`pages/app/detail` 文件夹下的代码是 Next.js 项目中用于展示应用详情和设置的页面。以下是对您上传的文件的分析：

### index.tsx

- \*\*用途\*\*：作为应用详情页面的入口组件，渲染应用的具体信息和操作界面。

- \*\*特点\*\*：

- 使用了动态导入来按需加载组件，例如 `CreateModal`、`EditFolderModal` 和 `HttpEditModal`。

- 通过 `useContextSelector` 钩子从 `AppContext` 中获取应用的状态和数据。

- 组件 `MyApps` 负责渲染应用列表和相关操作，如创建、编辑、删除应用。

- 支持应用类型的筛选，并通过 `LightRowTabs` 组件展示。

- 有创建新应用的菜单 `MyMenu`，允许用户选择创建不同类型的应用。

### CreateModal.tsx

- \*\*用途\*\*：创建新应用的模态框，允许用户输入应用的名称、选择头像和模板。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 支持文件选择和上传，用于设置应用的头像。

- 提供了多种应用模板供用户选择。

- 使用了 `useRequest` 钩子来处理应用创建的异步请求。

### context.tsx

- \*\*用途\*\*：定义了应用详情页面的上下文 `AppContext`，用于状态共享和状态管理。

- \*\*特点\*\*：

- 使用了 `createContext` 来创建上下文。

- 上下文提供了应用的加载状态、应用列表、当前选中的应用等信息。

- 通过 `useRequest2` 钩子来加载和更新应用列表。

### List.tsx

- \*\*用途\*\*：展示应用列表的组件。

- \*\*特点\*\*：

- 使用了 Chakra UI 的 `Grid` 组件来布局应用卡片。

- 支持应用的拖拽排序。

- 提供了应用的编辑、删除和权限配置的操作。

- 使用了动态导入的 `EditResourceModal` 和 `ConfigPerModal` 等组件来处理应用的编辑和权限配置。

### TypeTag.tsx

- \*\*用途\*\*：用于显示应用类型的标签组件。

- \*\*特点\*\*：

- 根据应用类型展示不同的图标和颜色。

### HttpPluginEditModal.tsx

- \*\*用途\*\*：用于编辑 HTTP 插件的模态框。

- \*\*特点\*\*：

- 支持从 URL 导入 API Schema。

- 允许用户自定义 HTTP 请求头。

- 展示了 API 的列表，包括方法、路径和描述。

### InfoModal.tsx

- \*\*用途\*\*：用于编辑应用信息的模态框。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 支持应用名称、头像、简介的编辑。

- 提供了应用权限和协作者的管理功能。

### ToolSelectModal.tsx

- \*\*用途\*\*：用于选择工具插件的模态框。

- \*\*特点\*\*：

- 支持从系统插件和团队插件中选择。

- 允许用户搜索和筛选可用的工具插件。

- 提供了工具插件的参数配置功能。

### AppCard.tsx

- \*\*用途\*\*：展示单个应用卡片的组件。

- \*\*特点\*\*：

- 显示应用的基本信息，如名称、头像、简介。

- 提供了应用的操作按钮，如聊天、设置、删除等。

- 支持应用的拖拽排序。

这些组件共同构成了应用详情页面，允许用户浏览、创建、编辑和管理应用。每个组件都有其特定的职责，通过上下文 `AppContext` 共享和传递状态。如果您需要对特定代码段或逻辑的更深入分析，请告知。

继续分析 `pages/app/detail` 文件夹下剩余的文件：

### constants.tsx

- \*\*用途\*\*：定义了一些常量样式，供页面内的组件使用。

- \*\*特点\*\*：

- `cardStyles`：定义了卡片的基本样式，如圆角和背景颜色。

- `workflowBoxStyles`：定义了工作流盒子的样式，包括定位、方向和背景颜色。

- `publishStatusStyle`：定义了发布状态的样式，包括未发布和已发布的文本和颜色。

### useChatTest.tsx

- \*\*用途\*\*：定义了一个自定义钩子 `useChatTest`，用于处理聊天测试相关的逻辑。

- \*\*特点\*\*：

- 使用了 `useUserStore` 来获取用户信息。

- 通过 `streamFetch` 发起流式请求，获取聊天数据。

- 利用 `useMemoizedFn` 来记忆函数，避免不必要的重新渲染。

### PublishHistoriesSlider.tsx

- \*\*用途\*\*：发布历史记录的侧边栏组件。

- \*\*特点\*\*：

- 使用了 `useScrollPagination` 钩子来实现滚动分页。

- 展示了应用发布的历史版本，允许用户预览和回退到旧版本。

### RouteTab.tsx

- \*\*用途\*\*：应用详情页面的路由标签组件。

- \*\*特点\*\*：

- 使用了 `useRouter` 和 `useTranslation` 钩子。

- 根据应用的类型和权限显示不同的标签。

### SimpleEdit.tsx

- \*\*用途\*\*：简单编辑页面的入口组件。

- \*\*特点\*\*：

- 包含头部组件 `Header` 和根据不同标签显示不同内容的布局。

- 使用了 `useBeforeunload` 钩子来防止用户在未保存时离开页面。

### Header.tsx

- \*\*用途\*\*：应用详情页面的头部组件。

- \*\*特点\*\*：

- 包含了返回按钮、应用信息、发布确认和发布按钮等。

- 使用了 `useInterval` 钩子来定期保存工作流数据。

### OutLink.tsx

- \*\*用途\*\*：处理应用的外链发布设置。

- \*\*特点\*\*：

- 包含了不同的发布渠道选项，如分享链接、API请求等。

- 使用了 `MyRadio` 组件来选择不同的发布方式。

### API.tsx

- \*\*用途\*\*：API 发布设置页面。

- \*\*特点\*\*：

- 显示了应用的 API 密钥信息。

- 使用了 `ApiKeyTable` 组件来展示和管理 API 密钥。

### FeiShuEditModal.tsx

- \*\*用途\*\*：飞书发布设置的模态框。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 允许用户创建或编辑飞书应用的发布设置。

### FeiShu.tsx

- \*\*用途\*\*：飞书应用详情页面。

- \*\*特点\*\*：

- 展示了飞书应用的列表，并允许用户创建、编辑和删除飞书应用。

### SelectUsingWayModal.tsx

- \*\*用途\*\*：选择使用方式的模态框。

- \*\*特点\*\*：

- 允许用户选择链接、iframe 或 script 作为集成方式。

- 提供了相应的代码示例和复制功能。

### Edit.tsx

- \*\*用途\*\*：应用编辑页面。

- \*\*特点\*\*：

- 用于展示和编辑应用的详细信息。

- 使用了 `useMount` 钩子来加载数据集和应用详情。

### EditForm.tsx

- \*\*用途\*\*：应用编辑表单组件。

- \*\*特点\*\*：

- 提供了应用的 AI 设置、数据集选择、工具选择等编辑功能。

- 使用了 `react-hook-form` 和 `useFieldArray` 钩子来管理表单状态。

### AppCard.tsx

- \*\*用途\*\*：应用卡片组件，展示单个应用的基本信息和操作。

- \*\*特点\*\*：

- 显示应用的名称、头像、简介和权限设置。

- 提供了聊天、设置、删除等操作按钮。

### ToolSelectModal.tsx

- \*\*用途\*\*：工具选择模态框，用于添加或配置工具插件。

- \*\*特点\*\*：

- 允许用户在系统插件和团队插件之间切换。

- 提供了搜索功能，帮助用户快速找到所需的工具插件。

- 支持添加工具到工作流，并允许用户配置工具的参数。

这些文件共同构成了应用详情页面的功能，包括应用信息展示、编辑、发布设置、权限管理等。每个组件和钩子都有其特定的职责，通过相互协作来实现页面的完整功能。如果您需要对特定代码段或逻辑的更深入分析，请告知。

由于您提供的文件数量较多，我将继续分析剩余的文件：

### index.tsx (位于 SimpleEdit 组件内部)

- \*\*用途\*\*：作为应用编辑页面的入口点，提供了应用编辑的界面和逻辑。

- \*\*特点\*\*：

- 使用了 `useState` 来初始化应用表单的状态。

- 利用 `useBeforeunload` 钩子来防止用户在未保存更改时离开页面。

- 根据当前激活的标签 (`currentTab`) 渲染不同的内容，例如编辑表单、发布渠道或日志。

### Header.tsx

- \*\*用途\*\*：提供了应用编辑页面的头部组件，包括应用的基本信息和操作按钮。

- \*\*特点\*\*：

- 包含了路径导航 (`FolderPath`)、发布确认和发布按钮。

- 使用了 `useRequest2` 钩子来处理应用路径的获取。

- 通过 `isPublished` 状态来确定应用是否已经发布，并据此启用或禁用发布按钮。

### ChatTest.tsx

- \*\*用途\*\*：提供了一个聊天测试界面，允许用户测试应用的聊天功能。

- \*\*特点\*\*：

- 使用了 `useSafeState` 钩子来安全地设置和更新状态。

- 利用 `useEffect` 钩子来响应应用表单数据的变化，并更新工作流数据。

- 通过 `useChatTest` 钩子来获取聊天框组件和重置聊天框的函数。

### EditForm.tsx

- \*\*用途\*\*：提供了应用编辑表单的详细界面，允许用户配置应用的各种设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 的 `useFieldArray` 来管理表单字段数组。

- 包括了 AI 设置、数据集选择、工具选择等多个配置部分。

- 提供了动态组件加载，例如 `DatasetSelectModal` 和 `ToolSelectModal`，以供用户选择和管理。

### AppCard.tsx

- \*\*用途\*\*：显示应用卡片的组件，提供了应用的基本信息和操作菜单。

- \*\*特点\*\*：

- 显示应用的头像、名称和简介。

- 提供了聊天、设置和删除按钮。

- 使用了 `useRequest2` 钩子来处理应用转换为工作流的异步请求。

### ToolSelectModal.tsx

- \*\*用途\*\*：工具选择模态框，允许用户选择和配置工具插件。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 支持从系统插件和团队插件中选择工具。

- 提供了工具参数配置的界面，允许用户输入必要的参数。

### SelectUsingWayModal.tsx

- \*\*用途\*\*：选择使用方式的模态框，允许用户选择不同的集成方式。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了链接、iframe 和 script 三种集成方式的选择。

- 允许用户根据选择的集成方式复制相应的代码。

### Edit.tsx

- \*\*用途\*\*：应用编辑页面的主体部分，展示应用的详细信息和配置选项。

- \*\*特点\*\*：

- 使用了 `useMount` 钩子来加载所有数据集。

- 根据应用的版本信息，可能将旧版本的工作流转换为新版本。

- 渲染了 `AppCard`、`EditForm` 和 `ChatTest` 组件来提供完整的应用编辑界面。

### FeiShuEditModal.tsx

- \*\*用途\*\*：飞书应用编辑模态框，允许用户创建或编辑飞书应用的设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了应用名称、QPM限制、最大使用点数、过期时间等配置选项。

### FeiShu.tsx

- \*\*用途\*\*：飞书应用详情页面，展示了飞书应用列表和相关操作。

- \*\*特点\*\*：

- 使用了 `useQuery` 钩子来获取飞书应用列表。

- 提供了创建链接、编辑和删除飞书应用的功能。

### Share.tsx

- \*\*用途\*\*：分享链接设置页面，允许用户创建和管理分享链接。

- \*\*特点\*\*：

- 使用了 `useQuery` 钩子来获取分享链接列表。

- 提供了创建链接、编辑和删除链接的功能。

### EditLinkModal.tsx

- \*\*用途\*\*：编辑链接模态框，允许用户创建或编辑分享链接的设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了链接名称、过期时间、最大使用点数等配置选项。

这些文件构成了 Next.js 项目中 `pages/app/detail` 文件夹的核心功能，涵盖了应用的详细信息展示、编辑、发布配置、分享链接管理等。每个组件都通过特定的状态管理和数据处理逻辑来实现其功能。如果您需要对特定代码段或逻辑的更深入分析，请告知。

在您提供的文件中，我注意到还有几个文件没有分析。以下是对这些文件的分析：

### constants.tsx (已在之前的回答中分析)

### PublishHistoriesSlider.tsx

- \*\*用途\*\*：展示应用发布历史的侧边栏组件。

- \*\*特点\*\*：

- 使用了 `useScrollPagination` 钩子来实现滚动分页。

- 允许用户查看应用的历史发布版本，支持预览和回退到特定版本。

### RouteTab.tsx

- \*\*用途\*\*：提供应用详情页面的路由标签切换功能。

- \*\*特点\*\*：

- 根据应用的类型和用户权限显示不同的标签页选项。

- 使用了 `useRouter` 和 `useTranslation` 钩子。

### SimpleEdit.tsx

- \*\*用途\*\*：简单编辑页面的入口组件。

- \*\*特点\*\*：

- 包含应用表单的初始化状态。

- 使用了 `useBeforeunload` 钩子来防止用户在未保存更改时离开页面。

### Header.tsx

- \*\*用途\*\*：应用详情页面的头部组件。

- \*\*特点\*\*：

- 显示应用的路径和名称。

- 提供了发布应用的功能，包括发布确认和发布历史。

### ChatTest.tsx

- \*\*用途\*\*：聊天测试组件，允许用户测试应用的聊天功能。

- \*\*特点\*\*：

- 使用了 `useSafeState` 和 `useEffect` 钩子。

- 通过 `useChatTest` 钩子获取聊天框组件和相关功能。

### EditForm.tsx

- \*\*用途\*\*：应用编辑表单组件。

- \*\*特点\*\*：

- 提供了应用的 AI 设置、数据集选择、工具选择等编辑功能。

- 使用了 `react-hook-form` 和 `useFieldArray` 钩子。

### AppCard.tsx

- \*\*用途\*\*：展示应用卡片的组件。

- \*\*特点\*\*：

- 显示应用的基本信息，如名称、头像、简介。

- 提供了应用操作，如聊天、设置、删除等。

### ToolSelectModal.tsx

- \*\*用途\*\*：工具选择模态框。

- \*\*特点\*\*：

- 允许用户选择和配置工具插件。

- 使用了 `react-hook-form` 来管理表单状态。

### SelectUsingWayModal.tsx

- \*\*用途\*\*：选择使用方式的模态框。

- \*\*特点\*\*：

- 允许用户选择不同的集成方式（链接、iframe、script）。

- 提供了相应的配置选项和代码示例。

### Edit.tsx

- \*\*用途\*\*：应用编辑页面的主体组件。

- \*\*特点\*\*：

- 使用了 `useMount` 钩子来加载数据集和应用详情。

- 根据应用的版本信息，可能将旧版本的工作流转换为新版本。

### FeiShuEditModal.tsx

- \*\*用途\*\*：飞书应用编辑模态框。

- \*\*特点\*\*：

- 提供了创建或编辑飞书应用的界面。

- 使用了 `react-hook-form` 来管理表单状态。

### FeiShu.tsx

- \*\*用途\*\*：飞书应用详情页面。

- \*\*特点\*\*：

- 展示了飞书应用列表，允许用户进行创建、编辑和删除操作。

### Share.tsx

- \*\*用途\*\*：分享链接设置页面。

- \*\*特点\*\*：

- 允许用户创建和管理分享链接。

- 提供了链接的编辑和删除功能。

### EditLinkModal.tsx

- \*\*用途\*\*：编辑链接模态框。

- \*\*特点\*\*：

- 允许用户设置分享链接的名称、过期时间等属性。

以上是您提供的文件的分析总结。如果您需要进一步的详细分析或有特定问题，请告知。

在您提供的文件中，我注意到还有几个文件没有详细分析。以下是对这些文件的分析：

### index.tsx (SimpleEdit 组件内部)

- \*\*用途\*\*：作为应用编辑页面的入口组件，提供应用编辑的界面。

- \*\*特点\*\*：

- 使用了 `useState` 来初始化应用表单状态。

- 利用 `useBeforeunload` 钩子来防止用户在未保存更改时离开页面。

- 根据当前激活的标签 (`currentTab`) 渲染不同的内容，例如编辑表单、发布渠道或日志。

### Header.tsx

- \*\*用途\*\*：应用编辑页面的头部组件，提供应用的基本信息和导航。

- \*\*特点\*\*：

- 使用了 `useRequest2` 来获取应用的路径信息。

- 通过 `isPublished` 状态来确定应用是否已经发布，并据此启用或禁用发布按钮。

- 提供了发布确认和发布历史的交互。

### ChatTest.tsx

- \*\*用途\*\*：提供应用聊天功能的测试界面。

- \*\*特点\*\*：

- 使用了 `useSafeState` 钩子来管理组件状态。

- 利用 `useEffect` 钩子来响应应用表单数据的变化。

- 通过 `useChatTest` 钩子来获取聊天框组件和重置聊天功能。

### EditForm.tsx

- \*\*用途\*\*：应用编辑表单的详细界面，允许用户配置应用的各种设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 的 `UseFormReturn` 和 `useFieldArray` 来管理表单状态和动态字段。

- 包括了 AI 设置、数据集选择、工具选择等多个配置部分。

- 提供了动态组件加载，例如 `DatasetSelectModal` 和 `ToolSelectModal`。

### AppCard.tsx

- \*\*用途\*\*：展示应用卡片的组件，提供了应用的基本信息和操作。

- \*\*特点\*\*：

- 显示应用的名称、头像、简介和权限设置。

- 提供了聊天、设置、删除等操作按钮。

- 使用了 `useRequest2` 钩子来处理应用转换为工作流的异步请求。

### ToolSelectModal.tsx

- \*\*用途\*\*：工具选择模态框，允许用户选择和配置工具插件。

- \*\*特点\*\*：

- 支持从系统插件和团队插件中选择工具。

- 提供了搜索功能，帮助用户快速找到所需的工具插件。

- 允许用户配置工具的参数，并添加到应用中。

### SelectUsingWayModal.tsx

- \*\*用途\*\*：选择使用方式的模态框，允许用户选择不同的集成方式。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了链接、iframe 和 script 三种集成方式的选择。

- 允许用户根据选择的集成方式复制相应的代码。

### Edit.tsx

- \*\*用途\*\*：应用编辑页面的主体部分，展示应用的详细信息和配置选项。

- \*\*特点\*\*：

- 使用了 `useMount` 钩子来加载所有数据集和应用详情。

- 根据应用的版本信息，可能将旧版本的工作流转换为新版本。

- 渲染了 `AppCard`、`EditForm` 和 `ChatTest` 组件来提供完整的应用编辑界面。

### FeiShuEditModal.tsx

- \*\*用途\*\*：飞书应用编辑模态框，允许用户创建或编辑飞书应用的设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了应用名称、QPM限制、最大使用点数、过期时间等配置选项。

### FeiShu.tsx

- \*\*用途\*\*：飞书应用详情页面，展示了飞书应用列表和相关操作。

- \*\*特点\*\*：

- 使用了 `useQuery` 钩子来获取飞书应用列表。

- 提供了创建链接、编辑和删除飞书应用的功能。

### Share.tsx

- \*\*用途\*\*：分享链接设置页面，允许用户创建和管理分享链接。

- \*\*特点\*\*：

- 使用了 `useQuery` 钩子来获取分享链接列表。

- 提供了创建链接、编辑和删除链接的功能。

### EditLinkModal.tsx

- \*\*用途\*\*：编辑链接模态框，允许用户创建或编辑分享链接的设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 提供了链接名称、过期时间、最大使用点数等配置选项。

以上是对您提供的文件的分析总结。如果您需要进一步的详细分析或有特定问题，请告知。

在您提供的文件中，似乎已经覆盖了 `pages/app/detail` 文件夹下的大部分组件。不过，我会确保没有遗漏任何文件，并继续分析剩余的文件：

### index.tsx (SimpleEdit 组件内部)

- \*\*用途\*\*：作为应用编辑页面的入口组件，提供应用编辑的界面。

- \*\*特点\*\*：

- 使用了 `useState` 来初始化应用表单状态。

- 使用 `useBeforeunload` 钩子来防止用户在未保存更改时离开页面。

- 根据当前激活的标签 (`currentTab`) 渲染不同的内容，例如编辑表单、发布渠道或日志。

### InfoModal.tsx

- \*\*用途\*\*：应用信息编辑模态框，允许用户编辑应用的基本信息和权限设置。

- \*\*特点\*\*：

- 使用了 `react-hook-form` 来管理表单状态。

- 支持应用名称、头像、简介和默认权限的编辑。

- 提供了协作者管理功能，包括获取协作者列表、更新协作者权限和删除协作者。

### TagsEditModal.tsx

- \*\*用途\*\*：标签编辑模态框，允许用户编辑应用的团队标签。

- \*\*特点\*\*：

- 使用了 `@chakra-ui/react` 的组件来构建用户界面。

- 支持搜索、选择和取消选择团队标签。

### context.tsx

- \*\*用途\*\*：定义了应用详情页面的上下文 `AppContext`，用于状态共享和状态管理。

- \*\*特点\*\*：

- 使用了 `createContext` 来创建上下文。

- 上下文提供了应用的加载状态、应用列表、当前选中的应用等信息。

- 通过 `useRequest2` 钩子来加载和更新应用列表。

### useChatTest.tsx

- \*\*用途\*\*：自定义钩子，用于处理聊天测试相关的逻辑。

- \*\*特点\*\*：

- 使用了 `useUserStore` 来获取用户信息。

- 通过 `streamFetch` 发起流式请求，获取聊天数据。

- 利用 `useMemoizedFn` 来记忆函数，避免不必要的重新渲染。

### PublishHistoriesSlider.tsx

- \*\*用途\*\*：发布历史记录的侧边栏组件。

- \*\*特点\*\*：

- 使用了 `useScrollPagination` 钩子来实现滚动分页。

- 允许用户查看应用的历史发布版本，支持预览和回退到特定版本。

### RouteTab.tsx

- \*\*用途\*\*：提供应用详情页面的路由标签切换功能。

- \*\*特点\*\*：

- 根据应用的类型和用户权限显示不同的标签页选项。

- 使用了 `useRouter` 和 `useTranslation` 钩子。

### Edit.tsx

- \*\*用途\*\*：应用编辑页面的主体组件。

- \*\*特点\*\*：

- 使用了 `useMount` 钩子来加载数据集和应用详情。

- 根据应用的版本信息，可能将旧版本的工作流转换为新版本。

- 渲染了 `AppCard`、`EditForm` 和 `ChatTest` 组件来提供完整的应用编辑界面。

### FeiShuEditModal.tsx

- \*\*用途\*\*：飞书应用编辑模态框。

- \*\*特点\*\*：

- 提供了创建或编辑飞书应用的界面。

- 使用了 `react-hook-form` 来管理表单状态。

### FeiShu.tsx

- \*\*用途\*\*：飞书应用详情页面。

- \*\*特点\*\*：

- 展示了飞书应用列表，允许用户进行创建、编辑和删除操作。

### Share.tsx

- \*\*用途\*\*：分享链接设置页面。

- \*\*特点\*\*：

- 允许用户创建和管理分享链接。

- 提供了链接的编辑和删除功能。

### EditLinkModal.tsx

- \*\*用途\*\*：编辑链接模态框。

- \*\*特点\*\*：

- 允许用户设置分享链接的名称、过期时间等属性。

### ToolSelectModal.tsx

- \*\*用途\*\*：工具选择模态框。

- \*\*特点\*\*：

- 允许用户选择和配置工具插件。

- 使用了 `react-hook-form` 来管理表单状态。

### SelectUsingWayModal.tsx

- \*\*用途\*\*：选择使用方式的模态框。

- \*\*特点\*\*：

- 允许用户选择不同的集成方式（链接、iframe、script）。

- 提供了相应的配置选项和代码示例。

### AppCard.tsx

- \*\*用途\*\*：展示应用卡片的组件。

- \*\*特点\*\*：

- 显示应用的基本信息，如名称、头像、简介。

- 提供了应用操作，如聊天、设置、删除等。

### EditForm.tsx

- \*\*用途\*\*：应用编辑表单组件。

- \*\*特点\*\*：

- 提供了应用的 AI 设置、数据集选择、工具选择等编辑功能。

- 使用了 `react-hook-form` 和 `useFieldArray` 钩子。

以上是您提供的文件的分析总结。现在看起来已经覆盖了所有文件。如果您需要进一步的详细分析或有特定问题，请告知。