

GUI開發與HA部署分析



以繁體中文詳細說明:
我是台積電it, 想要開發一個gui服務, 方便從原先輸入json來使用api, 變為使用gui來做到, 必須要有ha高可用, 之後要部署到k8s
主管要我跟他分析應使用哪一種軟體工具或框架來實作



了解, 你的情境是在台積電IT部門開發一個**提供 GUI 操作界面、取代直接輸入 JSON 使用 API 的服務**, 並要求:

- 支援 **高可用性 (High Availability, HA)**
- 將來部署至 **Kubernetes (K8s)**
- GUI 須能對應目前的 API 請求格式 (例如以 JSON 傳遞參數)
- 須與現有基礎架構整合良好

一、目標需求整理

項目	說明
功能	將原本使用 JSON 呼叫 API 的流程, 改為 Web GUI 操作
架構需求	高可用、可部署於 K8s
使用者介面	Web-based (網頁 GUI)
開發團隊背景	IT 工程師團隊, 具備內部 API 經驗, 需選擇合適的前後端技術

項目	說明
可擴展性	將來可能新增更多表單、頁面、甚至整合 SSO、RBAC、審核流程等

二、推薦技術組合

1. 前後端開發框架推薦

框架	適合原因	備註
前端：React (Next.js)	開發效率高、元件化、社群活躍，適合做動態表單、API呼叫	若有 Server-side rendering 需求則使用 Next.js
後端：FastAPI (Python)	原生支援 JSON 輸入/輸出、Swagger 文件自動生成、ASGI 高效能，HA 輕鬆實現	若團隊較熟 Python 建議使用
後端：Node.js + Express	若團隊熟 JavaScript / TypeScript，可選此架構整合前後端	可用 NestJS 提升可維護性

2. Web GUI 表單工具

若表單內容為 JSON 格式，推薦以下幾種 UI 表單工具：

工具	功能說明	優點
React JSON Schema Form	可依 JSON Schema 自動生成 GUI 表單	適合快速從 JSON 轉 GUI
Ant Design + React Hook Form	適合進階 GUI 設計，表單驗證、權限控制等	更彈性，可自定欄位、樣式
Formik / React Final Form	表單狀態管理好用，與 UI 框架整合好	適合複雜互動需求

3. HA 架構與部署方式

架構元件	說明
部署：Docker + K8s	將服務包成容器，透過 K8s 控制複本數量實現 HA
Ingress + HPA	使用 NGINX Ingress Controller 做負載均衡，搭配 Horizontal Pod Autoscaler
資料儲存（如有）	若涉及狀態管理，可用 Redis、PostgreSQL（RDS）等，需搭配 PVC 與 StatefulSet 考量 HA
設定儲存	configMap / secret 儲存靜態設定與敏感資訊，利於 CI/CD

4. CI/CD 與部署工具

工具	用途
GitLab CI / GitHub Actions	撰寫 pipeline，自動部署
Helm / Kustomize	管理 Kubernetes 資源模板

工具	用途
ArgoCD	GitOps 風格部署，可管理多環境

三、選型分析建議（彙整給主管）

A. 建議使用技術組合

層級	技術	原因
前端	React + JSON Schema Form or Ant Design	快速生成與高度自定表單
後端	FastAPI	高效能、JSON 支援佳，簡單開發
架構	K8s + HPA + Ingress	符合公司標準、HA 支援完善
部署	Docker + Helm + GitLab CI	可重用、易擴展、支援版本控管

B. 補充建議

- 若公司內已有標準 Web 技術棧（如 Vue、Spring Boot），也可優先考量使用一致技術
- 目標為將 JSON 輸入「模組化」、「可視化」，可考慮使用動態表單 schema + API 輸入驗證框架組合
- 開發過程可考慮導入 UI Mock 工具（如 Mockoon）進行初期測試
- 前期可開發 MVP 版本（例如一種 JSON 對應一個 GUI），逐步擴展 GUI 表單種類

如需，我可以幫你產出一份 PowerPoint 技術選型報告給主管使用，或範例專案結構，是否需要？



我想要理解React JSON Schema Form用法，為了做React JSON Schema Form開發Web GUI輸入json來使用api的實驗，我對React & React JSON Schema Form的用法不熟，你能教我嗎，或是你能給更簡單的範例：



已更新儲存的記憶

當然可以！這裡我會用最簡單的 **React + React JSON Schema Form (RJSF)** 範例一步步教你，讓你可以從 GUI 表單輸入 JSON，然後觸發 API 呼叫。

🔧 一、什麼是 React JSON Schema Form (RJSF) ？

RJSF 是一個由 JSON Schema 驅動的表單生成工具，可以讓你「用一份 JSON 定義表單結構」，自動產生表單 GUI。

例如：

```
{
  "title": "登入資訊",
  "type": "object",
  "properties": {
    "username": { "type": "string", "title": "帳號" },
    "password": { "type": "string", "title": "密碼" }
  }
}
```

會自動產生兩個輸入欄位「帳號」「密碼」。

二、快速入門範例

Step 1: 建立 React 專案

```
npx create-react-app json-form-demo
cd json-form-demo
npm install @rjsf/core axios
```

Step 2: 編輯 `App.js`

以下是最簡單的表單 + 呼叫 API 的範例：

```
import React from "react";
import Form from "@rjsf/core";
import axios from "axios";

const schema = {
  title: "API 輸入表單",
  type: "object",
  required: ["username", "password"],
  properties: {
    username: { type: "string", title: "帳號" },
    password: { type: "string", title: "密碼" },
  },
};

function App() {
  const onSubmit = async ({ formData }) => {
    console.log("要送出的資料:", formData);

    try {
      const response = await axios.post("https://your.api/endpoint", formData);
      console.log("API 回傳:", response.data);
      alert("送出成功!");
    } catch (error) {
```

```

    console.error("API 錯誤:", error);
    alert("送出失敗!");
  }
};

return (
  <div style={{ maxWidth: 500, margin: "auto" }}>
    <h1>JSON 表單 Demo</h1>
    <Form schema={schema} onSubmit={onSubmit} />
  </div>
);
}

export default App;

```



三、進階用法預告（等你熟悉基本後再碰）

功能	方法
自訂欄位 UI（如下拉選單、checkbox）	使用 `uiSchema`
欄位驗證	Schema 中加上 `"minLength"`, `"pattern"` 等條件
表單欄位動態顯示	加上 `if/then/else` 或用 state 控制 schema
上傳檔案	配合 custom widget 或用 base64 傳遞
整合認證 / token	用 Axios 加上 header（如 `Authorization: Bearer xxx`）

Bonus：表單 schema 產生器

如果你不會寫 JSON Schema，可以用這些工具：

- <https://jsonschema.net>
- <https://rjsf-team.github.io/react-jsonschema-form> (官方 Demo)

如果你需要我幫你產生一個對應你 API 輸入格式的 schema，我也可以幫你做！
是否要我根據你的 API JSON 結構寫一個表單範例？你只要提供：

```
{  
  "action": "create_user",  
  "name": "Jack",  
  "email": "jack@example.com",  
  "roles": ["admin", "dev"]  
}
```

我就能幫你產出對應 GUI 表單！是否要？



要！

```
{  
  "action": "create_user",  
  "name": "Jack",  
  "email": "jack@example.com",  
  "roles": ["admin", "dev"]  
}
```



太好了！根據你提供的 JSON，我幫你產生一個可直接用於 React JSON Schema Form 的 GUI 表單，並附上範例程式碼可立即實驗。



一、對應的 JSON Schema

```
const schema = {
  title: "建立使用者",
  type: "object",
  required: ["action", "name", "email", "roles"],
  properties: {
    action: {
      type: "string",
      title: "動作",
      enum: ["create_user"]
    },
    name: {
      type: "string",
      title: "姓名"
    },
    email: {
      type: "string",
      format: "email",
      title: "電子郵件"
    },
    roles: {
      type: "array",
      title: "角色",
      items: {
        type: "string",
        enum: ["admin", "dev", "ops", "guest"]
      },
      uniqueItems: true
    }
  }
};
```



二、可選 UI Schema (美化)



```
const uiSchema = {
  action: {
    "ui:disabled": true
  },
  roles: {
    "ui:widget": "checkboxes"
  }
};
```

三、完整 React 範例（可直接貼進 `App.js`）

```
import React from "react";
import Form from "@rjsf/core";
import axios from "axios";

const schema = {
  title: "建立使用者",
  type: "object",
  required: ["action", "name", "email", "roles"],
  properties: {
    action: {
      type: "string",
      title: "動作",
      enum: ["create_user"]
    },
    name: {
      type: "string",
      title: "姓名"
    },
    email: {
      type: "string",
      format: "email",
      title: "電子郵件"
    },
    roles: {
```

```

    type: "array",
    title: "角色",
    items: {
      type: "string",
      enum: ["admin", "dev", "ops", "guest"]
    },
    uniqueItems: true
  }
}
};

const uiSchema = {
  action: {
    "ui:disabled": true
  },
  roles: {
    "ui:widget": "checkboxes"
  }
};

function App() {
  const onSubmit = async ({ formData }) => {
    console.log("送出的 JSON:", formData);

    try {
      const response = await axios.post("https://your.api/endpoint", formData);
      console.log("API 回應:", response.data);
      alert("送出成功!");
    } catch (err) {
      console.error("API 錯誤:", err);
      alert("送出失敗!");
    }
  };

  return (
    <div style={{ maxWidth: 600, margin: "auto" }}>
      <h2>使用者建立表單</h2>
      <Form schema={schema} uiSchema={uiSchema} onSubmit={onSubmit} />
    </div>
  );
}

```

```
export default App;
```

四、啟動方式

1. 在專案資料夾中啟動伺服器：

```
npm start
```

2. 用瀏覽器打開 `http://localhost:3000`，你就會看到 GUI 表單！

下一步可加強項目

功能	可用方法
欄位預設值	加入 <code>const formData = { action: "create_user" }</code> 傳給 <code><Form></code>
欄位動態變更	改用 <code>useState</code> 動態變 <code>schema</code>
權限驗證、登入等	用 <code>React Context</code> 或加載 <code>JWT</code>

如果你還有下一個 JSON 結構想要轉 GUI，我也可以幫你設計對應 `schema`，或指導你怎麼自己寫！是否需要？



以繁體中文詳細說明:

我想要理解Ant Design + React Hook Form用法，為了做Ant Design + React Hook Form開發Web GUI輸入json來使用api的實驗，我對React & Ant Design + React Hook Form的用法不熟，你能教我嗎，或是你能給更簡單的範例:



非常好！你希望改用 **Ant Design + React Hook Form** 來開發 Web GUI，達到輸入 JSON 並送出 API 的功能。這比 React JSON Schema Form 更靈活、可客製化，但也稍微複雜一些。我會用最簡單的範例手把手教你：

一、React Hook Form + Ant Design 是什麼？

技術	功能
React Hook Form	輕量、效能高的表單處理框架，可控制欄位狀態、驗證、送出
Ant Design	UI 元件庫，提供美觀表單、按鈕、輸入框、layout

這兩者可搭配使用：Antd 負責**表單外觀**，Hook Form 負責**表單邏輯**（狀態、驗證、送出）。

二、先安裝必要套件

```
npm install antd react-hook-form
```

“※ 如用 TypeScript 可再裝 `@types/react` 等套件”

三、最簡單的 Ant Design + React Hook Form 實作

假設我們要輸入以下 JSON：

```
{
  "action": "create_user",
  "name": "Jack",
  "email": "jack@example.com",
  "roles": ["admin", "dev"]
}
```

我們要從 GUI 表單輸入這些欄位，轉成這個 JSON 並送出給 API。

App.js 實作範例

```
import React from "react";
import { useForm, Controller } from "react-hook-form";
import { Input, Button, Select, Form, message } from "antd";
import axios from "axios";

const { Option } = Select;

const defaultValues = {
  action: "create_user",
  name: "",
  email: "",
  roles: []
};

function App() {
  const { control, handleSubmit } = useForm({ defaultValues });

  const onSubmit = async (data) => {
```

```

console.log("送出 JSON:", data);
try {
  const res = await axios.post("https://your.api/endpoint", data);
  console.log("API 回傳:", res.data);
  message.success("送出成功");
} catch (err) {
  console.error("API 錯誤:", err);
  message.error("送出失敗");
}
};

return (
  <div style={{ maxWidth: 600, margin: "auto", paddingTop: 40 }}>
    <h2>建立使用者</h2>
    <Form layout="vertical" onFinish={handleSubmit(onSubmit)}>
      <Form.Item label="動作 (固定)">
        <Controller
          name="action"
          control={control}
          render={({ field }) => <Input {...field} disabled />}
        />
      </Form.Item>

      <Form.Item label="姓名">
        <Controller
          name="name"
          control={control}
          rules={{ required: "請輸入姓名" }}
          render={({ field }) => <Input {...field} />}
        />
      </Form.Item>

      <Form.Item label="Email">
        <Controller
          name="email"
          control={control}
          rules={{ required: "請輸入 email" }}
          render={({ field }) => <Input {...field} />}
        />
      </Form.Item>
    </Form>
  </div>
);

```

```

    <Form.Item label="角色">
      <Controller
        name="roles"
        control={control}
        render={({ field }) => (
          <Select
            {...field}
            mode="multiple"
            placeholder="選擇角色"
            onChange={(value) => field.onChange(value)}
          >
            <Option value="admin">admin</Option>
            <Option value="dev">dev</Option>
            <Option value="ops">ops</Option>
            <Option value="guest">guest</Option>
          </Select>
        )}
      />
    </Form.Item>

    <Form.Item>
      <Button type="primary" htmlType="submit">送出</Button>
    </Form.Item>
  </Form>
</div>
);
}

export default App;

```

四、補充說明

項目	說明
`Controller`	將 Antd 元件（非原生 input）與 React Hook Form 綁定

項目	說明
<code>`defaultValues`</code>	定義表單預設值
<code>`axios.post()`</code>	送出 JSON 到後端 API
<code>`Select mode="multiple"`</code>	多選下拉選單，用於 <code>`roles`</code> 欄位
<code>`disabled`</code>	禁止編輯 <code>`action`</code> 欄位（因為固定為 <code>`create_user`</code> ）

五、可擴充功能（進階時再學）

功能	作法
欄位驗證	在 <code>`rules`</code> 加上 <code>`required`</code> 、 <code>`pattern`</code> 等
顯示錯誤訊息	使用 <code>`formState.errors`</code> 顯示錯誤提示
用 TypeScript 強型別	使用 <code>`useForm<FormValues>()`</code>
表單動態增減欄位	搭配 <code>`useFieldArray()`</code>
整合登入 / SSO	在 <code>`axios`</code> 中加上 <code>`Authorization`</code> header

如果你有別的 JSON 格式，也可以再貼給我，我幫你寫對應的 Antd + Hook Form GUI！

是否需要我幫你生成另一個例子，或協助你部署到本地、K8s？



以繁體中文詳細說明：
我想要理解Formik / React Final Form用法，為了做Formik /

React Final Form開發Web GUI輸入json來使用api的實驗，我對React & Formik / React Final Form的用法不熟，你能教我嗎，或是你能給更簡單的範例：



當然可以！你想學會使用 **Formik** 或 **React Final Form** 來開發 Web GUI 表單，實作「輸入 JSON 呼叫 API」的功能，我會以繁體中文教你，並從最簡單的實例開始。

一、Formik vs React Final Form 差異概覽

特點	Formik	React Final Form
學習曲線	容易上手	更進階但彈性更高
文件完整性	優秀，入門快	強大但偏向資深開發者
控制方式	比較封閉	全部自訂可控
適合情境	中小型表單、快速開發	需要細粒度控制、大型專案

🟢 建議你從 **Formik** 入門，上手後可視需求轉向 React Final Form。

二、Formik 實作：「建立使用者」JSON 表單

✅ Step 1：安裝 Formik

```
npm install formik
```

✅ Step 2：簡單範例程式（可貼入 `App.js`）

這會產生表單，送出時組成你指定的 JSON 結構：

```
import React from "react";
import { useFormik } from "formik";
import axios from "axios";

function App() {
  const formik = useFormik({
    initialValues: {
      action: "create_user",
      name: "",
      email: "",
      roles: []
    },
    onSubmit: async (values) => {
      console.log("送出 JSON:", values);
      try {
        const res = await axios.post("https://your.api/endpoint", values);
        alert("送出成功!");
      } catch (err) {
        console.error("API 錯誤:", err);
        alert("送出失敗!");
      }
    }
  });

  return (
    <div style={{ maxWidth: 500, margin: "auto", paddingTop: 30 }}>
      <h2>使用者建立表單 (Formik)</h2>
      <form onSubmit={formik.handleSubmit}>
        <div>
          <label>動作：</label>
          <input name="action" value={formik.values.action} readOnly />
        </div>

        <div>
```

```

    <label>姓名：</label>
    <input
      name="name"
      onChange={formik.handleChange}
      value={formik.values.name}
    />
  </div>

  <div>
    <label>Email：</label>
    <input
      name="email"
      onChange={formik.handleChange}
      value={formik.values.email}
    />
  </div>

  <div>
    <label>角色（逗號分隔）：</label>
    <input
      name="roles"
      onChange={(e) => {
        formik.setFieldValue("roles", e.target.value.split(","));
      }}
      value={formik.values.roles.join(",")}
    />
  </div>

  <button type="submit">送出</button>
</form>
</div>
);
}

export default App;

```

三、React Final Form 簡單範例

✅ Step 1：安裝

```
npm install react-final-form
```

✅ Step 2：範例程式

```
import React from "react";
import { Form, Field } from "react-final-form";
import axios from "axios";

function App() {
  const onSubmit = async (values) => {
    console.log("送出 JSON:", values);
    try {
      const res = await axios.post("https://your.api/endpoint", values);
      alert("送出成功!");
    } catch (err) {
      console.error("API 錯誤:", err);
      alert("送出失敗!");
    }
  };

  return (
    <div style={{ maxWidth: 500, margin: "auto", padding: 30 }}>
      <h2>使用者建立表單 (React Final Form)</h2>
      <Form
        onSubmit={onSubmit}
        initialValues={{
          action: "create_user",
          name: "",
          email: "",
          roles: []
        }}
      />
    </div>
  );
}
```

```

render=(({ handleSubmit, values }) => (
  <form onSubmit={handleSubmit}>
    <div>
      <label>動作：</label>
      <Field name="action" component="input" readOnly />
    </div>

    <div>
      <label>姓名：</label>
      <Field name="name" component="input" />
    </div>

    <div>
      <label>Email：</label>
      <Field name="email" component="input" />
    </div>

    <div>
      <label>角色（逗號分隔）：</label>
      <Field
        name="roles"
        component="input"
        parse={(val) => val.split(",")}
        format={(val) => (Array.isArray(val) ? val.join(",") : "")}
      />
    </div>

    <button type="submit">送出</button>
  </form>
)
  </div>
);
}

export default App;

```



四、你該怎麼選？

需求	推薦
希望快速產出、文件好讀	✓ Formik
須整合 Ant Design / Material UI 精細控制表單互動	✓ React Hook Form
高度客製化、效能導向表單（大型企業 App）	✓ React Final Form



是否需要我幫你...

- 用 Formik 或 Final Form 寫出你其他 JSON 結構的表單？
- 整合 Ant Design 做出更美觀的 UI？
- 實作表單驗證（必填、格式限制等）？
- 建出整套專案模板（含 K8s、API 封裝）？

歡迎告訴我，我可以一步步帶你做！



你能給我目前對話內容的投影片大綱嗎，要有React JSON Schema Form、Ant Design + React Hook Form、Formik / React Final Form，三者的比較



當然可以，以下是你目前實驗開發 GUI 輸入 JSON 呼叫 API 的需求，針對 **React JSON Schema Form、Ant Design + React Hook Form、Formik / React Final Form** 三種

方案的說明與比較，整理成一份**投影片大綱**，可用來對主管報告或自我整理技術選型。



投影片大綱：React 表單框架比較與選型



Slide 1 | 開場：專案目標

- 目的：開發一個 Web GUI，取代直接輸入 JSON 呼叫 API
 - 要求：
 - 自動或可視化輸入 JSON 結構
 - 高可用（可部署至 K8s）
 - 可擴展（未來加入表單驗證、RBAC、SSO）
 - 輕量好維護
-



Slide 2 | 選項一：React JSON Schema Form (RJSF)

- 特色：
 - 可根據 JSON Schema 自動產生表單
 - 快速建置、適合固定格式的 API 輸入
- 適合情境：
 - JSON 結構穩定
 - 無需高度客製 UI



- 範例功能：
 - 自動產生帳號、密碼、角色選單
 - 配合 onSubmit 傳送完整 JSON 結構
-

Slide 3 | 選項二：Ant Design + React Hook Form

- 特色：
 - Antd 提供美觀 UI、Hook Form 管理狀態與驗證
 - 高客製化，適合企業內部工具開發
 - 適合情境：
 - 表單行為需細節控制（動態欄位、條件顯示）
 - 需整合 Antd Layout、Modal、Tabs 等元件
 - 範例功能：
 - 角色多選下拉（`Select[multiple]`）
 - 欄位驗證（email 格式、必填）
-

Slide 4 | 選項三：Formik / React Final Form

- 特色比較：

框架	優點	適合
Formik	上手簡單、文檔齊全	中小型表單
Final Form	可精細控制每個欄位狀態，效能高	大型/複雜專案

- 適合情境：
 - 不需 UI 元件庫，專注表單資料處理
 - 欲結合 Material UI / Tailwind / 客製畫面

Slide 5 | 三種方案總整理比較表

特性	RJSF	Antd + RHF	Formik / Final Form
快速產出	✅ 非常快	❌ 需手刻欄位	✅ Formik 快、Final 較難
表單驗證與控制	普通（靠 JSON Schema）	✅ 非常靈活	✅ 自訂程度高
UI 外觀	中等（自帶但難自訂）	✅ 完整 Antd 樣式	❌ 手刻或搭配其他 UI 庫
自訂欄位邏輯	❌ 限於 schema	✅ 完整掌控	✅ 高控制權
適合 JSON 輸入	✅ 最自然	✅ 適合 API 操作	✅ 輸出可組合成任意結構
學習曲線	低	中	Formik 中、Final 高
適合場景	固定 API GUI 工具	企業 Web 工具開發	客製化需求高的內部平台

Slide 6 | 建議選型策略

-  若要**快速驗證 JSON 表單可行性** → 使用 **RJSF**
 -  若要**整合內部 UI 樣式、彈性需求多** → 使用 **Ant Design + React Hook Form**
 -  若要**純表單資料處理邏輯強控制** → 使用 **Formik (初期) / Final Form (高階)**
-

Slide 7 | 後續建議與延伸

- 整合 API 驗證、token (JWT/Session)
 - 加入欄位動態顯示、切換模式 (進階)
 - 佈署到 K8s 的容器化設計
 - 考慮加入 CI/CD、自動化測試流程
-

如你需要，我也可以幫你做出實際的投影片 (PowerPoint 或 Google Slides 可用格式)，你只需告訴我格式偏好。是否要幫你產出？