Service HTTP

1. Centralisation des routes

Objectif : éviter les strings en dur et structurer les endpoints.

```
const BASE_URL = "http://localhost:8080"

export const API_ROUTES = {
  events: {
    all: () => ${BASE_URL}/events,
    byId: (id: string) => ${BASE_URL}/events/${id},
  },
  users: {
    all: () => ${BASE_URL}/users,
    byId: (id: string) => ${BASE_URL}/users/${id},
  },
}
```

2. Client HTTP générique

Permet de gérer toutes les requêtes (GET, POST, etc.)

```
type HttpMethod = "GET" | "POST" | "PUT" | "PATCH" | "DELETE"

async function request<T>(
  url: string,
  method: HttpMethod,
  body?: any
): Promise<T> {
  const res = await fetch(url, {
    method,
    headers: {
      "Content-Type": "application/json",
    },
    body: body ? JSON.stringify(body) : undefined,
  })
  if (!res.ok) {
    const errorText = await res.text()
    throw new Error(API Error: ${res.status} - ${errorText})
  }
  return res.json()
}
```

3. Méthodes HTTP exposées

```

```

```
export const apiClient = {
  get: <T>(url: string) => request<T>(url, "GET"),

  post: <T>(url: string, body: any) =>
    request<T>(url, "POST", body),
  put: <T>(url: string, body: any) =>
    request<T>(url, "PUT", body),
  patch: <T>(url: string, body: any) =>
    request<T>(url, "PATCH", body),
  delete: <T>(url: string) =>
    request<T>(url, "DELETE"),
}
```

4. Service métier (Events)

Combine routes + client HTTP.

```
import { apiClient } from "./api.client"
import { API_ROUTES } from "./api.routes"

export const EventsService = {
  getAll: () =>
    apiClient.get(API_ROUTES.events.all()),
  getById: (id: string) =>
    apiClient.get(API_ROUTES.events.byId(id)),
  create: (data: any) =>
    apiClient.post(API_ROUTES.events.all(), data),
  update: (id: string, data: any) =>
    apiClient.put(API_ROUTES.events.byId(id), data),
  patch: (id: string, data: any) =>
    apiClient.patch(API_ROUTES.events.byId(id), data),
  delete: (id: string) =>
    apiClient.delete(API_ROUTES.events.byId(id)),
}
```

5. Utilisation

Côté Server

```
import { EventsService } from "@/services/events.service"

export default async function Page() {
  const events = await EventsService.getAll()
  return <div>{events.length}</div>
}
```

Côté Client

```
"use client"

import { useEffect, useState } from "react"
import { EventsService } from "@/services/events.service"
export default function EventsList() {
  const [events, setEvents] = useState([])
  useEffect(() => {
    EventsService.getAll().then(setEvents)
  }, [])
  return <div>{events.length}</div>
}
```

6. Typage

```
export interface Event {
  id: string
  name: string
  description: string
}
```

```
getAll: () =>
  apiClient.get<Event[]>(API_ROUTES.events.all())
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/web/framework/nextjs/http-service

Last update: **2026/04/03 12:04**