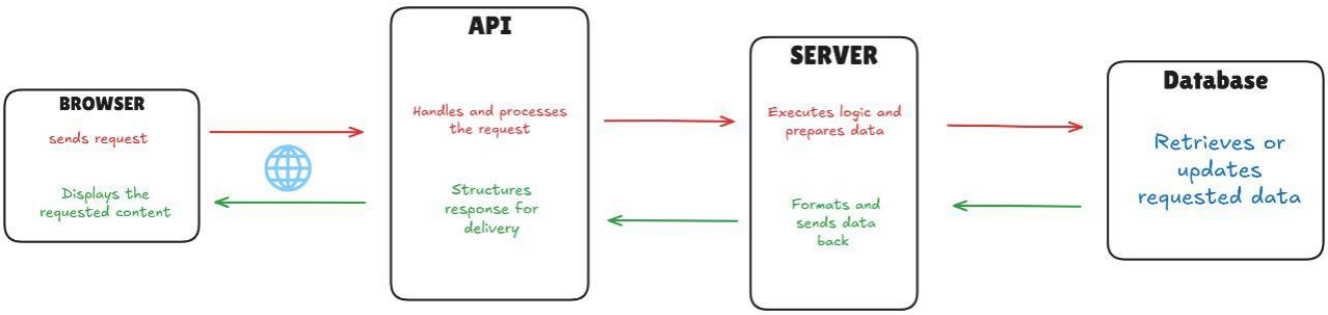
Day 03

API Integration and Data Migration



API Integration Process

1. Configuring Sanity CMS

- Set up a Sanity project and dataset.
- Install necessary dependencies like `@sanity/client`.
- Define schemas to store API data (e.g., product data).

2. Defining Schemas in Sanity

- Create custom schemas for data (e.g., product name, price, images).
- - Ensure schema fields match the API data structure.

3. Connecting to External API

- Obtaining necessary **API keys** for authentication.

4. Store Data in Sanity

- Use the Sanity client to create documents based on the fetched API data.

5. Query Data from Sanity

- Use GROQ queries to fetch stored data from Sanity.

6. Display Data on Frontend

- Next.js to display the fetched data on the frontend.
- Render dynamic content like product lists on the UI.

Tools & Technologies Used:

- **Sanity CMS:** Content management and data storage.
- **API Client:** Fetch API data.
- **GROQ:** Query data from Sanity CMS.
- **Next.js:** Frontend framework for displaying data.

Conclusion:

The API data is fetched, stored in Sanity CMS, and dynamically displayed on the frontend for seamless integration.

Migration Steps and Tools used

1. Setup Environment Variables

- Create a .env.local file to store sensitive data (e.g., Sanity project ID, dataset, and API token).

2. Initialize Sanity Client

- Use createClient from @sanity/client to configure the connection to your Sanity project.

3. Fetch Data

- fetch product data from the API endpoint **https://template-03-api.vercel.app/api/products**.

4. Image Upload

- Download images from the source URL using Axios and convert them to a buffer using Buffer.from().
- Upload images to Sanity using the assets.upload() method.

5. Transform and Upload Products

- Iterate over the fetched products, transforming the data to fit the structure of the Sanity schema before uploading.

6. Create Product Entries

- Insert the transformed product data into Sanity by using `client.create()` to create new entries.

7. Error Handling

- Use try-catch blocks to handle and log errors that may occur during data transformation or the upload process.

8. Run Migration Script

- Execute the importData function to start the migration process.
- Logs indicate progress and completion status.

Tools Used:

Sanity.io

- For content management and data storage.
- Used @sanity/client library for interacting with the Sanity API.

Axios

- For making HTTP requests to fetch data and images from the source API.

Dotenv

- For loading environment variables from .env.local.

Node.js Built-in Modules

- url, path, and Buffer for file handling and environment setup.