Recipe Nutrition Backend API

A Laravel-based REST API that manages recipes with ingredients and preparation steps. Nutritional values for entire recipes are automatically calculated by integrating a mock external Nutrition API.

🔧 Setup

1. **Clone the repository and install dependencies:**

```
git clone <your_repository_url> recipe-app
cd recipe-app/recipe-app-backend
composer install
```

2. **Configure Environment:**
   Copy `.env.example` to `.env`.
   Set your database connection (e.g., `DB_CONNECTION=sqlite` and `touch database/database.sqlite`).
   Add the Nutrition API credentials (from the exercise prompt, not your actual username/password):

```
NUTRITION_API_USERNAME="name"
NUTRITION_API_PASSWORD="password"
```

3. **Generate Application Key & Run Migrations:**

```
php artisan key:generate
php artisan migrate
```

4. **Start the Laravel development server:**

```
php artisan serve
```

   The API will be available at `http://127.0.0.1:8000` (or similar).

🧪 Nutrition API Integration (Part 1)

This part confirms interaction with the external mock Nutrition API.

Command:

```
php artisan nutrition:test
```

Functionality:
This Artisan command performs the following:

- Fetches and displays all existing ingredients from the external API.
- Searches for a specific ingredient (e.g., "Apple").
- **POSTs two new ingredients:** "Dragon Fruit" and "Jackfruit" to the external API.
- Confirms the successful addition of the new ingredients by searching for them.

🍽️ Recipe API (Part 2)

These are the RESTful endpoints for managing recipes within this application.

`POST /api/recipes`

Creates a new recipe, including its associated ingredients and steps.

Request Body (JSON Example):

```
{
  "title": "Spicy Chicken Stir-fry",
  "ingredients": [
    { "name": "Chicken Breast", "quantity": 300, "unit": "grams" },
    { "name": "Broccoli",       "quantity": 200, "unit": "grams" },
    { "name": "Soy Sauce",      "quantity": 50,  "unit": "ml" }
  ],
  "steps": [
    { "description": "Cut chicken into pieces.", "order": 1 },
    { "description": "Stir-fry chicken and vegetables.", "order": 2 },
    { "description": "Add soy sauce and serve.", "order": 3 }
  ]
}
```

`GET /api/recipes`

Returns a list of all recipe titles and their IDs.

`GET /api/recipes/{id}`

Returns a single recipe, including its full details (title, ingredients, steps) and the **calculated total nutritional values** (carbs, fat, protein) fetched from the external Nutrition API for each ingredient.

`DELETE /api/recipes/{id}`

Deletes a recipe and all its associated ingredients and steps.

💡 Development Notes

- **External API Interaction:** Laravel's `Http` facade is used to securely interact with the external Nutrition API, handling Basic Authentication. The `getNutritionData` helper method in `RecipeController` encapsulates this logic.
- **Nutrition Calculation:** Total recipe nutrition is calculated dynamically in the `GET /api/recipes/{id}` endpoint. It iterates through the recipe's ingredients, fetches individual nutrition data from the external API, and sums them up. The quantity of each ingredient acts as a direct multiplier for its nutritional values.
- **Database Structure:** Uses `Recipe`, `Ingredient` (for recipe-specific ingredients), and `Step` Eloquent models with defined `hasMany` and `belongsTo` relationships. `onDelete('cascade')` ensures related data is cleaned up on recipe deletion.
- **Validation:** All incoming API requests are validated using Laravel's robust validation features.
- **Credentials:** External API credentials are stored securely in the `.env` file and accessed via `env()`.

✅ Features Completed (Backend)

- Nutrition API integration (GET + POST)
  - Submitted "Dragon Fruit" and "Jackfruit"
- `Recipe`, `Ingredient`, `Step` models with migrations
- `POST`, `GET`, `DELETE` recipe endpoints
- Nutrition auto-calculation on recipe retrieval (`GET /api/recipes/{id}`)
- Laravel validation, Eloquent relationships
- Credentials in `.env`, clean structure, PSR-4 compliance