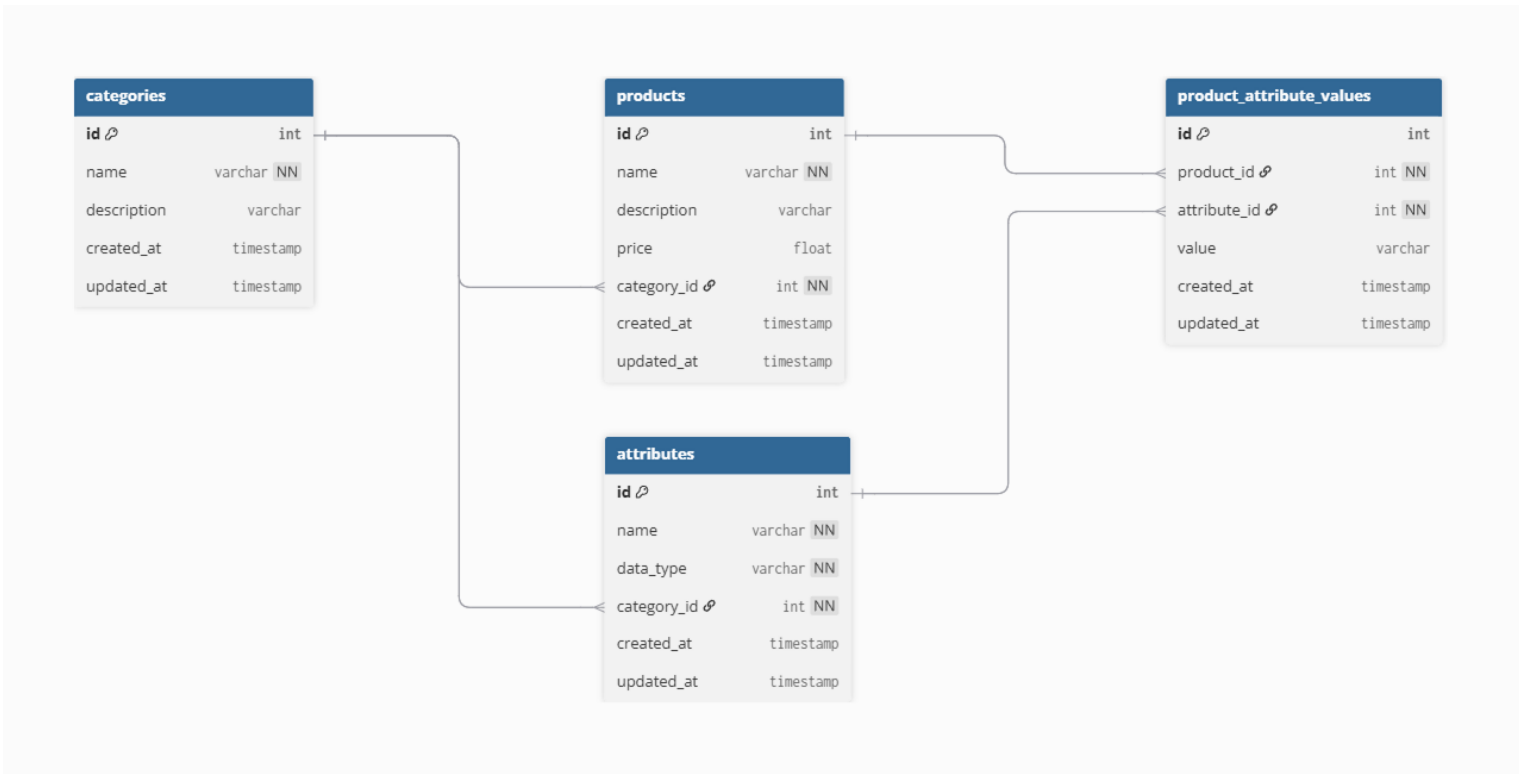
Internal Product Management Tool

1. Project Overview

This project is an **Internal Product Management Tool** for managing products, categories, and product-specific attributes. It allows CRUD operations for products, categories, and attributes, providing flexibility for eCommerce management.

---

2. ERD (Entity Relationship Diagram)

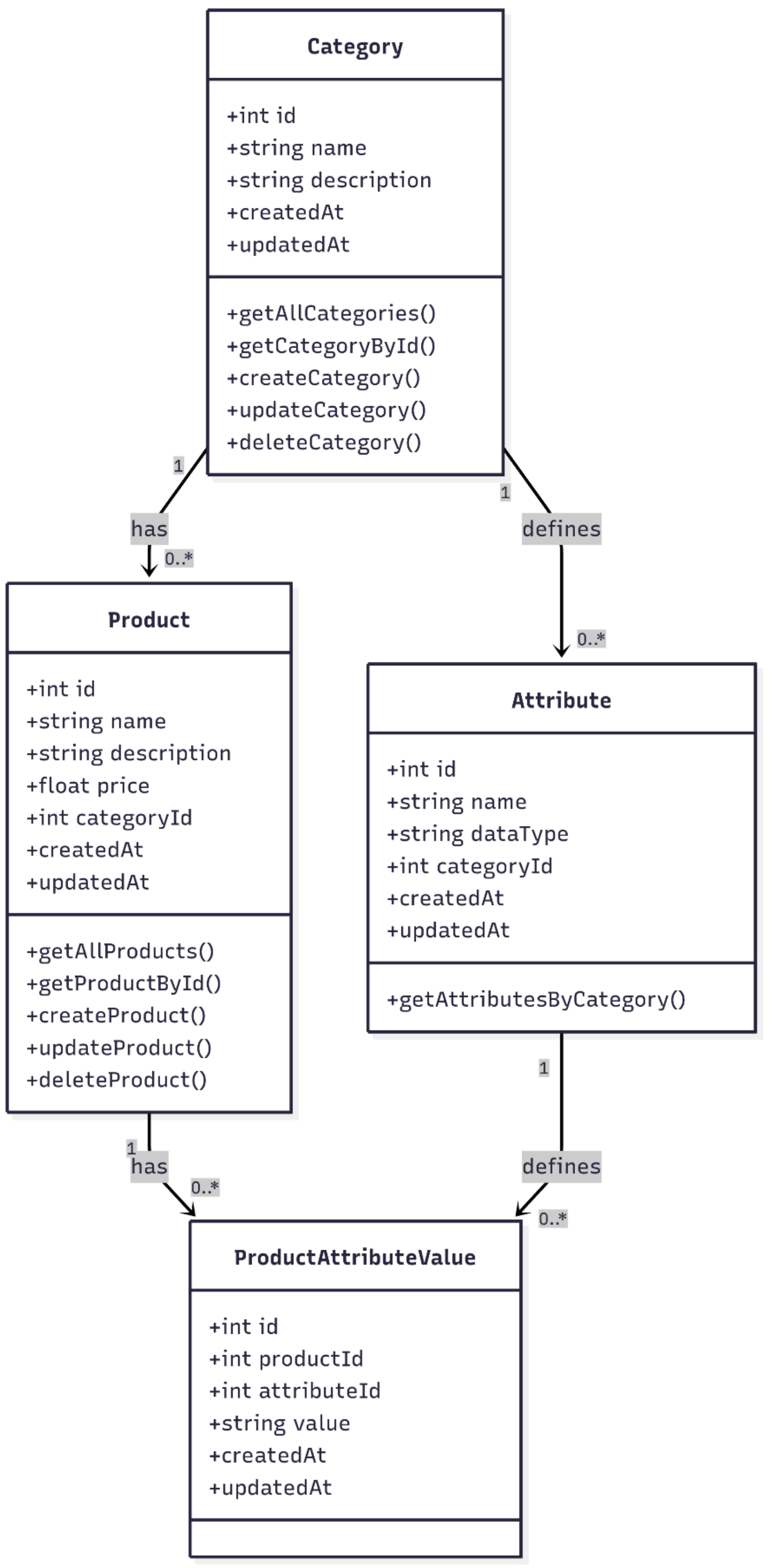


Justification:

- Entities normalized to reduce redundancy.
- Flexibility for adding new categories, products, or attributes.
- ProductAttributeValue acts as a join table to support many-to-many relationships between products and attributes.
- Scalable for future extensions like adding inventory, suppliers, or variants.

---

3. Class Diagram



Explanation:

- **Category**: Holds product categories. CRUD operations implemented via API.
- **Product**: Stores product details and links to a category. CRUD operations implemented.
- **Attribute**: Stores attributes like Color, Size, Brand. Can fetch attributes by category.
- **ProductAttributeValue**: Holds attribute values per product. Acts as a join table.

Relationships:

- 1 Category → N Products
- 1 Category → N Attributes
- 1 Product → N ProductAttributeValues
- 1 Attribute → N ProductAttributeValues

---

4. Tech Stack

- **Frontend**: React + Vite + JavaScript + CSS
- **Backend**: Node.js + Express.js
- **Database**: PostgreSQL + Sequelize ORM
- **Tools**: Postman for API testing, dbdiagram.io for ERD, Mermaid.js for diagrams

---

5. Backend Implementation

Database Models

1. **Category**: id, name, description
2. **Product**: id, name, description, price, categoryId
3. **Attribute**: id, name, dataType, categoryId
4. **ProductAttributeValue**: id, productId, attributeId, value

Relationships

- Product belongs to Category
- Product has many ProductAttributeValues
- ProductAttributeValue belongs to Product and Attribute
- Attribute belongs to Category

Role of ORM (Sequelize)

- **Abstraction**: Sequelize converts SQL queries into easy-to-use JavaScript methods for CRUD operations.
- **Modeling**: Each table is a model class with associations (belongsTo, hasMany).
- **Validation & Constraints**: Ensures data integrity automatically.
- **Migration Support**: Handles schema changes via migrations for flexibility and version control.

API Endpoints

Resource	Method	Endpoint	Description
Categories	GET	/api/categories	Fetch all categories
Categories	POST	/api/categories	Create a new category
Categories	PUT	/api/categories/:id	Update a category
Categories	DELETE	/api/categories/:id	Delete a category
Products	GET	/api/products	Fetch all products with category and attributes
Products	GET	/api/products/:id	Fetch single product
Products	POST	/api/products	Create a product (with attribute values)
Products	PUT	/api/products/:id	Update a product
Products	DELETE	/api/products/:id	Delete a product
Attributes	GET	/api/attributes?categoryId=:id	Fetch attributes by category

---

6. Frontend Implementation

- React + Vite setup with components:
  - ProductForm.jsx – Add/Edit product with dynamic attribute fields.
  - ProductList.jsx – Display list of products with category and attribute values.
  - CategoryForm.jsx – Add new category dynamically from the frontend.
- Handles dynamic fetching of attributes based on selected category.
- Supports adding new attributes to products (backend support required).
- Uses **Axios** for API requests and real-time UI updates.

---

7. Setup Instructions

1. **Clone Repository**

   ```
   git clone <repo_url>
   cd internal-product-management
   ```

2. **Backend Setup**

   ```
   cd backend
   npm install
   # Update database credentials in .env
   npx sequelize db:create
   npx sequelize db:migrate
   npm run dev
   ```

3. **Frontend Setup**

   ```
   cd frontend
   npm install
   npm run dev
   ```

4. Access the frontend at http://localhost:5173 and backend at http://localhost:5000.

---

8. Future Implementation

- Support for adding new attributes directly from the frontend.
- Adding inventory management and supplier linking.
- Product variant support (size, color combinations).
- Role-based access control for admin and staff users.
- Enhanced UI with responsive design and better UX.