https://gemini.google.com/u/1/app/8f755b2971a8fd3e

### OpenAPI 3.1 Spec (for Module TRAIL\_HIERARCHY - Revised Translation Model)

YAML

```

openapi: 3.1.0

info:

title: Via di Francesco - Core Trail Hierarchy API (Module 2)

version: v1.0.0

description: |-

API endpoints for managing and accessing core trail hierarchy data, including Trails, Routes, and Segments.

Supports multilingual content and standard pagination.

The `lang` query parameter determines the language of the primary translatable fields, with English as a fallback.

A `translations` object is provided for each translatable field, containing all available translations.

Authentication is required for write operations and potentially for accessing non-published content.

servers:

- url: https://<YOUR\_SUPABASE\_PROJECT\_REF>.supabase.co/rest/v1

description: Production Supabase API Server

- url: http://localhost:54321/rest/v1

description: Local Development Server

tags:

- name: Trails

description: Operations related to overarching pilgrimage trails.

- name: Routes

description: Operations related to specific routes or variations within trails.

- name: Segments

description: Operations related to granular, walkable sections of routes.

- name: Trail Associations

description: Operations for managing associations like trail regions, terrain types, and usage types.

- name: Segment Associations

description: Operations for managing associations like segment media and additional terrain.

- name: Master Data (Trail Hierarchy)

description: Lookup tables for terrain types and usage types relevant to trails.

components:

securitySchemes:

bearerAuth:

type: http

scheme: bearer

bearerFormat: JWT

description: Supabase JWT token for authenticated access.

parameters:

LangParam:

name: lang

in: query

description: ISO 639-1 language code for localized content (e.g., 'it', 'en'). Determines the language of primary translatable fields, defaulting to English if not provided or translation unavailable.

required: false

schema:

type: string

example: it

PageParam:

name: page

in: query

description: Page number for paginated results.

required: false

schema:

type: integer

minimum: 1

default: 1

PageSizeParam:

name: page\_size

in: query

description: Number of items per page for paginated results.

required: false

schema:

type: integer

minimum: 1

maximum: 100

default: 20

SelectParam:

name: select

in: query

description: |-

Comma-separated list of columns to retrieve.

Use `\*` for all columns.

Embed related resources using `foreign\_table(\*)` or `foreign\_table(column1,column2)`.

required: false

schema:

type: string

example: id,name,slug,routes(route\_id,name)

TrailIdParam:

name: trail\_id

in: path

required: true

description: Identifier of the trail.

schema:

type: integer

format: int64

RouteIdParam:

name: route\_id

in: path

required: true

description: Identifier of the route.

schema:

type: integer

format: int64

SegmentIdParam:

name: segment\_id

in: path

required: true

description: Identifier of the segment.

schema:

type: integer

format: int64

schemas:

Error:

type: object

required:

- code

- message

properties:

code:

type: string

message:

type: string

detail:

type: string

nullable: true

hint:

type: string

nullable: true

PaginationInfo:

type: object

properties:

current\_page: { type: integer }

page\_size: { type: integer }

total\_items: { type: integer }

total\_pages: { type: integer }

ImageVariants:

type: object

description: JSON object storing URLs or relative paths for different image variants/sizes.

additionalProperties:

type: string

format: uri-reference

example:

thumb\_100x100\_webp: "variants/media\_uuid\_thumb.webp"

MediaItemSummary: # Simplified for embedding

type: object

properties:

media\_id: { type: string, format: uuid }

media\_asset\_type: { type: string, enum: [image, document\_pdf, audio\_clip, video\_clip, gpx\_file, other\_file] }

file\_mime\_type: { type: string }

alt\_text: { type: string, nullable: true, description: "Localized alt text." }

alt\_text\_translations:

type: object

additionalProperties: { type: string }

description: "All available translations for alt text."

image\_variants\_json: { $ref: '#/components/schemas/ImageVariants' }

storage\_object\_path\_original: { type: string, nullable: true }

file\_name\_original: { type: string, nullable: true }

# ENUMs from Module 2

TrailDifficultyEnum: { type: string, enum: [easy, moderate, challenging, strenuous, variable] }

TrailOperationalStatusEnum: { type: string, enum: [fully\_operational, partially\_closed\_detours\_in\_place, seasonal\_access\_only, under\_development, closed, information\_unavailable] }

ContentVisibilityStatusEnum: { type: string, enum: [draft, pending\_review, published, archived] }

RouteCategoryEnum: { type: string, enum: [main\_section, official\_variant, unofficial\_variant, connector\_spur, extension, loop\_option, access\_route] }

SegmentSunExposureEnum: { type: string, enum: [mostly\_shaded, partially\_shaded, mostly\_exposed, variable] }

SegmentTravelDirectionEnum: { type: string, enum: [bidirectional, northbound\_only, southbound\_only, eastbound\_only, westbound\_only, uphill\_only, downhill\_only, clockwise\_only, counter\_clockwise\_only, as\_signposted] }

# --- terrain\_types\_master ---

TerrainTypeMaster:

type: object

description: Master definition of a terrain type.

properties:

id: { type: integer, readOnly: true }

code: { type: string, pattern: "^[a-z0-9\_]+$", maxLength: 50 }

name: { type: string, description: "Localized name of the terrain type (based on 'lang' param, fallback to English)." }

name\_translations:

type: object

additionalProperties: { type: string }

description: "All available translations for the name, including English."

description: { type: string, nullable: true, description: "Localized description." }

description\_translations:

type: object

additionalProperties: { type: string }

nullable: true

description: "All available translations for the description."

icon\_identifier: { type: string, nullable: true, maxLength: 100 }

display\_order: { type: integer, default: 0 }

is\_active: { type: boolean, default: true, readOnly: true }

# Audit fields omitted for brevity in this example, but would be included as readOnly

PaginatedTerrainTypeMasterList:

type: object

properties:

data: { type: array, items: { $ref: '#/components/schemas/TerrainTypeMaster' } }

pagination: { $ref: '#/components/schemas/PaginationInfo' }

# --- usage\_types\_master ---

UsageTypeMaster:

type: object

description: Master definition of a trail usage type.

properties:

id: { type: integer, readOnly: true }

code: { type: string, pattern: "^[a-z0-9\_]+$", maxLength: 50 }

name: { type: string, description: "Localized name." }

name\_translations:

type: object

additionalProperties: { type: string }

description: { type: string, nullable: true, description: "Localized description." }

description\_translations:

type: object

additionalProperties: { type: string }

nullable: true

icon\_identifier: { type: string, nullable: true, maxLength: 100 }

display\_order: { type: integer, default: 0 }

is\_active: { type: boolean, default: true, readOnly: true }

PaginatedUsageTypeMasterList:

type: object

properties:

data: { type: array, items: { $ref: '#/components/schemas/UsageTypeMaster' } }

pagination: { $ref: '#/components/schemas/PaginationInfo' }

# --- trails ---

TrailBase: # Fields that are part of create/update and store English in DB

type: object

properties:

# For translatable fields, DB stores English. API Create/Update takes English.

name: { type: string, maxLength: 255, description: "Primary name of the trail in English." }

slug: { type: string, maxLength: 255, pattern: "^[a-z0-9]+(?:-[a-z0-9]+)\*$", description: "URL-friendly identifier. Must be unique." }

trail\_short\_code: { type: string, nullable: true, maxLength: 50, pattern: "^[A-Z0-9\_-]+$" }

alternate\_names: { type: array, nullable: true, items: { type: string }, description: "Array of other known names in English." } # Stored as TEXT[] in DB

short\_description: { type: string, nullable: true, maxLength: 250, description: "Brief summary in English." }

full\_description: { type: string, nullable: true, description: "Detailed description in English." }

# ... (Repeat for all translatable\_en fields from previous TrailBase: historical\_significance, cultural\_significance, etc. naming them directly as they are in English in DB)

historical\_significance: { type: string, nullable: true }

cultural\_significance: { type: string, nullable: true }

pilgrimage\_focus: { type: string, nullable: true }

primary\_start\_point\_name: { type: string, nullable: true }

primary\_end\_point\_name: { type: string, nullable: true }

typical\_direction\_of\_travel: { type: string, nullable: true }

waymarking\_description: { type: string, nullable: true }

overall\_safety\_considerations: { type: string, nullable: true }

best\_seasons\_to\_walk: { type: array, nullable: true, items: { type: string } } # Stored as TEXT[]

key\_attractions\_summary: { type: string, nullable: true }

pilgrim\_credential\_info: { type: string, nullable: true }

contact\_organization\_name: { type: string, nullable: true }

primary\_data\_source\_credit: { type: string, nullable: true }

data\_licence\_info: { type: string, nullable: true }

general\_notes\_for\_pilgrims: { type: string, nullable: true }

meta\_description\_seo: { type: string, nullable: true, maxLength: 160 }

wordpress\_excerpt: { type: string, nullable: true }

# Non-translatable fields

estimated\_total\_distance\_km: { type: number, format: float, nullable: true }

estimated\_total\_duration\_days: { type: string, nullable: true }

overall\_difficulty: { $ref: '#/components/schemas/TrailDifficultyEnum', nullable: true }

operational\_status: { $ref: '#/components/schemas/TrailOperationalStatusEnum', default: information\_unavailable }

content\_visibility\_status: { $ref: '#/components/schemas/ContentVisibilityStatusEnum', default: draft } # On create, often defaults to draft

official\_website\_url: { type: string, nullable: true, format: url }

logo\_media\_id: { type: string, format: "uuid", nullable: true }

banner\_media\_id: { type: string, format: "uuid", nullable: true }

contact\_organization\_url: { type: string, nullable: true, format: url }

is\_featured: { type: boolean, default: false }

Trail: # Read Schema

type: object

properties:

id: { type: integer, format: int64, readOnly: true }

slug: { type: string, readOnly: true }

trail\_short\_code: { type: string, nullable: true }

# Localized convenience fields + translations objects

name: { type: string, description: "Localized name." }

name\_translations: { type: object, additionalProperties: { type: string } }

alternate\_names: { type: array, items: {type: string}, nullable: true, description: "Localized alternate names." } # API provides this array localized.

alternate\_names\_translations: { type: object, additionalProperties: { type: array, items: {type: string} } , description: "Object mapping lang\_code to array of translated alternate names."}

short\_description: { type: string, nullable: true, description: "Localized short description." }

short\_description\_translations: { type: object, additionalProperties: { type: string } }

# ... (Repeat for all translatable fields from TrailBase, providing 'field' and 'field\_translations')

full\_description: { type: string, nullable: true }

full\_description\_translations: { type: object, additionalProperties: { type: string } }

historical\_significance: { type: string, nullable: true }

historical\_significance\_translations: { type: object, additionalProperties: { type: string } }

# ... (and so on for all translatable fields defined in trails spec)

# Non-translatable fields

estimated\_total\_distance\_km: { type: number, format: float, nullable: true }

estimated\_total\_duration\_days: { type: string, nullable: true }

overall\_difficulty: { $ref: '#/components/schemas/TrailDifficultyEnum', nullable: true }

operational\_status: { $ref: '#/components/schemas/TrailOperationalStatusEnum' }

content\_visibility\_status: { $ref: '#/components/schemas/ContentVisibilityStatusEnum', readOnly: true }

official\_website\_url: { type: string, nullable: true, format: url }

logo\_media\_id: { type: string, format: "uuid", nullable: true }

logo\_media: { $ref: '#/components/schemas/MediaItemSummary', nullable: true, readOnly: true } # Embedded

banner\_media\_id: { type: string, format: "uuid", nullable: true }

banner\_media: { $ref: '#/components/schemas/MediaItemSummary', nullable: true, readOnly: true } # Embedded

is\_featured: { type: boolean }

# Audit fields

created\_at: { type: string, format: "date-time", readOnly: true }

updated\_at: { type: string, format: "date-time", readOnly: true }

deleted\_at: { type: string, format: "date-time", nullable: true, readOnly: true }

TrailCreate:

allOf:

- $ref: '#/components/schemas/TrailBase'

- type: object

required: [name, slug, operational\_status] # English name for creation

TrailUpdate: # For PATCH, all fields are optional

allOf:

- $ref: '#/components/schemas/TrailBase' # All fields from base become optional

- type: object

properties: # Example of making a base field optional

name: { type: string, maxLength: 255, nullable: true}

slug: { type: string, maxLength: 255, pattern: "^[a-z0-9]+(?:-[a-z0-9]+)\*$", nullable: true }

PaginatedTrailList:

type: object

properties:

data: { type: array, items: { $ref: '#/components/schemas/Trail' } }

pagination: { $ref: '#/components/schemas/PaginationInfo' }

# --- Other Module 2 Schemas (Routes, Segments, Associations) would follow similar pattern ---

# For brevity, I'll outline one more major entity (Route) and one association.

RouteBase:

type: object

properties:

trail\_id: { type: integer, format: int64 }

name: { type: string, maxLength: 255, description: "Name of the route in English." } # English for create/update

slug: { type: string, maxLength: 255, pattern: "^[a-z0-9]+(?:-[a-z0-9]+)\*$" }

# ... other fields similar to routes table spec, English for translatable ones

operational\_status: { $ref: '#/components/schemas/TrailOperationalStatusEnum', default: information\_unavailable }

content\_visibility\_status: { $ref: '#/components/schemas/ContentVisibilityStatusEnum', default: draft }

overall\_gpx\_media\_id: { type: string, format: "uuid", nullable: true }

Route: # Read Schema

type: object

properties:

id: { type: integer, format: int64, readOnly: true }

trail\_id: { type: integer, format: int64, readOnly: true }

name: { type: string, description: "Localized name." }

name\_translations: { type: object, additionalProperties: { type: string } }

slug: { type: string, readOnly: true }

# ... other fields including localized versions and \_translations objects

total\_distance\_km: { type: number, format: float, nullable: true, readOnly: true }

estimated\_total\_elevation\_gain\_meters: { type: integer, nullable: true, readOnly: true }

operational\_status: { $ref: '#/components/schemas/TrailOperationalStatusEnum' }

content\_visibility\_status: { $ref: '#/components/schemas/ContentVisibilityStatusEnum', readOnly: true }

overall\_gpx\_media: { $ref: '#/components/schemas/MediaItemSummary', nullable: true, readOnly: true }

created\_at: { type: string, format: "date-time", readOnly: true }

updated\_at: { type: string, format: "date-time", readOnly: true }

deleted\_at: { type: string, format: "date-time", nullable: true, readOnly: true }

RouteCreate:

allOf:

- $ref: '#/components/schemas/RouteBase'

- type: object

required: [trail\_id, name, slug]

PaginatedRouteList:

type: object

properties:

data: { type: array, items: { $ref: '#/components/schemas/Route' } }

pagination: { $ref: '#/components/schemas/PaginationInfo' }

Segment: # Read Schema

type: object

properties:

id: { type: integer, format: int64, readOnly: true }

name: { type: string, description: "Localized name." }

name\_translations: { type: object, additionalProperties: { type: string } }

# ... other fields, including path\_geom (GeoJSON), distance\_km etc.

# ... other translatable fields like short\_description with their \_translations objects

gpx\_media: { $ref: '#/components/schemas/MediaItemSummary', nullable: true, readOnly: true }

PaginatedSegmentList:

type: object

properties:

data: { type: array, items: { $ref: '#/components/schemas/Segment' } }

pagination: { $ref: '#/components/schemas/PaginationInfo' }

# Association for creating a link (example)

TrailTerrainTypeLinkCreate:

type: object

required: [terrain\_type\_id]

properties:

terrain\_type\_id: { type: integer, description: "ID of the terrain\_types\_master to link." }

TrailTerrainTypeLink: # Response for the link

type: object

properties:

trail\_id: {type: integer, format: int64}

terrain\_type\_id: {type: integer}

# Optionally embed TerrainTypeMaster details here

terrain\_type: { $ref: '#/components/schemas/TerrainTypeMaster' }

created\_at: { type: string, format: "date-time", readOnly: true }

# ... audit fields

security:

- bearerAuth: []

paths:

# Master Data

/terrain\_types\_master:

get:

tags: [Master Data (Trail Hierarchy)]

summary: List Terrain Types

operationId: listTerrainTypes

parameters:

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/PageParam'

- $ref: '#/components/parameters/PageSizeParam'

- name: is\_active # Specific filter for this endpoint

in: query

schema: { type: boolean, default: true }

responses:

'200':

description: A list of terrain types.

content:

application/json:

schema: { $ref: '#/components/schemas/PaginatedTerrainTypeMasterList' }

# SQL View: A view joining terrain\_types\_master with translations would be beneficial.

/usage\_types\_master:

get:

tags: [Master Data (Trail Hierarchy)]

summary: List Usage Types

operationId: listUsageTypes

parameters:

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/PageParam'

- $ref: '#/components/parameters/PageSizeParam'

- name: is\_active

in: query

schema: { type: boolean, default: true }

responses:

'200':

description: A list of usage types.

content:

application/json:

schema: { $ref: '#/components/schemas/PaginatedUsageTypeMasterList' }

# SQL View: A view joining usage\_types\_master with translations.

# Trails

/trails:

get:

tags: [Trails]

summary: List Trails

operationId: listTrails

# SQL View: Utilizes public.v\_trails\_detailed\_localized or similar, transformed by API.

# Index: On filterable columns like operational\_status, overall\_difficulty.

parameters:

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/PageParam'

- $ref: '#/components/parameters/PageSizeParam'

- $ref: '#/components/parameters/SelectParam'

- name: operational\_status

in: query

schema: { $ref: '#/components/schemas/TrailOperationalStatusEnum' }

responses:

'200':

description: A paginated list of trails.

content: { application/json: { schema: { $ref: '#/components/schemas/PaginatedTrailList' } } }

post:

tags: [Trails]

summary: Create Trail

operationId: createTrail

security: [{ bearerAuth: [] }]

requestBody:

required: true

content: { application/json: { schema: { $ref: '#/components/schemas/TrailCreate' } } }

responses:

'201':

description: Trail created.

content: { application/json: { schema: { $ref: '#/components/schemas/Trail' } } }

'400': { $ref: '#/components/responses/BadRequestError' }

'401': { $ref: '#/components/responses/UnauthorizedError' }

/trails/{trail\_id}:

get:

tags: [Trails]

summary: Get Trail by ID

operationId: getTrailById

# SQL View: Utilizes public.v\_trails\_detailed\_localized, transformed by API.

parameters:

- $ref: '#/components/parameters/TrailIdParam'

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/SelectParam'

responses:

'200':

description: Detailed information about the trail.

content: { application/json: { schema: { $ref: '#/components/schemas/Trail' } } }

'404': { $ref: '#/components/responses/NotFoundError' }

patch:

tags: [Trails]

summary: Update Trail

operationId: updateTrailById # Renamed for consistency

security: [{ bearerAuth: [] }]

parameters:

- $ref: '#/components/parameters/TrailIdParam'

requestBody:

required: true

content: { application/json: { schema: { $ref: '#/components/schemas/TrailUpdate' } } } # Use TrailUpdate

responses:

'200':

description: Trail updated.

content: { application/json: { schema: { $ref: '#/components/schemas/Trail' } } }

'404': { $ref: '#/components/responses/NotFoundError' }

'401': { $ref: '#/components/responses/UnauthorizedError' }

/trails/{trail\_id}/terrain\_types: # POST example for M-M

post:

tags: [Trail Associations]

summary: Add Terrain Type to Trail

operationId: addTerrainTypeToTrail

security: [{ bearerAuth: [] }]

parameters:

- $ref: '#/components/parameters/TrailIdParam'

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/TrailTerrainTypeLinkCreate'

responses:

'201':

description: Terrain type linked to trail.

content: { application/json: { schema: { $ref: '#/components/schemas/TrailTerrainTypeLink' } } }

'404': { $ref: '#/components/responses/NotFoundError' } # Trail or TerrainType not found

'409': { $ref: '#/components/responses/ConflictError' } # Already linked

# Routes

/routes:

get:

tags: [Routes]

summary: List Routes

operationId: listRoutes

# SQL View: Utilizes public.routes\_summary\_view or public.v\_routes\_detailed\_localized for richer list.

parameters:

- name: trail\_id

in: query

schema: { type: integer, format: int64 }

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/PageParam'

- $ref: '#/components/parameters/PageSizeParam'

- $ref: '#/components/parameters/SelectParam'

responses:

'200':

description: A paginated list of routes.

content: { application/json: { schema: { $ref: '#/components/schemas/PaginatedRouteList' } } }

post:

tags: [Routes]

summary: Create Route

operationId: createRoute

security: [{ bearerAuth: [] }]

requestBody:

required: true

content: { application/json: { schema: { $ref: '#/components/schemas/RouteCreate' } } }

responses:

'201':

description: Route created.

content: { application/json: { schema: { $ref: '#/components/schemas/Route' } } }

'400': { $ref: '#/components/responses/BadRequestError' }

/routes/{route\_id}:

get:

tags: [Routes]

summary: Get Route by ID

operationId: getRouteById

# SQL View: Utilizes public.v\_routes\_detailed\_localized, transformed by API.

parameters:

- $ref: '#/components/parameters/RouteIdParam'

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/SelectParam'

responses:

'200':

description: Detailed information about the route.

content: { application/json: { schema: { $ref: '#/components/schemas/Route' } } }

'404': { $ref: '#/components/responses/NotFoundError' }

/routes/{route\_id}/segments: # POST to associate segments with a route

post:

tags: [Routes, Segments] # Belongs to both conceptually

summary: Add ordered Segment to Route

operationId: addSegmentToRoute

security: [{ bearerAuth: [] }]

parameters:

- $ref: '#/components/parameters/RouteIdParam'

requestBody:

description: Specify segment ID and its order in this route.

required: true

content:

application/json:

schema:

type: object

required: [segment\_id, order\_in\_route]

properties:

segment\_id: { type: integer, format: int64 }

order\_in\_route: { type: integer, minimum: 1 }

contextual\_notes\_en: { type: string, nullable: true, description: "Contextual notes for this segment within this route, in English."}

responses:

'201':

description: Segment association created.

# Response should be the created route\_segments link, ideally embedding segment info

content:

application/json:

schema:

# Simplified response for brevity; ideally it's the created RouteSegment entry

type: object

properties:

message: { type: string, example: "Segment added to route." }

route\_segment\_id: { type: integer, format: int64 }

'400': { $ref: '#/components/responses/BadRequestError' }

'404': { $ref: '#/components/responses/NotFoundError' } # Route or segment not found

'409': { $ref: '#/components/responses/ConflictError' } # e.g. order conflict

# Segments

/segments:

get:

tags: [Segments]

summary: List Segments

operationId: listSegments

# SQL View: Utilizes public.segments\_summary\_view or public.v\_segments\_detailed\_localized.

parameters:

- name: route\_id # Common filter

in: query

schema: { type: integer, format: int64 }

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/PageParam'

- $ref: '#/components/parameters/PageSizeParam'

- $ref: '#/components/parameters/SelectParam'

responses:

'200':

description: A paginated list of segments.

content: { application/json: { schema: { $ref: '#/components/schemas/PaginatedSegmentList' } } }

post:

tags: [Segments]

summary: Create Segment

operationId: createSegment

security: [{ bearerAuth: [] }]

requestBody:

required: true

content:

application/json:

schema:

# Define SegmentCreate schema (based on SegmentBase)

type: object # Placeholder, define fully based on SegmentBase

required: [name, start\_waypoint\_id, end\_waypoint\_id, path\_geom]

properties:

name: { type: string, description: "Name in English" }

start\_waypoint\_id: { type: integer, format: int64 }

end\_waypoint\_id: { type: integer, format: int64 }

path\_geom: { type: object, description: "GeoJSON LineStringZ" }

# ... other writable fields from SegmentBase

responses:

'201':

description: Segment created.

content: { application/json: { schema: { $ref: '#/components/schemas/Segment' } } }

'400': { $ref: '#/components/responses/BadRequestError' }

/segments/{segment\_id}:

get:

tags: [Segments]

summary: Get Segment by ID

operationId: getSegmentById

# SQL View: Utilizes public.v\_segments\_detailed\_localized, transformed by API.

parameters:

- $ref: '#/components/parameters/SegmentIdParam'

- $ref: '#/components/parameters/LangParam'

- $ref: '#/components/parameters/SelectParam'

responses:

'200':

description: Detailed information about the segment.

content: { application/json: { schema: { $ref: '#/components/schemas/Segment' } } }

'404': { $ref: '#/components/responses/NotFoundError' }

# Standard shared responses

responses:

NotFoundError:

description: The specified resource was not found.

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

UnauthorizedError:

description: Authentication failed or user does not have credentials to access the resource.

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

ForbiddenError:

description: User is authenticated but does not have permissions to perform this action on the resource.

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

BadRequestError:

description: The request was malformed or contained invalid parameters.

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

UnprocessableEntityError:

description: The request was well-formed but contained semantic errors (e.g., validation failed).

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

ConflictError:

description: The request could not be completed due to a conflict with the current state of the target resource.

content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }

```

### Quick-Start Examples

1\. List Trails (filtered by difficulty 'moderate', requesting Italian, page 1, 2 items per page)

- Request:

Bash

```

curl -X GET

'https://<YOUR\_SUPABASE\_PROJECT\_REF>.supabase.co/rest/v1/trails?overall\_difficulty=eq.moderate&page=1&page\_size=2&lang=it&select=id,name,slug,overall\_difficulty,short\_description,name\_translations'

-H "APIKey: <YOUR\_SUPABASE\_ANON\_KEY>"

-H "Authorization: Bearer <YOUR\_VALID\_JWT>"

-H "Accept: application/json"

```

- Sample Response (200 OK):

JSON

```

{

"data": [

{

"id": 10,

"name": "Sentiero delle Foreste Casentinesi (Tratto Moderato)", // Localized name

"name\_translations": {

"it": "Sentiero delle Foreste Casentinesi (Tratto Moderato)",

"en": "Casentinesi Forests Trail (Moderate Section)"

},

"slug": "foreste-casentinesi-moderato",

"overall\_difficulty": "moderate",

"short\_description": "Un percorso affascinante attraverso antiche foreste, con pendenze gestibili." // Localized

}

],

"pagination": {

"current\_page": 1,

"page\_size": 2,

"total\_items": 2, // Assuming only 2 moderate trails for this example

"total\_pages": 1

}

}

```

2\. Get Segment Details (Segment ID 101, default language English)

- Request:

Bash

```

curl -X GET

'https://<YOUR\_SUPABASE\_PROJECT\_REF>.supabase.co/rest/v1/segments?id=eq.101&select=id,name,distance\_km,name\_translations,gpx\_media:media(storage\_object\_path\_original)'

-H "APIKey: <YOUR\_SUPABASE\_ANON\_KEY>"

-H "Authorization: Bearer <YOUR\_VALID\_JWT>"

-H "Accept: application/json"

```

- Sample Response (200 OK):

JSON

```

[

{

"id": 101,

"name": "La Verna Ascent - Forest Path", // English default

"name\_translations": {

"en": "La Verna Ascent - Forest Path",

"it": "Salita La Verna - Sentiero Forestale"

},

"distance\_km": 5.5,

"gpx\_media": {

"storage\_object\_path\_original": "gpx\_files/segment\_101\_la\_verna\_ascent.gpx"

}

}

]

```

3\. Create a New Route for Trail ID 1 (English content)

- Request:

Bash

```

curl -X POST

'https://<YOUR\_SUPABASE\_PROJECT\_REF>.supabase.co/rest/v1/routes'

-H "APIKey: <YOUR\_SUPABASE\_ANON\_KEY>"

-H "Authorization: Bearer <YOUR\_VALID\_JWT>"

-H "Content-Type: application/json"

-H "Accept: application/json"

-H "Prefer: return=representation"

-d '{

"trail\_id": 1,

"name": "Alternative Route via Sanctuary",

"slug": "alt-route-sanctuary-1",

"short\_description": "A scenic alternative path visiting the upper sanctuary.",

"route\_category": "official\_variant",

"operational\_status": "fully\_operational",

"content\_visibility\_status": "draft",

"route\_difficulty": "moderate"

}'

```

- Sample Response (201 Created):

JSON

```

{

"id": 25, // New route\_id

"trail\_id": 1,

"name": "Alternative Route via Sanctuary", // English, as provided

"name\_translations": {

"en": "Alternative Route via Sanctuary" // Backend would create this from the 'name' field

},

"slug": "alt-route-sanctuary-1",

"short\_description": "A scenic alternative path visiting the upper sanctuary.",

"short\_description\_translations": {

"en": "A scenic alternative path visiting the upper sanctuary."

},

"route\_category": "official\_variant",

"operational\_status": "fully\_operational",

"content\_visibility\_status": "draft",

"route\_difficulty": "moderate",

"total\_distance\_km": null, // Initially null, updated by segment associations

"estimated\_total\_elevation\_gain\_meters": null, // Initially null

"created\_at": "2025-05-18T20:00:00Z",

"updated\_at": "2025-05-18T20:00:00Z",

"deleted\_at": null

// other fields would have their defaults or null

}

```

### Implementation Notes

- SQL Views & Data Preparation for API:

- The database views like `public.v\_trails\_detailed\_localized`, `public.v\_routes\_detailed\_localized`, and `public.v\_segments\_detailed\_localized` (documented in `2.0.2 Views.docx`) are highly valuable. The API backend will query these (or use similar logic) based on the `id` and `lang` parameter.

- The API layer must then transform the result from these views to fit the specified API response structure:

- The main localized field (e.g., `name`) should be populated with the value corresponding to the `lang` parameter (or English if `lang` is not provided/translation missing).

- The `\*\_translations` object (e.g., `name\_translations`) needs to be constructed by querying `public.translations` for all available translations for that specific entity instance and field, including English from the base table column.

- For list endpoints, summary views (`public.routes\_summary\_view`, `public.segments\_summary\_view`) are useful. The API layer would similarly need to augment these with the `\*\_translations` objects.

- Indexes:

- The indexes defined in the individual table specification documents (e.g., on foreign keys, `deleted\_at`, `content\_visibility\_status`, GIST on `segments.path\_geom`) are crucial.

- Ensure `public.translations` has a high-performance composite index (e.g., on `table\_identifier, column\_identifier, row\_foreign\_key, language\_code`) to support efficient fetching for the `\*\_translations` objects.

- Materialized Views:

- If constructing the `\*\_translations` objects for every request on detailed or list endpoints proves to be a performance issue, consider materializing the localized views (`v\_\*\_detailed\_localized`) or creating new materialized views that pre-aggregate the `\*\_translations` objects. This would require a refresh strategy.

- Schema Feedback & `\_en` Suffix Review:

- 🟢 Schema OK. The database schema for Module 2 is well-defined. The decision to store English directly in primary table columns and other languages in `public.translations` is standard and sound.

- Review of `\_en` in Module 2 Documents:

- Database Table Columns: All Module 2 table specifications (`2.1 Trails.docx` through `2.9 trail\_usage\_types.docx`) correctly define columns like `name`, `short\_description` etc., to store the English (base language) text directly, without an `\_en` suffix in the column name itself. This is consistent with the overall DB design.

- Database View Output Columns (`2.0.2 Views.docx`):

- `public.routes\_summary\_view`: Uses `route\_name\_en`, `trail\_name\_en`, etc.

- `public.segments\_summary\_view`: Uses `segment\_name\_en`, etc.

- `public.v\_trails\_detailed\_localized`: The DDL shows it selecting the base English values with aliases like `t.best\_seasons\_to\_walk AS best\_seasons\_to\_walk\_en`.

- `public.v\_routes\_detailed\_localized`: Uses `trail\_name\_en`, `start\_town\_name\_en`, etc.

- `public.v\_segments\_detailed\_localized`: Uses `segment\_name\_en`, `start\_waypoint\_name\_en`, etc. This usage of `\_en` in the \*output column names of SQL views\* is not incorrect; it's a deliberate choice within the view definition to distinguish the base English text fetched from the main tables, especially when the view's purpose is to aid in localization. The API layer consuming these views would then map these `\*\_en` view columns to the appropriate fields in the final API JSON structure (e.g., to the `en` key within the `fieldname\_translations` object and to the main `fieldname` if English is the requested/fallback language).

- Therefore, no \*incorrect\* usage of `\_en` that contradicts the database design was found within the Module 2 specification documents themselves. The previous API model using `\_en` in the JSON output was based on my interpretation of an earlier specific instruction from you for that task, which you've now clarified.

- Pagination: The API gateway or backend logic needs to translate `page` and `page\_size` parameters to `LIMIT` and `OFFSET` if using PostgREST directly and construct the `pagination` object in the response.

- Write Operations: For `POST` operations creating new entities with translatable fields (e.g., creating a new Trail), the request body will contain the English version (e.g., `name: "New Trail Name"`). The backend logic should save this to the main table's `name` column and also create the corresponding entry in the `public.translations` table for English (`language\_code: 'en'`). Translations to other languages would be managed via separate processes or subsequent update operations.