Backend API
Flask Service

Base URL examples (configurable via deployment):

- Local development (Docker or direct): `http://localhost:8080`
- In Next.js: `${process.env.NEXT_PUBLIC_API_URL}`

All JSON responses are UTF-8 encoded. Unless stated otherwise, endpoints respond with `application/json`.

1 Authentication – /api/auth

Module: `backend/app/routes/auth_routes.py`

1.1 POST /api/auth/register

Register a new user.

Request body (JSON)

- `email` (string, required)
- `password` (string, required)

Responses

- **201 Created** – `"message": "User registered"`
- **409 Conflict** – `"error": "Email already exists"`
- **500 Internal Server Error** – `"error": <string>`

1.2 POST /api/auth/login

Authenticate a user and return a JWT.

Request body (JSON)

- `email` (string, required)
- `password` (string, required)

Responses

- **200 OK** – JSON including message, token, and user payload (id, email, name).
- **400 Bad Request** – missing email/password or invalid JSON.
- **401 Unauthorized** – `"error": "Invalid credentials"`
- **500 Internal Server Error** – `"error": "Login error: ..."`

The JWT payload includes `id`, `email`, and `name`. Tokens expire 10 days after issuance.

1.3 GET /api/auth/profile

Get the user profile encoded in the JWT.

Headers

- `Authorization: Bearer <jwt>` (required)

Responses

- **200 OK** – `"user": {...jwtPayload}`
- **400 Bad Request** – `"error": "Token missing"`
- **401 Unauthorized** – `"error": "Invalid or expired token"`

1.4 GET /api/auth/logout

Stateless logout endpoint; the client should simply discard the token.

Responses

- **200 OK** – `"message": "Logged out"`

1.5 GET /api/auth/users

Retrieve all users.

Responses

- **200 OK** – array of user objects (id, email, created_at).
- **500 Internal Server Error** – `"error": <string>`

1.6 PUT /api/auth/users/<user_id>

Update a user.

Request body (JSON)

- `email` (string, required)
- `password` (string, optional; replaces existing password if present)

Responses

- **200 OK** – `"message": "User updated"`
- **404 Not Found** – `"error": "User not found"`
- **409 Conflict** – `"error": "Email already exists"`
- **500 Internal Server Error** – `"error": <string>`

1.7 DELETE /api/auth/users/<user_id>

Delete a user by ID.

Responses

- **200 OK** – `"message": "User deleted"`
- **404 Not Found** – `"error": "User not found"`
- **500 Internal Server Error** – `"error": <string>`

2 Products – /api/products

Module: `backend/app/routes/product_routes.py`

2.1 GET /api/products/categories

List all available product categories.

Responses

- **200 OK** – array of `{value, label}` objects.
- **500 Internal Server Error** – `"error": <string>`

2.2 POST /api/products

Create a new product.

Request body (JSON)

- `name` (string, required)
- `description` (string, optional)
- `price` (number, required)
- `image_cover` (string, optional; URL)
- `image_details` (array of strings, optional)
- `colors` (array of strings, optional; default `["black", "white"]`)
- `quantity` (number, required)
- `category` (string, required; name of a ProductCategory value)

Responses

- **201 Created** – `"message": "Product created", "id": <number>`
- **400 Bad Request** – missing or invalid fields.
- **500 Internal Server Error** – `"error": <string>`

2.3 GET /api/products

List products, optionally filtered by category and limited in count.

Query parameters

- `category` (string, optional; enum name)
- `elements` (integer, optional; maximum number of products)

Responses

- **200 OK** – array of products with optional active discount percentage.
- **400 Bad Request** – invalid category.
- **500 Internal Server Error** – `"error": <string>`

2.4 GET /api/products/recommande

Get up to eight random recommended products, optionally filtered by category.

Query parameters

- `category` (string, optional; enum name)

Responses

- **200 OK** – array of product objects (without discount field).
- **400 Bad Request** – invalid category.
- **500 Internal Server Error** – `"error": <string>`

2.5 GET /api/products/<product_id>

Get a product by ID.

Responses

- **200 OK** – product object.
- **404 Not Found** – `"error": "Product not found"`
- **500 Internal Server Error** – `"error": <string>`

2.6 PUT /api/products/<product_id>

Update an existing product (same shape as create; some fields optional).

Responses

- **200 OK** – `"message": "Product updated", "id": <number>`
- **400 Bad Request** – invalid or missing fields.
- **404 Not Found** – `"error": "Product not found"`
- **500 Internal Server Error** – `"error": <string>`

2.7 DELETE /api/products/<product_id>

Delete a product.

Responses

- **200 OK** – `"message": "Product deleted"`
- **404 Not Found** – `"error": "Product not found"`
- **500 Internal Server Error** – `"error": <string>`

3 Offers – /api/offers

Module: `backend/app/routes/offers_routes.py`

3.1 GET /api/offers/

Get all active offers with product details.

Responses

- **200 OK** – array of `Offer` objects enriched with product metadata.
- **500 Internal Server Error** – `"error": <string>`

3.2 GET /api/offers/all

Get all offers (active and inactive) with product details.

3.3 GET /api/offers/<offer_id>

Get an offer by ID.

Responses

- **200 OK** – offer object with product details.
- **404 Not Found** – `"error": "Offer not found"`
- **500 Internal Server Error** – `"error": <string>`

3.4 POST /api/offers/

Create a new offer.

Request body (JSON)

- `product_id` (number, required; must refer to existing product)
- `title` (string, required)
- `discount_percentage` (number, 0–100, required)
- `start_date` (string, ISO-8601, required)
- `end_date` (string, ISO-8601, required)

Responses

- **201 Created** – new offer object.
- **400 Bad Request** – invalid discount or missing fields.
- **404 Not Found** – `"error": "Product not found"`
- **500 Internal Server Error** – `"error": <string>`

3.5 PUT /api/offers/<offer_id>

Update an existing offer (all fields optional).

Responses

- **200 OK** – updated offer.
- **400 Bad Request** – invalid discount.
- **404 Not Found** – `"error": "Offer not found"`
- **500 Internal Server Error** – `"error": <string>`

3.6 DELETE /api/offers/<offer_id>

Delete an offer.

Responses

- **200 OK** – `"message": "Offer deleted"`
- **404 Not Found** – `"error": "Offer not found"`
- **500 Internal Server Error** – `"error": <string>`

4 Orders – /api/orders

Module: `backend/app/routes/order_routes.py`

4.1 POST /api/orders

Create a new order with line items.

Request body (JSON)

- `utilisateur_id` (number, required)
- `items` (array, required). Each item includes:
  - `product_id` (number, required)
  - `quantity` (integer, required, > 0)
  - `size` (string, required)
  - `color` (string, required)
- `status` (string, optional; defaults to `"Pending"`)

Responses

- **201 Created** – `"message": "Order created", "order_id": <number>, "status": <string>`
- **400 Bad Request** – validation errors or unknown product.

4.2 GET /api/orders

List all orders (without line items).

Responses

- **200 OK** – array of `Order.to_dict()`.
- **500 Internal Server Error** – `"error": <string>`

4.3 GET /api/orders/<order_id>

Get a single order with its line items and product names.

Responses

- **200 OK** – order object including `items` list.
- **404 Not Found** – `"error": "Order not found"`
- **500 Internal Server Error** – `"error": <string>`

4.4 GET /api/orders/user/<user_id>

Get all orders for a given user.

Responses

- **200 OK** – array of order objects as above.
- **500 Internal Server Error** – `"error": <string>`

4.5 PUT /api/orders/<order_id>

Update an order's line items and optionally its status.

Responses

- **200 OK** – `"message": "Order updated", "order_id": <number>`
- **400 Bad Request** – validation errors.
- **404 Not Found** – `"error": "Order not found"`

4.6 DELETE /api/orders/<order_id>

Delete an order (and its line items via cascade).

Responses

- **200 OK** – `"message": "Order deleted"`
- **404 Not Found** – `"error": "Order not found"`
- **400 Bad Request** – `"error": <string>`

5 Blogs – /api/blogs

Module: `backend/app/routes/blogs_routes.py`

5.1 GET /api/blogs/

List all blogs, newest first.

Responses

- **200 OK** – array of `Blog.to_dict()`.
- **500 Internal Server Error** – `"error": <string>`

5.2 GET /api/blogs/<blog_id>

Get a single blog post.

Responses

- **200 OK** – blog object.
- **404 Not Found** – `"error": "Blog not found"`
- **500 Internal Server Error** – `"error": <string>`

5.3 POST /api/blogs/

Create a blog post.

Request body (JSON)

- `title` (string, required)
- `content` (string, required)
- `image_url` (string, optional)
- `is_published` (boolean or string, optional; defaults to true)

Responses

- **201 Created** – `"message": "Blog created successfully"`
- **400 Bad Request** – invalid JSON or missing fields.
- **500 Internal Server Error** – `"error": <string>`

5.4 PUT /api/blogs/<blog_id>

Update an existing blog post.

Responses

- **200 OK** – updated blog.
- **404 Not Found** – `"error": "Blog not found"`
- **500 Internal Server Error** – `"error": <string>`

5.5 DELETE /api/blogs/<blog_id>

Delete a blog post.

Responses

- **200 OK** – `"message": "Blog deleted"`
- **404 Not Found** – `"error": "Blog not found"`
- **500 Internal Server Error** – `"error": <string>`

6 Uploads – /api/upload

Module: `backend/app/routes/upload_routes.py`

6.1 POST /api/upload/image

Upload a single image to Cloudinary.

Request (multipart/form-data)

- `file` (file, required)
- `folder` (string, optional; default `"ecommerce"`)

Responses

- **200 OK** – JSON including secure URL, public ID, format, and dimensions.
- **400 Bad Request** – e.g., no file provided or selected.

6.2 POST /api/upload/images

Upload multiple images to Cloudinary.

Request (multipart/form-data)

- `files` (array of files, required)
- `folder` (string, optional; default `"ecommerce"`)

Responses

- **200 OK** – summary of uploaded and failed items when at least one upload succeeds.
- **400 Bad Request** – if all uploads fail or inputs are missing.

7 AI Chat – /api/opnai

Module: `backend/app/routes/opnai_routes.py`

7.1 POST /api/opnai/chat

Forward chat to the OpenAI API with catalog-aware system prompts.

Environment

- `OPENAI_API_KEY` (required; otherwise 501 is returned)
- `OPENAI_MODEL` (optional; default `gpt-4o-mini`)

Request body (JSON)

- Either `"prompt": <string>` (simple mode) or `"messages": [{role, content}, ...]` (advanced mode).
- Optional: `temperature` (float, default 0.7), `max_tokens` (integer, default 512).

Responses

- **200 OK** – object with assistant message and raw OpenAI response.
- **400 Bad Request** – missing prompt or messages.
- **501 Not Implemented** – missing API key.
- **502 Bad Gateway** – error from OpenAI API.
- **500 Internal Server Error** – `"error": <string>`

8 Admin panel – /admin

Module: `backend/app/admin/main.py`

The admin panel is mostly HTML-based but is part of the backend surface.

- Session-based admin login using environment variables `ADMIN_EMAIL` and `ADMIN_PASSWORD`.
- Routes include:
  - `GET/POST /admin/login` – admin login form.
  - `GET /admin/logout` – logout and redirect to login.

- `GET /admin/` – dashboard home.
- `GET /admin/products`, `/admin/users`, `/admin/orders`, `/admin/offers`, `/admin/blogs` – HTML views.

These pages are expected to call the JSON endpoints documented above.