REST API

Introdução

A API do Zoopolis oferece acesso eficiente a informações sobre animais, os pontos dos usuários e localização dos recintos, permitindo consultas, visualizações e recolhas de informações. Desenvolvida para integração com a aplicação móvel, a API fornece os dados atualizados e é compatível várias outras plataformas.

Os dados são entregues em formato JSON, garantindo respostas consistentes e facilitando a integração com diversos sistemas. A estrutura dos dados e os Endpoints são flexíveis, projetados para suportar futuras atualizações e melhorias contínuas na aplicação.

Endpoints

- Mostrar Animais
  - **URL:**

  `/animals`

  - **METHOD:**

  `GET`
- **SUCESS RESPONSE:**

```
[
  {


        "id": [integer],
        "name": [string],
        "ciName": [string],
        "description": [string],
        "weight": [float],
        "height": [float],
        "length": [float],
        "classe": {
              "id": [integer],
              "name": [string],
              "order": [string],
                "family": [string]
              },
        "imageUrl": [string]
  },
]
```

- **ERROR RESPONSE:**

```
{
        "status": 500,
        "message": "An unexpected error occurred.",
        "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```

@GET("animalsDTO")
suspend fun getAnimals(): List<AnimalDTO>

```

- Mostrar Animais por ID
  - **URL:**

  `/animals/:id`

  - **METHOD:**

  `GET`
  - **URL Paramethers:**
    - Required:

    `id: [integer]`
- **SUCESS RESPONSE:**

```
[
  {

        "id": [integer],
        "name": [string],
        "ciName": [string],
        "description": [string],
        "weight": [float],
        "height": [float],
        "length": [float],
        "classe": {
              "id": [integer],
              "name": [string],
              "order": [string],
              "family": [string]
            },
        "imageUrl": [string]
          },

]
```

- **ERROR RESPONSE:**

```
{


        "status": 404,
        "message": "Animal with id {id} not found.",
        "timestamp": [datetime]


}
```

- **SAMPLE CALL:**

```

@GET("animalsDTO/{id}")
suspend fun getAnimalsById(@Path("id") id: Int): AnimalDTO

```

Listar Usuários

- **URL:**

  `/api/persons`

- **METHOD:**

  `GET`
- **SUCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "email": [string],
    "points": [integer]
  }
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@Get("persons")
suspend fun getPersons(): List<Person>
```

Obter Usuário por ID

- **URL:**

  `/api/persons/{id}`
- **METHOD:**

  `GET`
- **URL Parameters:**
  - Required:

`id: [integer]`

- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "User not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@Get("persons/{id}")
suspend fun getPersonById(id: Int): Person
```

Registrar Usuário

- **URL:**

  `/api/persons/register`
- **METHOD:**

  `POST`
- **REQUEST BODY:**

```
{
  "name": [string],
  "email": [string],
  "password": [string]
}
```

- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": 0
}
```

- **ERROR RESPONSE:**

```
{
  "status": 409,
  "message": "Email already in use.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("persons/register")
suspend fun register(@Body person: Person): Person
```

Login

- **URL:**

  `/api/persons/login`
- **METHOD:**

  `POST`
- **REQUEST BODY:**

```
{
  "email": [string],
  "password": [string]
}
```

- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "token": [string]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 401,
  "message": "Invalid password.",
  "timestamp": [datetime]
}

{
  "status": 404,
  "message": "User not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("persons/login")
suspend fun login(@Body loginRequestDTO: LoginRequestDTO): LoginResponseDTO
```

Validar Token

- **URL:**

  `/api/persons/validate`
- **METHOD:**

  `GET`

- **HEADERS:**
  - Authorization: `Bearer [token]`
- **SUCESS RESPONSE:**

```
"Token is valid"
```

- **ERROR RESPONSE:**

```
{
  "status": 401,
  "message": "Invalid or expired token.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@Get("persons/validate")
suspend fun validateToken(@Header("Authorization") token: String): String
```

Adicionar Pontuação

- **URL:**

  `/api/persons/{id}/add-point`
- **METHOD:**

  `POST`
- **URL Parameters:**
  - Required:

    `id: [integer]`
- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "User not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("persons/{id}/add-point")
suspend fun addPointToUser(@Path("id") id: Int): Person
```

Remover Pontuação

- **URL:**

  `/api/persons/{id}/remove-point`
- **METHOD:**

  `POST`
- **URL Parameters:**
  - Required:

    `id: [integer]`
- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "name": [string],
  "email": [string],
  "points": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "User not found or points are already at 0.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("persons/{id}/remove-point")
suspend fun removePointToUser(@Path("id") id: Int): Person
```

Listar Visitas

- **URL:**

  `/api/visited`
- **METHOD:**

  `GET`
- **SUCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "person": {
      "id": [integer],
      "name": [string],
      "email": [string]
    },
    "subArea": {
      "id": [integer],
      "name": [string]
    },
    "dtime": [datetime]
  }
]
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("visited")
suspend fun getAllVisits(): List<Visited>
```

Obter Visita por ID

- **URL:**

  `/api/visited/{id}`
- **METHOD:**

  `GET`
- **URL Parameters:**
  - Required:

    `id: [integer]`
- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "person": {
    "id": [integer],
    "name": [string],
    "email": [string]
  },
  "subArea": {
    "id": [integer],
    "name": [string]
  },
  "dtime": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "Visit not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("visited/{id}")
suspend fun getVisitById(@Path("id") id: Int): Visited
```

Criar Visita

- **URL:**

  `/api/visited`
- **METHOD:**

  `POST`
- **REQUEST BODY:**

```
{
  "personId": [integer],
  "animalId": [integer]
}
```

- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "person": {
    "id": [integer],
    "name": [string],
    "email": [string]
  },
  "subArea": {
    "id": [integer],
    "name": [string]
  },
  "dtime": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 400,
  "message": "Invalid person ID.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@POST("visited")
suspend fun createVisit(
    @Query("personId") personId: Int,
    @Query("animalId") animalId: Int
): Visited
```

Atualizar Visita

- **URL:**

  `/api/visited/{id}`
- **METHOD:**

`PUT`

- **REQUEST BODY:**

```
{
  "dtime": [datetime],
  "person": {
    "id": [integer]
  },
  "subArea": {
    "id": [integer]
  }
}
```

- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "person": {
    "id": [integer],
    "name": [string],
    "email": [string]
  },
  "subArea": {
    "id": [integer],
    "name": [string]
  },
  "dtime": [datetime]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "Visit not found.",
  "timestamp": [datetime]
}
```

Deletar Visita

- **URL:**

  `/api/visited/{id}`
- **METHOD:**

  `DELETE`
- **URL Parameters:**
  - Required:

    `id: [integer]`
- **SUCESS RESPONSE:**

```
{
  "message": "Visit deleted successfully."
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "Visit not found.",
  "timestamp": [datetime]
}
```

Obter Sub-área Mais Visitada

- **URL:**

  `/api/visited/most-visited-subarea`
- **METHOD:**

  `GET`
- **SUCESS RESPONSE:**

```
{
  "subArea": [string],
  "visitCount": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 500,
  "message": "An unexpected error occurred.",
  "timestamp": [datetime]
}
```

Get All Favorites

- **URL:**
  `/api/favorite`
- **METHOD:**
  `GET`
- **SUCESS RESPONSE:**

  ```
  [
    {
      "id": [integer],
      "personId": [integer],
      "animalId": [integer]
    },
    ...
  ]
  ```

- **ERROR RESPONSE:**

  ```
  {
    "status": 500,
    "message": "Internal Server Error.",
    "timestamp": [datetime]
  }
  ```

- **SAMPLE CALL:**

```
@GET("favorite")
suspend fun getFavorites(): List<Favorite>
```

Get Favorite by ID

- **URL:**
  `/api/favorite/{id}`
- **METHOD:**
  `GET`
- **URL Parameters:**
  - Required:
    `id: [integer]`
- **SUCESS RESPONSE:**

```
{
  "id": [integer],
  "personId": [integer],
  "animalId": [integer]
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "Favorite not found.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("favorite/{id}")
suspend fun getFavorite(@Path("id") id: Int): Optional<Favorite>
```

Get Favorite Animals by Person

- **URL:**
  `/api/favorite/person/{personId}`
- **METHOD:**
  `GET`
- **URL Parameters:**
  - Required:
    `personId: [integer]`
- **SUCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "name": [string],
    "species": [string]
  },
  ...
]
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "No favorite animals found for the person.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("favorite/person/{personId}")
suspend fun getFavoriteAnimalsByPerson(@Path("personId") personId: Int):
List<AnimalDTO>
```

Check if Animal is Favorite

- **URL:**
  `/api/favorite/isFavorite`
- **METHOD:**
  `GET`
- **QUERY Parameters:**
  - Required:
    `personId: [integer]`
    `animalId: [integer]`
- **SUCESS RESPONSE:**

  ```
  true
  ```

- **ERROR RESPONSE:**

  ```
  {
    "status": 400,
    "message": "Bad Request.",
    "timestamp": [datetime]
  }
  ```

- **SAMPLE CALL:**

  ```
  @GET("favorite/isFavorite")
  suspend fun isFavorite(
      @Query("personId") personId: Int,
      @Query("animalId") animalId: Int
  ): Boolean
  ```

Add Favorite

- **URL:**
  `/api/favorite/add`
- **METHOD:**
  `POST`
- **QUERY Parameters:**

    - Required:
      `personId: [integer]`
      `animalId: [integer]`
- **SUCESS RESPONSE:**

  ```
  "Animal added to favorites successfully."
  ```

- **ERROR RESPONSE:**

  ```
  {
    "status": 409,
    "message": "Animal is already a favorite for this person.",
    "timestamp": [datetime]
  }
  ```

- **SAMPLE CALL:**

  ```
  @POST("favorite/add")
  suspend fun addFavorite(
      @Query("personId") personId: Int,
      @Query("animalId") animalId: Int
  ): String
  ```

Remove Favorite

- **URL:**
  `/api/favorite/remove`
- **METHOD:**
  `DELETE`
- **QUERY Parameters:**
    - Required:
      `personId: [integer]`
      `animalId: [integer]`
- **SUCESS RESPONSE:**

```
"Animal removed from favorites successfully."
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "Animal is not a favorite for this person.",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@DELETE("favorite/remove")
suspend fun removeFavorite(
    @Query("personId") personId: Int,
    @Query("animalId") animalId: Int
): String
```

Retrieve All Enclosures

URL:

`/api/enclosuresDTO`

METHOD:

`GET`

SUCCESS RESPONSE:

```
[
  {
    "id": [integer],
    "name": [string],
    "animalClass": [string],
    "mapsId": [integer],
    "supportedAmount": [integer],
    "latitude": [float],
    "longitude": [float]
  }
]
```

ERROR RESPONSE:

- **No specific error responses defined for this endpoint.**

SAMPLE CALL:

```
@GET("enclosuresDTO")
suspend fun getAllEnclosures(): List<EnclosureDTO>
```

Retrieve Enclosure by ID

URL:

`/api/enclosuresDTO/{id}`

METHOD:

`GET`

URL Parameters:

- Required:

  `id: [integer]`

SUCCESS RESPONSE:

```
{
  "id": [integer],
  "name": [string],
  "animalClass": [string],
  "mapsId": [integer],
  "supportedAmount": [integer],
  "latitude": [float],
  "longitude": [float]
}
```

ERROR RESPONSE:

```
{
  "status": 404,
  "message": "Enclosure not found",
  "timestamp": [datetime]
}
```

SAMPLE CALL:

```
@GET("enclosuresDTO/{id}")
suspend fun getEnclosureById(@Path("id") id: Int): EnclosureDTO
```

Obter Todos os Registros AE

- **URL:**

  `/api/ae`
- **METHOD:**

  `GET`
- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "dateIn": [string], // Date in ISO format
    "dateOut": [string], // Date in ISO format
    "code": [string],
    "animalDTO": {
      "id": [integer],
      "name": [string],
      "ciName": [string],
      "description": [string],
      "imageUrl": [string]
    },
    "enclosureDTO": {
      "id": [integer],
      "name": [string],
      "animalClass": [string],
      "mapsId": [string],
      "supportedAmount": [integer],
      "latitude": [double],
      "longitude": [double]
    }
  }
]
```

- **SAMPLE CALL:**

```
@GET("ae")
suspend fun getAllAE(): List<AEDTO>
```

Obter Registro AE por ID

- **URL:**

  `/api/ae/{id}`
- **METHOD:**

  `GET`
- **URL Parameters:**
  - Required:

    `id: [integer]`
- **SUCCESS RESPONSE:**

```
{
  "id": [integer],
  "dateIn": [string], // Date in ISO format
  "dateOut": [string], // Date in ISO format
  "code": [string],
  "animalDTO": {
    "id": [integer],
    "name": [string],
    "ciName": [string],
    "description": [string],
    "imageUrl": [string]
  },
  "enclosureDTO": {
    "id": [integer],
    "name": [string],
    "animalClass": [string],
    "mapsId": [string],
    "supportedAmount": [integer],
    "latitude": [double],
    "longitude": [double]
  }
}
```

- **ERROR RESPONSE:**

```
{
  "status": 404,
  "message": "AE with id [id] not found",
  "timestamp": [datetime]
}
```

- **SAMPLE CALL:**

```
@GET("ae/{id}")
suspend fun getAEById(@Path("id") id: Int): AEDTO
```

Obter Registros AE por ID do Animal

- **URL:**

  `/api/ae/animal/{animalId}`
- **METHOD:**

`GET`

- **URL Parameters:**
  - Required:

    `animalId: [integer]`
- **SUCCESS RESPONSE:**

```
[
  {
    "id": [integer],
    "dateIn": [string], // Date in ISO format
    "dateOut": [string], // Date in ISO format
    "code": [string],
    "animalDTO": {
      "id": [integer],
      "name": [string],
      "ciName": [string],
      "description": [string],
      "imageUrl": [string]
    },
    "enclosureDTO": {
      "id": [integer],
      "name": [string],
      "animalClass": [string],
      "mapsId": [string],
      "supportedAmount": [integer],
      "latitude": [double],
      "longitude": [double]
    }
  }
]
```

- **SAMPLE CALL:**

```
@GET("ae/animal/{animalId}")
suspend fun getAEByAnimalId(@Path("animalId") animalId: Int): List<AEDTO>
```

URL:

`/api/images/{imageName}`

METHOD:

`GET`

URL Parameters:

- Required:
  - `imageName: [string]`
    - The name of the image file to retrieve.

SUCCESS RESPONSE:

- **Code:** 200 OK
- **Content:**
  - Returns the requested image file as a binary stream.
  - Headers include `Content-Disposition` to suggest a download with the original filename, and `Content-Type` set to the appropriate media type (e.g., `image/jpeg`).

ERROR RESPONSE:

- **Code:** 404 NOT FOUND
- **Content:**

  ```
  No content returned.
  ```