https://gemini.google.com/u/1/app/951c86a3d09aaadb

### OpenAPI 3.1 Spec (for Dynamic Conditions Module)

YAML

```

openapi: 3.1.0

info:

title: Dynamic Conditions Module API

version: v1.0

description: API for managing and retrieving trail segment warnings and related lookup data.

servers:

- url: https://api.pilgrimage-platform.example.com/v1

description: Production Server

- url: http://localhost:3000/v1

description: Development Server

security:

- bearerAuth: [] # Global security requirement, can be overridden at operation level

paths:

/segments/{segment\_id}/warnings:

get:

summary: List Active Warnings for a Segment

operationId: listSegmentWarnings

tags:

- Segment Warnings

parameters:

- name: segment\_id

in: path

required: true

description: ID of the trail segment.

schema:

type: integer

format: int64

- name: lang

in: query

description: ISO language code for response localization (e.g., 'it', 'de'). Affects translatable fields.

schema:

type: string

example: "it"

- name: page

in: query

description: Page number for pagination.

schema:

type: integer

default: 1

minimum: 1

- name: page\_size

in: query

description: Number of items per page.

schema:

type: integer

default: 20

minimum: 1

maximum: 100

responses:

'200':

description: A paginated list of active and published warnings for the segment.

content:

application/json:

schema:

type: object

properties:

data:

type: array

items:

$ref: '#/components/schemas/SegmentWarning\_View\_Output'

pagination:

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

# This endpoint likely uses the public\_active\_segment\_warnings\_view for efficiency.

'404':

description: Segment not found.

content:

application/json:

schema:

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

default:

description: Unexpected error.

content:

application/json:

schema:

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

post:

summary: Create a New Warning for a Segment

operationId: createSegmentWarning

tags:

- Segment Warnings

security:

- bearerAuth: ['manage:warnings'] # Example scope

parameters:

- name: segment\_id

in: path

required: true

description: ID of the trail segment for which to create the warning.

schema:

type: integer

format: int64

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/SegmentWarning\_Input'

responses:

'201':

description: Segment warning created successfully.

content:

application/json:

schema:

$ref: '#/components/schemas/SegmentWarning\_Detail\_Output'

'400':

description: Invalid input data.

content:

application/json:

schema:

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

'403':

description: Forbidden - user cannot create warnings for this segment.

content:

application/json:

schema:

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

'404':

description: Segment not found.

content:

application/json:

schema:

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

default:

description: Unexpected error.

content:

application/json:

schema:

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

/warnings/search/geo:

get:

summary: Search Active Warnings by Geographical Area

operationId: searchGeoWarnings

tags:

- Segment Warnings

parameters:

- name: bbox

in: query

required: true

description: Bounding box coordinates as 'minLng,minLat,maxLng,maxLat'.

schema:

type: string

example: "12.0,43.0,12.5,43.5"

- name: lang

in: query

description: ISO language code for response localization.

schema:

type: string

- name: min\_severity\_code

in: query

description: Filter by minimum warning severity code (inclusive, based on sort\_order).

schema:

type: string

example: "caution\_advised"

- name: page

in: query

description: Page number.

schema:

type: integer

default: 1

- name: page\_size

in: query

description: Items per page.

schema:

type: integer

default: 50

responses:

'200':

description: A paginated list of active warnings within the geographical area.

content:

application/json:

schema:

type: object

properties:

data:

type: array

items:

$ref: '#/components/schemas/SegmentWarning\_View\_Output'

pagination:

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

# This endpoint uses the public\_active\_segment\_warnings\_view and a GIST index on location\_on\_segment\_geom.

default:

description: Unexpected error.

content:

application/json:

schema:

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

/warnings/{warning\_id}:

get:

summary: Get Segment Warning Details

operationId: getSegmentWarningById

tags:

- Segment Warnings

parameters:

- name: warning\_id

in: path

required: true

description: ID of the segment warning.

schema:

type: integer

format: int64

- name: lang

in: query

description: ISO language code for response localization.

schema:

type: string

responses:

'200':

description: Detailed information about the segment warning.

content:

application/json:

schema:

$ref: '#/components/schemas/SegmentWarning\_Detail\_Output'

'404':

description: Segment warning not found.

content:

application/json:

schema:

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

default:

description: Unexpected error.

content:

application/json:

schema:

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

patch:

summary: Update a Segment Warning

operationId: updateSegmentWarning

tags:

- Segment Warnings

security:

- bearerAuth: ['manage:warnings'] # Example scope

parameters:

- name: warning\_id

in: path

required: true

description: ID of the segment warning to update.

schema:

type: integer

format: int64

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/SegmentWarning\_Update\_Input' # Similar to Input, but fields optional

responses:

'200':

description: Segment warning updated successfully.

content:

application/json:

schema:

$ref: '#/components/schemas/SegmentWarning\_Detail\_Output'

'400':

description: Invalid input data.

'403':

description: Forbidden.

'404':

description: Segment warning not found.

default:

description: Unexpected error.

# --- Lookup Table Endpoints (Example for Warning Types) ---

/warnings/types:

get:

summary: List Warning Types

operationId: listWarningTypes

tags:

- Warning Configuration

parameters:

- name: lang

in: query

description: ISO language code for response localization.

schema:

type: string

- name: page

in: query

schema:

type: integer

default: 1

- name: page\_size

in: query

schema:

type: integer

default: 100 # Typically all types are returned

- name: is\_active

in: query

description: Filter by active status.

schema:

type: boolean

default: true

responses:

'200':

description: A list of warning types.

content:

application/json:

schema:

type: object

properties:

data:

type: array

items:

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

pagination:

$ref: '#/components/schemas/Pagination' # Pagination might be overkill if list is always short

default:

description: Unexpected error.

/warnings/types/{type\_code}:

get:

summary: Get Warning Type by Code

operationId: getWarningTypeByCode

tags:

- Warning Configuration

parameters:

- name: type\_code

in: path

required: true

description: Code of the warning type.

schema:

type: string

- name: lang

in: query

description: ISO language code for response localization.

schema:

type: string

responses:

'200':

description: Detailed information about the warning type.

content:

application/json:

schema:

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

'404':

description: Warning type not found.

default:

description: Unexpected error.

# Similar GET endpoints would exist for /warnings/severities and /warnings/sources, /warnings/workflow-statuses

components:

schemas:

Pagination:

type: object

properties:

page:

type: integer

description: Current page number.

example: 1

page\_size:

type: integer

description: Number of items per page.

example: 20

total\_items:

type: integer

description: Total number of items available.

example: 152

total\_pages:

type: integer

description: Total number of pages.

example: 8

ErrorResponse:

type: object

properties:

code:

type: string

description: Machine-readable error code.

example: "resource\_not\_found"

message:

type: string

description: Human-readable error message.

example: "The requested segment warning was not found."

detail:

type: string

nullable: true

description: Optional additional details about the error.

example: "Warning ID 12345 does not exist."

# --- Schemas for Dynamic Conditions Module Tables ---

WarningTypeMaster:

type: object

properties:

id:

type: integer

readOnly: true

code:

type: string

example: "trail\_damage\_general"

display\_name: # Base language text

type: string

example: "General Trail Damage"

description: # Base language text

type: string

nullable: true

example: "General damage to the trail surface or structures."

icon\_identifier:

type: string

nullable: true

example: "fas fa-tools"

sort\_order:

type: integer

default: 0

is\_active:

type: boolean

default: true

notes:

type: string

nullable: true

created\_at:

type: string

format: date-time

readOnly: true

updated\_at:

type: string

format: date-time

readOnly: true

created\_by\_profile\_id:

type: string

format: uuid

nullable: true

readOnly: true

updated\_by\_profile\_id:

type: string

format: uuid

nullable: true

readOnly: true

translations: # Populated by API backend based on 'lang' query param

type: object

additionalProperties:

type: string

example:

it: "Danno Generale al Sentiero"

de: "Allgemeiner Wegschaden"

WarningSeverityMaster:

# Similar to WarningTypeMaster, with ui\_color\_hex

type: object

properties:

id: { type: integer, readOnly: true }

code: { type: string, example: "caution\_advised" }

display\_name: { type: string, example: "Caution Advised" } # Base language

description: { type: string, nullable: true, example: "Proceed with caution." } # Base language

ui\_color\_hex: { type: string, nullable: true, example: "#F1C40F" }

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

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

notes: { type: string, nullable: true }

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

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

created\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

updated\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

translations:

type: object

additionalProperties: { type: string }

WarningSourceTypeMaster:

# Similar to WarningTypeMaster, with default\_trust\_level

type: object

properties:

id: { type: integer, readOnly: true }

code: { type: string, example: "official\_authority" }

display\_name: { type: string, example: "Official Authority" } # Base language

description: { type: string, nullable: true, example: "Warnings from CAI, Park Authorities, etc." } # Base language

default\_trust\_level: { type: integer, nullable: true, minimum: 1, maximum: 5 }

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

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

notes: { type: string, nullable: true }

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

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

created\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

updated\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

translations:

type: object

additionalProperties: { type: string }

WorkflowStatusMaster:

type: object

properties:

code: { type: string, example: "published" }

display\_name: { type: string, example: "Published" } # Base language

description: { type: string, nullable: true, example: "Content is live." } # Base language

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

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

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

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

created\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

updated\_by\_profile\_id: { type: string, format: uuid, nullable: true, readOnly: true }

translations:

type: object

additionalProperties: { type: string }

SegmentWarning\_Base: # Common fields for input/output

type: object

properties:

title: # Base language text

type: string

maxLength: 255

example: "Minor Landslide near Oak Tree"

description\_message: # Base language text

type: string

example: "A small landslide has occurred on the path approximately 2.5km from the start of the segment, near the old oak tree. Path is still passable with caution."

location\_on\_segment\_description: # Base language text

type: string

nullable: true

example: "Around km 2.5, just after the anbandoned shack"

location\_on\_segment\_km\_approx:

type: number

format: float

nullable: true

minimum: 0

example: 2.5

location\_on\_segment\_geom: # GeoJSON PointZ or similar

type: object

nullable: true

properties:

type: { type: string, enum: ["Point"] }

coordinates: { type: array, items: { type: number }, minItems: 2, maxItems: 3 }

example: { "type": "Point", "coordinates": [12.345, 43.210, 150.0] }

detour\_information\_url:

type: string

format: url

nullable: true

example: "https://example.com/detours/frana-oak-tree"

detour\_description\_notes: # Base language text

type: string

nullable: true

example: "Alternative route is marked with yellow signs. Adds approximately 30 minutes."

warning\_type\_id:

type: integer

description: FK to warning\_types\_master.id

warning\_severity\_id:

type: integer

description: FK to warning\_severities\_master.id

warning\_source\_type\_id:

type: integer

nullable: true

description: FK to warning\_source\_types\_master.id

reported\_by\_source\_detail:

type: string

maxLength: 500

nullable: true

example: "Reported by local hiker Giovanni Rossi"

date\_warning\_reported:

type: string

format: date-time

example: "2025-05-18T14:30:00Z"

date\_warning\_effective\_from:

type: string

format: date-time

nullable: true

date\_warning\_expected\_resolution:

type: string

format: date-time

nullable: true

date\_warning\_resolved\_or\_expired:

type: string

format: date-time

nullable: true

admin\_verification\_notes:

type: string

nullable: true

primary\_image\_media\_id:

type: string

format: uuid

nullable: true

workflow\_status\_code: # Code from workflow\_statuses\_master

type: string

nullable: true

example: "pending\_review"

required:

- title

- description\_message

- warning\_type\_id

- warning\_severity\_id

- date\_warning\_reported

# created\_by\_user\_id is usually set by system from auth context on POST

SegmentWarning\_Input:

allOf:

- $ref: '#/components/schemas/SegmentWarning\_Base'

# No additional properties for input beyond base, created\_by\_user\_id is system-set

SegmentWarning\_Update\_Input: # For PATCH, all fields are optional

type: object

properties: # Mirror SegmentWarning\_Base but all nullable/optional

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

description\_message: { type: string, nullable: true }

location\_on\_segment\_description: { type: string, nullable: true }

location\_on\_segment\_km\_approx: { type: number, format: float, nullable: true, minimum: 0 }

location\_on\_segment\_geom: { type: object, nullable: true, properties: { type: { type: string, enum: ["Point"] }, coordinates: { type: array, items: { type: number }, minItems: 2, maxItems: 3 } } }

detour\_information\_url: { type: string, format: url, nullable: true }

detour\_description\_notes: { type: string, nullable: true }

warning\_type\_id: { type: integer, nullable: true }

warning\_severity\_id: { type: integer, nullable: true }

warning\_source\_type\_id: { type: integer, nullable: true }

reported\_by\_source\_detail: { type: string, maxLength: 500, nullable: true }

date\_warning\_effective\_from: { type: string, format: date-time, nullable: true }

date\_warning\_expected\_resolution: { type: string, format: date-time, nullable: true }

date\_warning\_resolved\_or\_expired: { type: string, format: date-time, nullable: true }

admin\_verification\_notes: { type: string, nullable: true }

primary\_image\_media\_id: { type: string, format: uuid, nullable: true }

workflow\_status\_code: { type: string, nullable: true }

SegmentWarning\_View\_Output: # Schema for items from public\_active\_segment\_warnings\_view

type: object

properties:

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

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

title: { type: string } # Base language

description\_message: { type: string } # Base language

location\_on\_segment\_description: { type: string, nullable: true } # Base language

location\_on\_segment\_km\_approx: { type: number, format: float, nullable: true }

location\_on\_segment\_geom: { type: object, nullable: true } # GeoJSON

detour\_information\_url: { type: string, format: url, nullable: true }

detour\_description\_notes: { type: string, nullable: true } # Base language

date\_warning\_reported: { type: string, format: date-time }

date\_warning\_effective\_from: { type: string, format: date-time, nullable: true }

date\_warning\_expected\_resolution: { type: string, format: date-time, nullable: true }

warning\_type:

type: object

properties:

code: { type: string }

name: { type: string } # Base language

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

warning\_severity:

type: object

properties:

code: { type: string }

name: { type: string } # Base language

ui\_color\_hex: { type: string, nullable: true }

warning\_source\_type\_name: { type: string, nullable: true } # Base language

primary\_image\_media\_id: { type: string, format: uuid, nullable: true }

# primary\_image\_variants: { type: object, nullable: true } # If API provides this structure

translations: # Populated by API backend based on 'lang' query param

type: object

description: "Key-value pairs where key is language code (e.g., 'it') and value is translated string for fields like title, description\_message etc. Specific fields depend on API implementation."

additionalProperties:

type: string # Or object if translating multiple fields

example:

title: # This example implies 'title' is the key for its translations

it: "Titolo Frana"

description\_message:

it: "Descrizione frana"

# Note: The translations object structure needs careful API design.

# Alternative: top-level translated fields if 'lang' param is used to return one language.

SegmentWarning\_Detail\_Output: # Schema for GET /warnings/{id}

allOf:

- $ref: '#/components/schemas/SegmentWarning\_Base'

properties:

id:

type: integer

format: int64

readOnly: true

is\_currently\_active: # Read-only, computed by DB

type: boolean

readOnly: true

created\_at:

type: string

format: date-time

readOnly: true

updated\_at:

type: string

format: date-time

readOnly: true

created\_by\_user\_id: # Should be profile ID if aligning with other audit

type: string

format: uuid

readOnly: true

updated\_by\_user\_id:

type: string

format: uuid

nullable: true

readOnly: true

# Expanded details for FKs

warning\_type: { $ref: '#/components/schemas/WarningTypeMaster' }

warning\_severity: { $ref: '#/components/schemas/WarningSeverityMaster' }

warning\_source\_type: { $ref: '#/components/schemas/WarningSourceTypeMaster' , nullable: true }

workflow\_status: { $ref: '#/components/schemas/WorkflowStatusMaster', nullable: true }

# primary\_image: { $ref: '#/components/schemas/MediaItem', nullable: true } # If embedding media details

translations: # Populated by API backend based on 'lang' query param

type: object

additionalProperties:

type: string # Or specific object per field

example:

title:

it: "Titolo Frana"

description\_message:

it: "Descrizione frana"

securitySchemes:

bearerAuth: # Assumed to be defined globally as per prompt

type: http

scheme: bearer

bearerFormat: JWT

description: "Supabase JWT token"

```

&lt;hr/>

### Quick-Start Examples

1. List active warnings for Segment ID 42 (requesting Italian translations)

Bash

```

curl -X GET "https://api.pilgrimage-platform.example.com/v1/segments/42/warnings?lang=it&page\_size=5"\

-H "Authorization: Bearer <YOUR\_SUPABASE\_JWT\_OR\_ANON\_KEY\_IF\_PUBLIC>"\

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

```

Sample Response (conceptual, paginated):

JSON

```

{

"data": [

{

"warning\_id": 101,

"segment\_id": 42,

"title": "Frana Minore",

"description\_message": "Piccola frana sul sentiero...",

"warning\_type": { "code": "hazard\_natural\_landslide", "name": "Pericolo Naturale - Frana", "icon\_identifier": "fas fa-mountain-city" },

"warning\_severity": { "code": "caution\_advised", "name": "Cautela Consigliata", "ui\_color\_hex": "#F1C40F" },

"translations": {

"title": { "it": "Frana Minore" },

"description\_message": { "it": "Piccola frana sul sentiero..." }

}

// ... other fields from SegmentWarning\_View\_Output

}

],

"pagination": {

"page": 1,

"page\_size": 5,

"total\_items": 1,

"total\_pages": 1

}

}

```

2. Get details for Warning ID 101

Bash

```

curl -X GET "https://api.pilgrimage-platform.example.com/v1/warnings/101?lang=en"\

-H "Authorization: Bearer <YOUR\_SUPABASE\_JWT\_OR\_ANON\_KEY\_IF\_PUBLIC>"\

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

```

Sample Response:

JSON

```

{

"id": 101,

"segment\_id": 42,

"title": "Minor Landslide", // Base language or English if lang=en

"description\_message": "Small landslide on path...",

"is\_currently\_active": true,

"workflow\_status\_code": "published",

"warning\_type": {

"id": 1, "code": "hazard\_natural\_landslide", "display\_name": "Landslide Hazard", "is\_active": true /\* ... \*/

},

"warning\_severity": {

"id": 2, "code": "caution\_advised", "display\_name": "Caution Advised", "ui\_color\_hex": "#F1C40F", "is\_active": true /\* ... \*/

},

// ... other fields from SegmentWarning\_Detail\_Output

"translations": {

"title": { "en": "Minor Landslide" }, // Could be redundant if lang=en and title is already English

"description\_message": { "en": "Small landslide on path..." }

}

}

```

3. Create a new warning for Segment ID 77 (Regional Manager action)

Bash

```

curl -X POST "https://api.pilgrimage-platform.example.com/v1/segments/77/warnings"\

-H "Authorization: Bearer <MANAGER\_JWT\_TOKEN>"\

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

-H "Content-Type: application/json"\

-d '{

"title": "Unstable Rocks Reported",

"description\_message": "Hikers reported unstable rocks above the path near the waterfall viewpoint. Assess before proceeding.",

"warning\_type\_id": 1, # Assuming ID for 'hazard\_natural\_rockfall'

"warning\_severity\_id": 2, # Assuming ID for 'caution\_advised'

"date\_warning\_reported": "2025-05-19T11:00:00Z",

"location\_on\_segment\_km\_approx": 3.1,

"workflow\_status\_code": "pending\_review"

}'

```

Sample Response (201 Created):

JSON

```

{

"id": 106, // New warning\_id

"segment\_id": 77,

"title": "Unstable Rocks Reported",

"description\_message": "Hikers reported unstable rocks...",

"created\_by\_user\_id": "<manager\_user\_uuid>", // Set by backend

"workflow\_status\_code": "pending\_review",

// ... other fields from SegmentWarning\_Detail\_Output

"warning\_type": { /\* ... \*/ },

"warning\_severity": { /\* ... \*/ }

}

```

&lt;hr/>

### Implementation Notes

- 🟢 Schema OK: The current database schema for the "Dynamic Conditions Module," including the tables (`segment\_warnings`, `warning\_types\_master`, `warning\_severities\_master`, `warning\_source\_types\_master`, `workflow\_statuses\_master`) and the proposed `public\_active\_segment\_warnings\_view`, appears sufficient to support these core API endpoints.

- Key Database Objects for API Efficiency:

- `public\_active\_segment\_warnings\_view`: This view is critical for the `GET /segments/{segment\_id}/warnings` and `GET /warnings/search/geo` endpoints. It pre-joins and pre-filters data, simplifying API logic and improving readability of queries.

- \*Enhancement to View\*: Ensure the view includes `warning\_severities\_master.sort\_order AS warning\_severity\_sort\_order` to efficiently support filtering by minimum severity on the geo search endpoint.

- \*Enhancement to View (Optional)\*: Consider adding `media.image\_variants\_json` directly to the view if frequently requested by clients with warnings.

- GIST Index on `segment\_warnings.location\_on\_segment\_geom`: Essential for the performance of `GET /warnings/search/geo`.

- Composite Index on `segment\_warnings(is\_currently\_active, workflow\_status\_code)`: (`idx\_segment\_warnings\_is\_currently\_active\_published`) Vital for quickly finding publicly visible warnings.

- Indexes on Foreign Keys: All FKs in `segment\_warnings` (to `segment\_id`, `warning\_type\_id`, `warning\_severity\_id`, etc.) should be indexed, which was part of their table specs.

- Translation Handling:

- The API backend will be responsible for fetching translations from the `public.translations` table based on the `lang` query parameter and the primary reference language text from the view or base tables. The OpenAPI schema reflects this by showing a base language field (e.g., `title`) and a `translations` object.

- An efficient index on `public.translations (table\_identifier, column\_identifier, row\_foreign\_key, language\_code)` is crucial.

- Write Operations (POST, PATCH):

- These will operate directly on the `public.segment\_warnings` table.

- RLS policies on `segment\_warnings` will enforce permissions (e.g., only regional managers for their segments, or admins).

- The `created\_by\_user\_id` (and `updated\_by\_user\_id`) will be set by the backend based on the authenticated user's ID (`auth.uid()`).

- Complexity & Performance:

- Read endpoints leveraging the view should be performant for most use cases.

- Write endpoints are straightforward inserts/updates to a single table.

- The main complexity lies in the API backend's logic for assembling translations if not handled by a more advanced database function/procedure.

No 🔴 critical schema tweaks are required for this module to support the conceptualized API endpoints, assuming the `public\_active\_segment\_warnings\_view` is implemented and slightly enhanced as noted.