Exoplanet Travel Agency API Documentation (PDF)

Version: 0.1.0

This document describes the REST API for browsing exoplanet destinations and managing travel bookings.

Base URL

Local: http://localhost:3000

Deployed (optional): https://YOUR-DEPLOYED-URL-HERE

Quick start (local)

1) Install dependencies: npm install
2) Run migrations + seed: npx prisma migrate dev npx prisma db seed
3) Start the server: npm run dev
4) Verify the API is running: curl -s "http://localhost:3000/api/exoplanets?page=1&pageSize=1"

Conventions

Data format

- Requests and responses are JSON unless otherwise stated.
- Set request header: Content-Type: application/json

Pagination

List endpoints support: - page (1-indexed) - pageSize (max 100)

Error format

All error responses follow this structure:

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid query parameters",
    "details": {}
  }
}
```

Common status codes: - 200 OK: Successful GET - 201 Created: Successful POST - 204 No Content: Successful DELETE - 400 Bad Request: Validation / malformed JSON - 404 Not Found: Resource does not exist

GET /api/exoplanets

Purpose: List exoplanets (catalogue page) with pagination, search, and filters.

Query parameters

- page (integer, default 1): Page number (1-indexed)
- pageSize (integer, default 20, max 100): Items per page
- q (string, optional): Case-insensitive substring search in planet name
- vibe (string, optional): Filter by derived category (e.g., "Molten Rock", "Mysterious")
- minDistance (number, optional): Minimum distance in light years (inclusive)
- maxDistance (number, optional): Maximum distance in light years (inclusive)
- sort (distance | discoveryYear | name, default distance)
- order (asc | desc, default asc)

Example requests

Basic (page 1, 20 items): curl -s "http://localhost:3000/api/exoplanets?page=1&pageSize=20" | jq

Search by name: curl -s "http://localhost:3000/api/exoplanets?q=gliese" | jq

Filter by vibe: curl -s "http://localhost:3000/api/exoplanets?vibe=Molten%20Rock" | jq

Filter by distance range: curl -s "http://localhost:3000/api/exoplanets?minDistance=0&maxDistance=25" | jq

Validation error example: curl -i "http://localhost:3000/api/exoplanets?page=-1"

Success response (200)

Returns a paginated list:

```
{
  "items": [
    {
      "id": "cmlp9k8qn0003dskufekwmztt",
      "name": "HD 219134 c",
      "distance": 21.3,
      "temperature": 782,
      "gravity": 1.91,
      "vibe": "Molten Rock",
      "discoveryYear": 2015
    }
  ],
  "page": 1,
  "pageSize": 20,
  "total": 500,
  "totalPages": 25
}
```

Error responses

- 400: Invalid query parameters (e.g. page < 1)

GET /api/exoplanets/{id}

Purpose: Fetch full details for a single exoplanet.

Path parameters

- id (string): Exoplanet id (Prisma cuid)

Example request

```
curl -s "http://localhost:3000/api/exoplanets/cmlp9k8qn0000dskuetg0pq62" | jq
```

Tip: get a valid id from the list endpoint:

```
curl -s "http://localhost:3000/api/exoplanets?pageSize=1" | jq -r ".items[0].id"
```

Success response (200)

Returns a single exoplanet object:

```
{
  "id": "cmlp9k8qn0000dskuetg0pq62",
  "name": "HD 219134 b",
  "distance": 21.3,
  "temperature": 782,
  "gravity": 1.91,
  "vibe": "Molten Rock",
  "discoveryYear": 2015
}
```

Error responses

- 400: Invalid id format
- 404: Exoplanet not found

GET /api/bookings

Purpose: List bookings. Optionally filter by userId.

Query parameters

- page (integer, default 1)
- pageSize (integer, default 20, max 100)
- userId (string, optional): Filter bookings for a given user

Example requests

List all bookings: curl -s “http://localhost:3000/api/bookings?page=1&pageSize=10” | jq

Filter by userId: curl -s “http://localhost:3000/api/bookings?userId=cmlp9k8ut00dwdskud6g70sxc” | jq

Tip: obtain a userId (if you seeded users) via Prisma Studio: npx prisma studio

Success response (200)

Returns a paginated list of bookings including user and planet summary info:

```
{
  "items": [
    {
      "id": "cmlp9k90d00dzdskuje80hqav",
      "bookingDate": "2026-02-16T14:52:20.000Z",
      "travelClass": "Economy (Cryo-Sleep)",
      "user": {
        "id": "cmlp9k8ut00dwdskud6g70sxc",
        "name": "Peter Quill",
        "email": "star.lord@guardians.com"
      },
      "planet": {
        "id": "cmlp9k8qn0007dskuhej1ifkh",
        "name": "HD 219134 c",
        "distance": 21.3,
        "vibe": "Molten Rock",
        "discoveryYear": 2015
      }
    }
  ],
  "page": 1,
  "pageSize": 10,
  "total": 1,
  "totalPages": 1
}
```

Error responses

- 400: Invalid query parameters

POST /api/bookings

Purpose: Create a booking for a user to travel to an exoplanet.

Note: In the current implementation, userId is supplied in the request body. In a future version, userId would typically come from authentication (JWT/session).

Request body fields

- userId (string): User id
- planetId (string): Exoplanet id
- travelClass (string): Travel class label (e.g. "Economy (Cryo-Sleep)")

Example request

```
curl -s -X POST "http://localhost:3000/api/bookings" \
  -H "Content-Type: application/json" \
  -d '{"userId":"cmlp9k8ut00dwdskud6g70sxc","planetId":"cmlp9k8qn0007dskuhej1ifkh","travelCl
```

Success response (201)

Returns the created booking:

```
{
  "id": "cmlp9k90d00dzdskuje80hqav",
  "userId": "cmlp9k8ut00dwdskud6g70sxc",
  "planetId": "cmlp9k8qn0007dskuhej1ifkh",
  "travelClass": "Economy (Cryo-Sleep)",
  "bookingDate": "2026-02-16T14:52:20.000Z"
}
```

Error responses

- 400: Invalid JSON or validation error
- 404: User or exoplanet not found

PATCH /api/bookings/{id}

Purpose: Partially update a booking (e.g., change travel class).

Path parameters

- id (string): Booking id

Request body

Provide at least one field: - travelClass (string)

Example request

```
curl -s -X PATCH "http://localhost:3000/api/bookings/cmlp9k90d00dzdskuje80hqav" \
  -H "Content-Type: application/json" \
  -d '{"travelClass":"First Class (Warp Drive)"}' | jq
```

Validation error example (empty update payload): curl -i -X PATCH "http://localhost:3000/api/bookings/cmlp9k90d00dzdskuje80hqav"
-H "Content-Type: application/json"
-d "{}"

Success response (200)

Returns the updated booking:

```
{
  "id": "cmlp9k90d00dzdskuje80hqav",
  "userId": "cmlp9k8ut00dwdskud6g70sxc",
  "planetId": "cmlp9k8qn0007dskuhej1ifkh",
  "travelClass": "First Class (Warp Drive)",
  "bookingDate": "2026-02-16T14:52:20.000Z"
}
```

Error responses

- 400: Invalid id or invalid payload
- 404: Booking not found

DELETE /api/bookings/{id}

Purpose: Delete a booking.

Path parameters

- id (string): Booking id

Example request

```
curl -i -X DELETE "http://localhost:3000/api/bookings/cmlp9k90d00dzdskuje80hqav"
```

Success response (204)

No response body.

Error responses

- 400: Invalid id
- 404: Booking not found