255	Any	N		
Trail Description	Description tied to trail ID	Alphanumeric	255	Any	N		
Difficulty	Difficulty tied to trail ID	Alpha	8	Easy, Moderate, Hard	N	Must be easy, moderate or hard	
Location	Location tied to trail ID	Alphanumeric	128	Any	N		
Length	Length tied to trail ID	Numeric	5	0.0 – 999.9	N	Must be float value between 0.0 and 999.9	
Elevation Gain	Elevation gain tied to trail ID	Numeric	5	0 - 99999	N	Must be integer value between 0 and 99999	

Route Type	Route type tied to trail ID	Alpha	14	Loop, Out and Back, Point to Point	N	Must be loop, out and back or point to point	
Owner ID	Unique identifier for owner	Alphanumeric	6	UR0000 – UR9999	N	Start with UR, End with 4 numbers	Foreign Key
Point 1 Latitude	Latitude of point 1	Numeric	4	-90.0 – 90.0	N	Must be float value between -90.0 and 90.0	
Point 1 Longitude	Longitude of point 1	Numeric	5	-180.0 – 180.0	N	Must be float value between -180.0 and 180.0	
Point 1 Description	Description of point 1	Alphanumeric	255	Any	Y		
Point 2 Latitude	Latitude of point 2	Numeric	4	-90.0 – 90.0	N	Must be float value between -90.0 and 90.0	

Point 2 Longitude	Longitude of point 2	Numeric	5	-180.0 – 180.0	N	Must be float value between -180.0 and 180.0	
Point 2 Description	Description of point 2	Alphanumeric	255	Any	Y		
Point 3 Latitude	Latitude of point 3	Numeric	4	-90.0 – 90.0	N	Must be float value between -90.0 and 90.0	
Point 3 Longitude	Longitude of point 3	Numeric	5	-180.0 – 180.0	N	Must be float value between -180.0 and 180.0	
Point 3 Description	Description of point 3	Alphanumeric	255	Any	Y		
Point 4 Latitude	Latitude of point 4	Numeric	4	-90.0 – 90.0	N	Must be float value between -90.0 and 90.0	
Point 4 Longitude	Longitude of point 4	Numeric	5	-180.0 – 180.0	N	Must be float value between -180.0 and 180.0	
Point 4 Description	Description of point 4	Alphanumeric	255	Any	Y		

Point 5 Latitude	Latitude of point 5	Numeric	4	-90.0 – 90.0	N	Must be float value between -90.0 and 90.0	
Point 5 Longitude	Longitude of point 5	Numeric	5	-180.0 – 180.0	N	Must be float value between -180.0 and 180.0	
Point 5 Description	Description of point 5	Alphanumeric	255	Any	Y		

Entity Name: User

Attribute Names	Descriptions	Data Types	Size (*=size)	Possible Data Values	Optional?	Validation Rules	Key?
User ID	Unique identifier for user	Alphanumeric	6	UR0000 – UR9999	N	Start with UR, End with 4 numbers	Primary Key
Email Address	Email tied to user ID	Alphanumeric with special characters	50	Any@Any.any	N	Must include “@” and a “.”	
Password	Password tied to user ID	Alphanumeric with special characters	20	Any	N		
Role	Role tied to user ID	Alpha	5	User, Owner	N	Must be either user or owner	
Username	Username tied to user ID	Alphanumeric with special characters	64	Any	N		

Entity Name: Feature

Attribute Names	Descriptions	Data Types	Size (*=size)	Possible Data Values	Optional?	Validation Rules	Key?
Trail Feature ID	Unique identifier for feature	Alphanumeric	6	TF0000 – TF9999	N	Start with TF, End with 4 numbers	Primary Key
Feature	Description and name of the feature	Alphanumeric with spaces	255	Any	Y		

Entity Name: Trail Feature

Attribute Names	Descriptions	Data Types	Size (*=size)	Possible Data Values	Optional?	Validation Rules	Key?
Trail ID	Unique identifier for trail	Alphanumeric	6	TR0000 – TR9999	N	Start with TR, End with 4 numbers	Primary Key, Foreign Key
Trail Feature ID	Unique identifier for feature	Alphanumeric	6	TF0000 – TF9999	N	Start with TF, End with 4 numbers	Primary Key, Foreign Key

Legal, Social, Ethical and Professional (LSEP):

In this section, I will discuss the legal, social, ethical and professional considerations which are relevant to this project and explain the effect they will have on its development.

The first concern I will cover is a legal issue relating to this project. One of these legal concerns is GDPR, which provides lots of rules and regulations surrounding personal data, including collection, management and usage of this data. Since personal data needs to be stored within the user table of this database, I need to ensure I follow these rules and do not breach any legal requirements. I can do this by making sure to only collect relevant information, such as email addresses and passwords, and making sure it is protected from harm by implementing correct security measures to keep it safe.

Next, I will discuss one of the social issues surrounding this project, which is data transparency. Although only small amounts of information are required, being the user's email address and a password, some users may not wish to provide these if I am not transparent with the relevant security measures in place to protect their information from being used maliciously. This is a concern which needs to be addressed through clearly stating why an email address is required, and what will be done with it.

Furthermore, there are ethical concerns which need to be considered with the project. An example of this is ensuring that I allow for consumer data privacy, including only collecting minimal data. To ensure I am compliant with this, I am only taking in minimal information about users, being their email address, password and role. This is vital information to allow for the verification of users, and ensuring that they have the relevant permissions, while also making sure I remain ethical since I am not asking for any information which may be irrelevant.

Finally, I will discuss the professional issues which may be of concern with the project. An example of these professional issues is ensuring my code is of high quality. The reason for ensuring high code quality is so that it can be easily debugged in the future or built upon and expanded. I will ensure that this is not an issue through using things such as comments to explain what each section of my code does, as well as indentation to make sure that it is readable, and all makes sense at a glance without looking messy.

Implementation:

The implementation of the microservice was done through a variety of technology, which was each used in its own way to achieve the end result. These technologies were swagger, marshmallow, flask, python, jupyter notebook and pyodbc. I used a combination of all of these to help to create the microservice. When implementing the tables, I used the field definition grid created in the design section of this report to figure out the SQL and which attributes should exist in which tables. I then followed this up by creating python models within a models.py file to help with the CRUD procedures, as well as to help to prevent against things such as SQL injections which could harm the microservice. After creating the tables and relevant models, I implemented the CRUD procedures for each table. These involved creating, retrieving, updating and deleting information in each table. While these procedures were present in most tables, I did not feel it necessary to implement an update procedure within the trail features table, since this was simply stating the links between the trail and their relevant features. Additionally, I did not feel it was relevant to add an update procedure into the users table, since this would exist within the user microservice and not that of the trail. Finally, I deemed it necessary to implement an authentication feature for the user table as this

would allow me to follow the brief that only admins can create trails, and therefore I needed a method to verify admins.

Evaluation:

Overall, I felt that the creation of this project went incredibly well, despite the lack of knowledge which I had surrounding database creation before it began. This was a limiting factor in how effective I was able to make the database, however through a combination of lectures, other students and Google, I managed to find solutions to most of the problems which I encountered to allow me to complete the project. I feel that if I were to attempt something similar to this in the future, I would be able to approach it with a much clearer mindset than I originally started this project with, which would help me to complete it much more effectively and quicker.

Additionally, my use of version control was very effective throughout the project. Every time I created a function or added a table which I had tested to work, I committed it to a GitHub repository, which can be found using the link in the introduction section of this document. This has allowed me to show my progress over time, as well as allowing me to backtrack whenever any errors occurred within the code as, if necessary, I could reset to a previous version of my project with working code. As I have used GitHub for previous projects, it allowed me to make effective use of it within this project.

In order to make sure the microservice worked as intended, I had to perform many tests on it. These tests involved using the Swagger UI linked to my database to test the addition, deletion, retrieval and updating of data inside all of my tables. I tested every time I created or changed one of the functions to ensure everything remained up-to-date, and only committed to the GitHub repository once I was happy that everything was functioning as intended. I have included pictures below to show each function working on each table.

User Authentication:

The screenshot displays the Swagger UI interface for a REST API. At the top, a green bar indicates a **POST** method for the endpoint `/authenticate`, with a description 'Authenticates user'. Below this, the 'Parameters' section shows 'No parameters'. The 'Request body' section is marked as 'required' and has a dropdown menu set to 'application/json'. The request body is a JSON object:

```
{  "email": "grace@plymouth.ac.uk",  "password": "ISAD123!"}
```

. The 'Response body' section shows a 200 status code with the response:

```
Authentication complete: ['Verified', 'True']
```

. The 'Response headers' section lists:

```
connection: close
content-length: 45
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:13:45 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

. There are 'Cancel' and 'Reset' buttons in the top right, and a 'Download' button next to the response body.

Read All Users:

Code

Details

200

Response body

```
{
  {
    "email_address": "grace@plymouth.ac.uk",
    "password": "ISAD123!",
    "role": "user",
    "user_id": "UR0001",
    "username": "Grace Hopper"
  },
  {
    "email_address": "tim@plymouth.ac.uk",
    "password": "CMP2001!",
    "role": "user",
    "user_id": "UR0002",
    "username": "Tim Berners-Lee"
  },
  {
    "email_address": "ada@plymouth.ac.uk",
    "password": "insecurePassword",
    "role": "user",
    "user_id": "UR0003",
    "username": "Ada Lovelace"
  }
}
```

Download

Response headers

```
connection: close
content-length: 485
content-type: application/json
date: Sun,05 Jan 2025 20:19:04 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Create User:

POST

/user

Creates a user

Cancel

Reset

Parameters

No parameters

Request body required

application/json

User to create

```
{
  "email_address": "mark@test.com",
  "password": "abc123",
  "role": "user",
  "user_id": "UR0018",
  "username": "mark1"
}
```

Code

Details

201

Response body

```
{
  "email_address": "mark@test.com",
  "password": "abc123",
  "role": "user",
  "user_id": "UR0018",
  "username": "mark1"
}
```

Download

Response headers

```
connection: close
content-length: 127
content-type: application/json
date: Sun,05 Jan 2025 20:20:27 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read One User:

GET

/user/{user_id}

Reads one individual user

Cancel

Parameters

Name	Description
user_id * required	
string (query)	The unique ID of the user
	UR0018


Code

Details

200

Response body

```
{
  "email_address": "mark@test.com",
  "password": "abc123",
  "role": "user",
  "user_id": "UR0018",
  "username": "mark1"
}
```

 Download

Response headers

```
connection: close
content-length: 127
content-type: application/json
date: Sun, 05 Jan 2025 20:21:40 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Delete User:

DELETE

/user/{user_id} Deletes a user

Parameters

Cancel

Name	Description
user_id * required string (query)	The unique ID of the user

UR0018

Code

Details

200

Undocumented

Response body

```
UR0018 has been successfully deleted
```

 Download

Response headers

```
connection: close
content-length: 26
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:22:29 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read All Features:

Code

Details

200

Response body

```
[
  {
    "trail_feature": "hello",
    "trail_feature_id": "TF0001"
  },
  {
    "trail_feature": "Child Friendly",
    "trail_feature_id": "TF0002"
  },
  {
    "trail_feature": "Hiking",
    "trail_feature_id": "TF0003"
  },
  {
    "trail_feature": "Cave",
    "trail_feature_id": "TF0004"
  },
  {
    "trail_feature": "Forest",
    "trail_feature_id": "TF0005"
  }
]
```

 Download

Response headers

```
connection: close
content-length: 373
content-type: application/json
date: Sun, 05 Jan 2025 20:25:04 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Create Feature:

POST

/feature

Creates a new feature

Parameters

No parameters

Request body

required

application/json

Feature to create

```
{
  "trail_feature": "Hills",
  "trail_feature_id": "TF0006"
}
```

Code

Details

201

Response body

```
{
  "trail_feature": "Hills",
  "trail_feature_id": "TF0006"
}
```

Download

Response headers

connection: close
content-length: 63
content-type: application/json
date: Sun,05 Jan 2025 20:30:01 GMT
server: Werkzeug/2.2.2 Python/3.9.21

Read One Feature:

GET

/feature/{trail_feature_id}

Reads one specific feature

Parameters

Cancel

Name

Description

trail_feature_id

* required

string

(query)

The unique ID of the trail feature

TF0006

Code

Details

200

Response body

```
{
  "trail_feature": "Hills",
  "trail_feature_id": "TF0006"
}
```

Download

Response headers

connection: close
content-length: 63
content-type: application/json
date: Sun,05 Jan 2025 20:30:37 GMT
server: Werkzeug/2.2.2 Python/3.9.21

Update Feature:

PUT

/feature/{trail_feature_id}

Update feature

Parameters

Cancel

Reset

Name

Description

trail_feature_id

* required

string

(query)

The unique ID of the trail feature

TF0006

Request body

application/json

```
{
  "trail_feature": "No Hills"
}
```

Code

Details

201

Undocumented

Response body

```
{
  "trail_feature": "No hills",
  "trail_feature_id": "TF0006"
}
```

Download

Response headers

```
connection: close
content-length: 66
content-type: application/json
date: Sun, 05 Jan 2025 20:31:22 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Delete Feature:

DELETE

/feature/{trail_feature_id}

Deletes a feature

Parameters

Cancel

Name	Description
trail_feature_id * required	The unique ID of the trail feature
string (query)	<input type="text" value="TF0006"/>

Code

Details

200

Undocumented

Response body

```
TF0006 has been successfully deleted
```

Download

Response headers

```
connection: close
content-length: 36
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:32:45 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read All Trails:

Code

Details

200

Response body

```
{
  {
    "difficulty": "Easy",
    "elevation gain": 1000,
    "length": 15.5,
    "location": "Plymouth, Devon, UK",
    "owner_id": "UR0001",
    "pt1_desc": "Point 1",
    "pt1_lat": 05,
    "pt1_long": -15,
    "pt2_desc": "Point 2",
    "pt2_lat": 70,
    "pt2_long": -10,
    "pt3_desc": "Point 3",
    "pt3_lat": 75,
    "pt3_long": -5,
    "pt4_desc": "Point 4",
    "pt4_lat": 80,
    "pt4_long": 0,
    "pt5_desc": "Point 5",
    "pt5_lat": 85,
    "pt5_long": 5,
    "route_type": "Point to Point",
    "trail_description": "An awesome trail",
    "trail_id": "TR0001",
    "trail_name": "Awesome trail",
    "trail_summary": "An awesome trail"
  },
  "difficulty": "Hard"
}
```

Download

Response headers

```
connection: close
content-length: 3046
content-type: application/json
date: Sun, 05 Jan 2025 20:33:24 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Create Trail:

POST

/trail

Creates a new trail

Parameters

Cancel

Reset

No parameters

Request body

required

application/json

Trail to create

```
{
  "pt1_lat": 0,
  "pt1_long": 0,
  "pt2_desc": "Point 2",
  "pt2_lat": 10,
  "pt2_long": -20,
  "pt3_desc": "Point 3",
  "pt3_lat": 20,
  "pt3_long": -40,
  "pt4_desc": "Point 4",
  "pt4_lat": 30,
  "pt4_long": -60,
  "pt5_desc": "Point 5",
  "pt5_lat": 40,
  "pt5_long": -80,
  "route_type": "Out and Back",
  "trail_description": "The coolest trail",
  "trail_id": "TR0005",
  "trail_name": "Coolest Trail",
  "trail_summary": "A cool trail around a lake"
}
```

Code

Details

201

Response body

```
{
  "difficulty": "Easy",
  "elevation_gain": 1000,
  "length": 60.5,
  "location": "Alabama",
  "owner_id": "UR0001",
  "pt1_desc": "Point 1",
  "pt1_lat": 0,
  "pt1_long": 0,
  "pt2_desc": "Point 2",
  "pt2_lat": 10,
  "pt2_long": -20,
  "pt3_desc": "Point 3",
  "pt3_lat": 20,
  "pt3_long": -40,
  "pt4_desc": "Point 4",
  "pt4_lat": 30,
  "pt4_long": -60,
  "pt5_desc": "Point 5",
  "pt5_lat": 40,
  "pt5_long": -80,
  "route_type": "Out and Back",
  "trail_description": "The coolest trail",
  "trail_id": "TR0005",
  "trail_name": "Coolest Trail",
  "trail_summary": "A cool trail around a lake"
}
```

Download

Response headers

```
connection: close
content-length: 624
content-type: application/json
date: Sun, 05 Jun 2025 20:36:15 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read One Trail:

GET

/trail/{trail_id}

Reads one specific trail

Parameters

Cancel

Name	Description
trail_id <div>required</div>	The unique ID of the trail
string <div>(query)</div>	
<div>TR0005</div>	

Code

Details

200

Response body

```
{
  "difficulty": "Easy",
  "elevation_gain": 1000,
  "length": 60.9,
  "location": "Alabama",
  "owner_id": "000001",
  "pt1_desc": "Point 1",
  "pt1_lat": 0,
  "pt1_long": 0,
  "pt2_desc": "Point 2",
  "pt2_lat": 10,
  "pt2_long": -20,
  "pt3_desc": "Point 3",
  "pt3_lat": 20,
  "pt3_long": -40,
  "pt4_desc": "Point 4",
  "pt4_lat": 30,
  "pt4_long": -60,
  "pt5_desc": "Point 5",
  "pt5_lat": 40,
  "pt5_long": -80,
  "route_type": "Out and Back",
  "trail_description": "The coolest trail",
  "trail_id": "TR0005",
  "trail_name": "Coolest Trail",
  "trail_summary": "A cool trail around a lake"
}
```

Download

Response headers

```
connection: close
content-length: 624
content-type: application/json
date: Sun,05 Jan 2025 20:38:01 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Update Trail:

PUT

/trail/{trail_id}

Update trail details

Parameters

Cancel

Reset

Name	Description
trail_id ★ required	The unique ID of the trail
string (query)	
	TR0005

Request body

application/json

```
{
  "pt1_desc": "Pt1",
  "pt1_lat": 10,
  "pt1_long": -20,
  "pt2_desc": "Pt2",
  "pt2_lat": 20,
  "pt2_long": -40,
  "pt3_desc": "Pt3",
  "pt3_lat": 30,
  "pt3_long": -60,
  "pt4_desc": "Pt4",
  "pt4_lat": 40,
  "pt4_long": -80,
  "pt5_desc": "Pt5",
  "pt5_lat": 50,
  "pt5_long": -100,
  "route_type": "Point to Point",
  "trail_description": "A less cool trail",
  "trail_name": "Less cool trail",
  "trail_summary": "A less cool trail around the lake"
}
```

Code

Details

201

Response body

```
{
  "difficulty": "Hard",
  "elevation_gain": 1000,
  "length": 100,
  "location": "Plymouth",
  "owner_id": "000001",
  "pt1_desc": "Pt1",
  "pt1_lat": 10,
  "pt1_long": -20,
  "pt2_desc": "Pt2",
  "pt2_lat": 20,
  "pt2_long": -40,
  "pt3_desc": "Pt3",
  "pt3_lat": 30,
  "pt3_long": -60,
  "pt4_desc": "Pt4",
  "pt4_lat": 40,
  "pt4_long": -80,
  "pt5_desc": "Pt5",
  "pt5_lat": 50,
  "pt5_long": -100,
  "route_type": "Point to Point",
  "trail_description": "A less cool trail",
  "trail_id": "TR0001",
  "trail_name": "Less cool trail",
  "trail_summary": "A less cool trail around the lake"
}
```

Download

Response headers

```
connection: close
content-length: 621
content-type: application/json
date: Sun,05 Jan 2025 20:40:10 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Delete Trail:

DELETE

/trail/{trail_id}

Deletes a trail

Parameters

Cancel

Name	Description
trail_id <small>required</small>	The unique ID of the trail
string <small>(query)</small>	<div>TR0005</div>

Code

Details

200

Undocumented

Response body

TR0005 has been successfully deleted

Download

Response headers

connection: close
content-length: 36
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:42:11 GMT
server: Werkzeug/2.2.2 Python/3.9.21

Read All Trail Features:

Code

Details

200

Response body

```
{
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0001"
  },
  {
    "trail_feature_id": "TF0004",
    "trail_id": "TR0001"
  },
  {
    "trail_feature_id": "TF0005",
    "trail_id": "TR0001"
  },
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0002"
  },
  {
    "trail_feature_id": "TF0004",
    "trail_id": "TR0002"
  },
  {
    "trail_feature_id": "TF0003",
    "trail_id": "TR0004"
  }
}
```

Download

Response headers

connection: close
content-length: 411
content-type: application/json
date: Sun, 05 Jan 2025 20:43:18 GMT
server: Werkzeug/2.2.2 Python/3.9.21

Create Trail Feature:

POST

/trail-feature

Creates a new trail feature

Parameters

Cancel

Reset

No parameters

Request body required

application/json

Trail feature to create

```
{
  "trail_feature_id": "TF0002",
  "trail_id": "TR0004"
}
```


Code

Details

201

Response body

```
{
  "trail_feature_id": "TF0002",
  "trail_id": "TR0004"
}
```

Download

Response headers

```
connection: close
content-length: 59
content-type: application/json
date: Sun,05 Jan 2025 20:44:01 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read One Trail Feature (Trail):

GET

/trail-feature/{id}

Reads one specific trail feature

⌵

Parameters

Cancel

Name	Description
<div><div>id</div><div>✱ required</div></div> <div>string</div> <div>(path)</div>	The ID to search
<div>TR0002</div>	
<div><div>type</div><div>✱ required</div></div> <div>string</div> <div>(query)</div>	Trail ID or Trail Feature ID
<div>Trail</div> <div>▼</div>	


Code

Details

200

Response body

```
[
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0002"
  },
  {
    "trail_feature_id": "TF0004",
    "trail_id": "TR0002"
  }
]
```

Download

Response headers

```
connection: close
content-length: 139
content-type: application/json
date: Sun,05 Jan 2025 20:44:47 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Read One Trail Feature (Feature):

GET

/trail-feature/{id}

Reads one specific trail feature

⌵

Parameters

Cancel

Name	Description
<div><div>id</div><div>✱ required</div></div> <div>string</div> <div>(path)</div>	The ID to search
<div>TF0002</div>	
<div><div>type</div><div>✱ required</div></div> <div>string</div> <div>(query)</div>	Trail ID or Trail Feature ID
<div>Feature</div> <div>▼</div>	


Code

Details

200

Response body

```
[
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0001"
  },
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0002"
  },
  {
    "trail_feature_id": "TF0002",
    "trail_id": "TR0004"
  }
]
```

Download

Response headers

```
connection: close
content-length: 207
content-type: application/json
date: Sun,05 Jan 2025 20:45:41 GMT
server: Werkzeug/2.2.2 Python/3.9.21
```

Delete Trail Feature (Feature ID):

DELETE

/trail-feature/feature/{trail_feature_id}

Deletes all trails with specific trail feature id

Parameters

Cancel

Name	Description
trail_feature_id * required string (path)	The ID of the trail feature to delete from link entity table

TF0002

Code

Details

200
Undocumented

Response body

[<Trail_Feature TR0001, TF0002>, <Trail_Feature TR0002, TF0002>, <Trail_Feature TR0004, TF0002>] has been successfully deleted

Download

Response headers

connection: close
content-length: 126
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:46:20 GMT
server: Werkzeug/2.2.2 Python/3.9.21

Delete Trail Feature (Trail ID):

DELETE

/trail-feature/trail/{trail_id}

Deletes all trails with specific trail id

Parameters

Cancel

Name	Description
trail_id * required string (path)	The ID of the trail to delete from the link entity

TR0002

Code

Details

200
Undocumented

Response body

[<Trail_Feature TR0002, TF0004>] has been successfully deleted

Download

Response headers

connection: close
content-length: 62
content-type: text/html; charset=utf-8
date: Sun, 05 Jan 2025 20:47:02 GMT
server: Werkzeug/2.2.2 Python/3.9.21