目录

1. 总结 (Summary) 2
2. 模块设计详解 4
2.1ArticleStore 设计 4
2.2 首页的构建 5
2.3ArticleList 结构 6
2.4ImageCircle 轮播图结构 6
2.5ArticleRankList 结构如下 8
2.6Article 文章详情页结构 9

1. 总结 (Summary)

TechM 基于 React 生态系统构建。涵盖项目搭建、组件化开发、集中式状态管理到声明式路由设计。

UI 框架 (React 18 + TypeScript):。React 进行组件化开发，将复杂的 UI 拆解为独立、可复用的组件。引入 TypeScript 为整个项目提供了静态类型检查，

UI 组件库 (Ant Design 5.x):项目采用了 Ant Design。如 Layout、Card、List、Avatar 等

状态管理 (Redux Toolkit): 对于跨组件共享的应用级状态（如文章详情、评论列表），使用 Redux Toolkit。通过 createSlice 和 createAsyncThunk 等工具，简化了 Redux 的样板代码。例如 articleStore，它将 reducers 和 extraReducers 组织在一起。

路由: 应用内的页面导航由 React Router 驱动。我们利用其提供的 createBrowserRouter API 集中配置所有路由规则，通过在 Layout 组件中使用 <Outlet />，实现了嵌套路由，保留公共页头和页脚，动态渲染匹配当前 URL 的页面内容。

后端与数据交互 (Axios + json-server):用 json-server 模拟后端。前端通过 Axios，所有 API 请求都被封装在 src/utils/request.ts 中，便于统一管理和维护。

主要功能模块剖析

通用布局模块 (src/layout/index.tsx): Layout 模块 定义了所有页面的共享结构。包含了网站的 Logo、主导航栏、全局搜索框和用户状态（登录/注册），通过 <Outlet/> 展示独立页面的内容

首页模块 (src/pages/HomePage/index.tsx):组织了多个功能性子组件，如横向图片轮播（ImageCircle）、文章列表（ArticleList）和排行榜（ArticleRankList）。通过组件内部的 useState 和 useEffect 管理局部 UI 状态

文章详情模块 (src/pages/Article/index.tsx):使用 useParams 从 URL 中动态获取文章 ID，并 dispatch getArticle 这个异步 thunk 来加载文章和评论数据。用户可以进行点赞、收藏等操作，这些交互会 dispatch 相应的 Redux action 来同步更新 UI。评论发布功能（handlePublishComment）异步流程：用户提交表单

-> dispatch addComment thunk -> thunk 调用 API -> 成功后 extraReducers 更新评论列表状态 -> UI 自动刷新。

状态管理模块 (src/store/): store 提供了全局状态的单一数据源。我们按照功能领域划分 slice（如 articleStore），每个 slice 内部封装了其 initialState、同步 reducers 和与异步 thunks 关联的 extraReducers。

相关结构图如下：

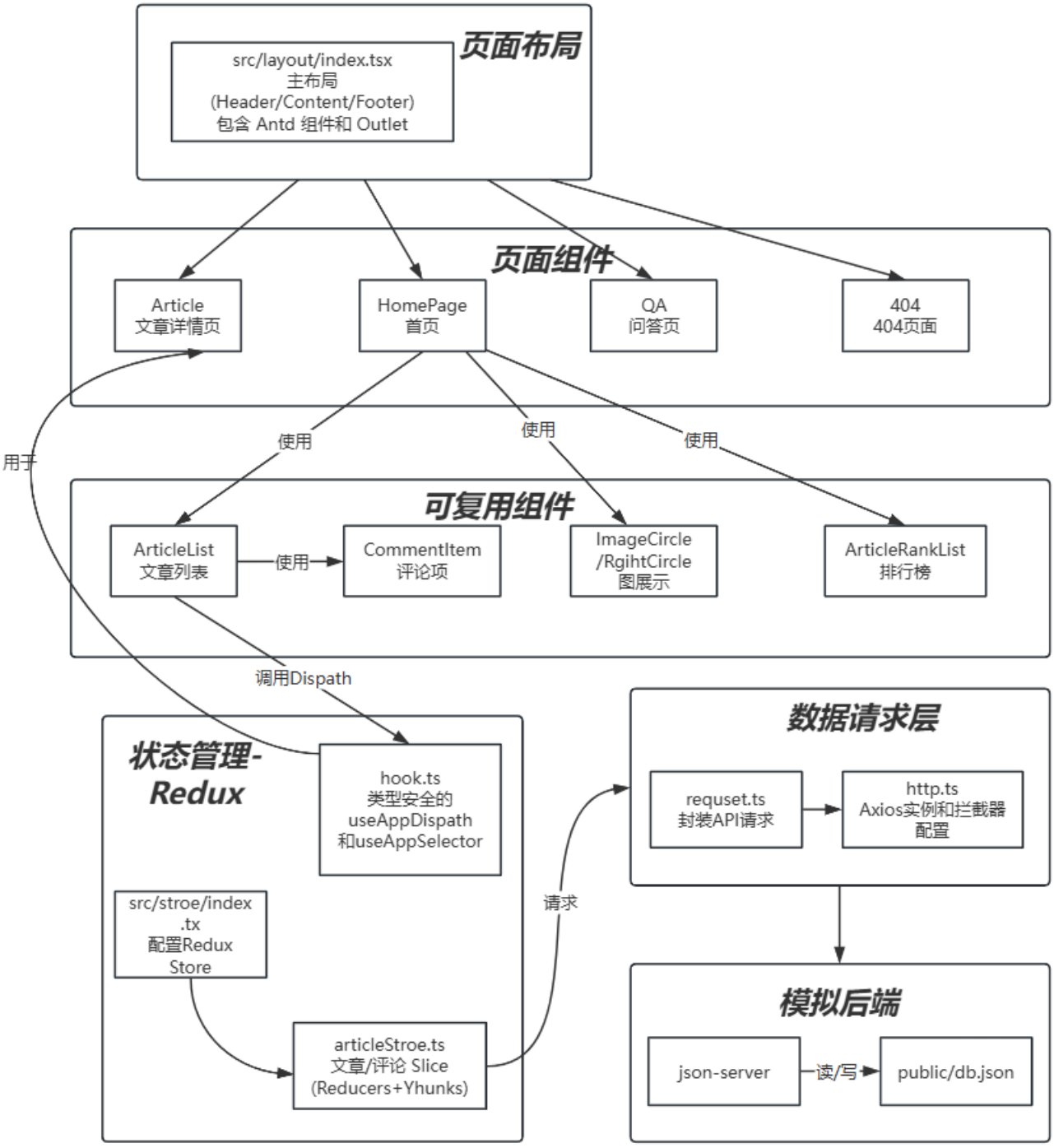

图 1.1 结构图

2. 模块设计详解

2.1ArticleStore 设计

使用的是 Redux 进行状态管理，大概的结构图如下

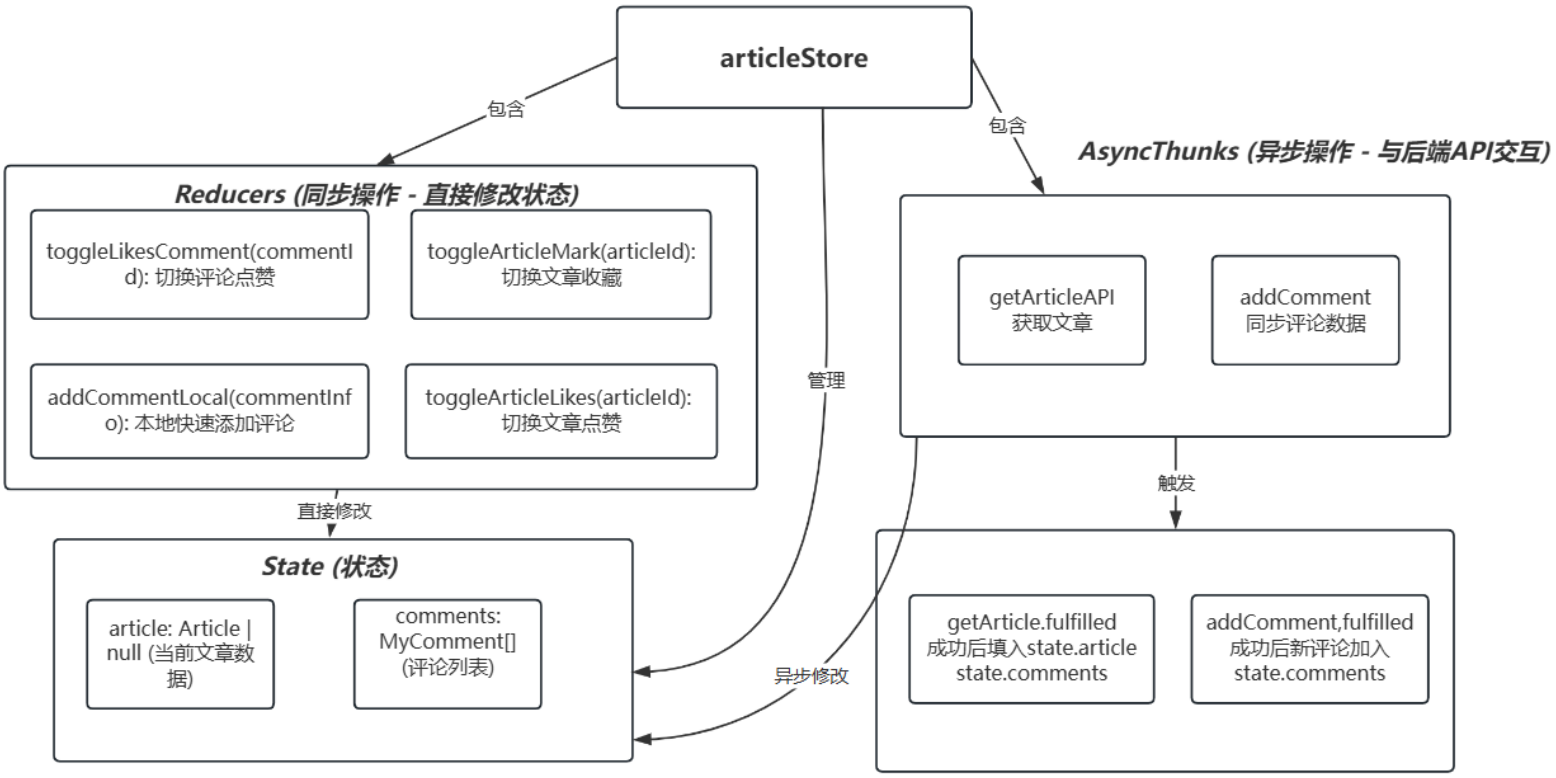

图 2.1 ArticleStore 结构图

简单来说 Stroe 就是

1. State 管理数据

article: 当前正在查看的文章的详细信息（如标题、内容、作者等）。

comments: 这篇文章对应的所有评论列表。

2. Reducers (同步操作)

这些是能够立即、同步完成的状态修改任务。

toggleLikesComment:/toggleArticleMark/ toggleArticleLikes

3. AsyncThunks (异步操作)

服务器进行通信的异步任务:getArticle:/addComment:

4. ExtraReducers -

这个部分负责监听异步任务（AsyncThunks） 的执行结果，来更新 State。

反思和问题：

当时是想把评论也设计成一个 Store，但是文章和评论联系有的紧密，文章里面有评论，文章自己可以管理自己的评论，虽然增加了耦合性但是逻辑上是通顺的。

其实最大的问题是 数据的异步更新 ，就是更新的时候导致了 State 的地址发生了变化，会强制更新，当时在做评论的添加的时候，先更新本地后更新云端，本地没有问题，云端更新就会刷新页面，很奇怪。

2.2 首页的构建

左侧由 ArticleList.ts 和 ImageCircle 右侧由 ArticleRankList.tsx, RightImageList.tsx 四部分构成，构成图如下

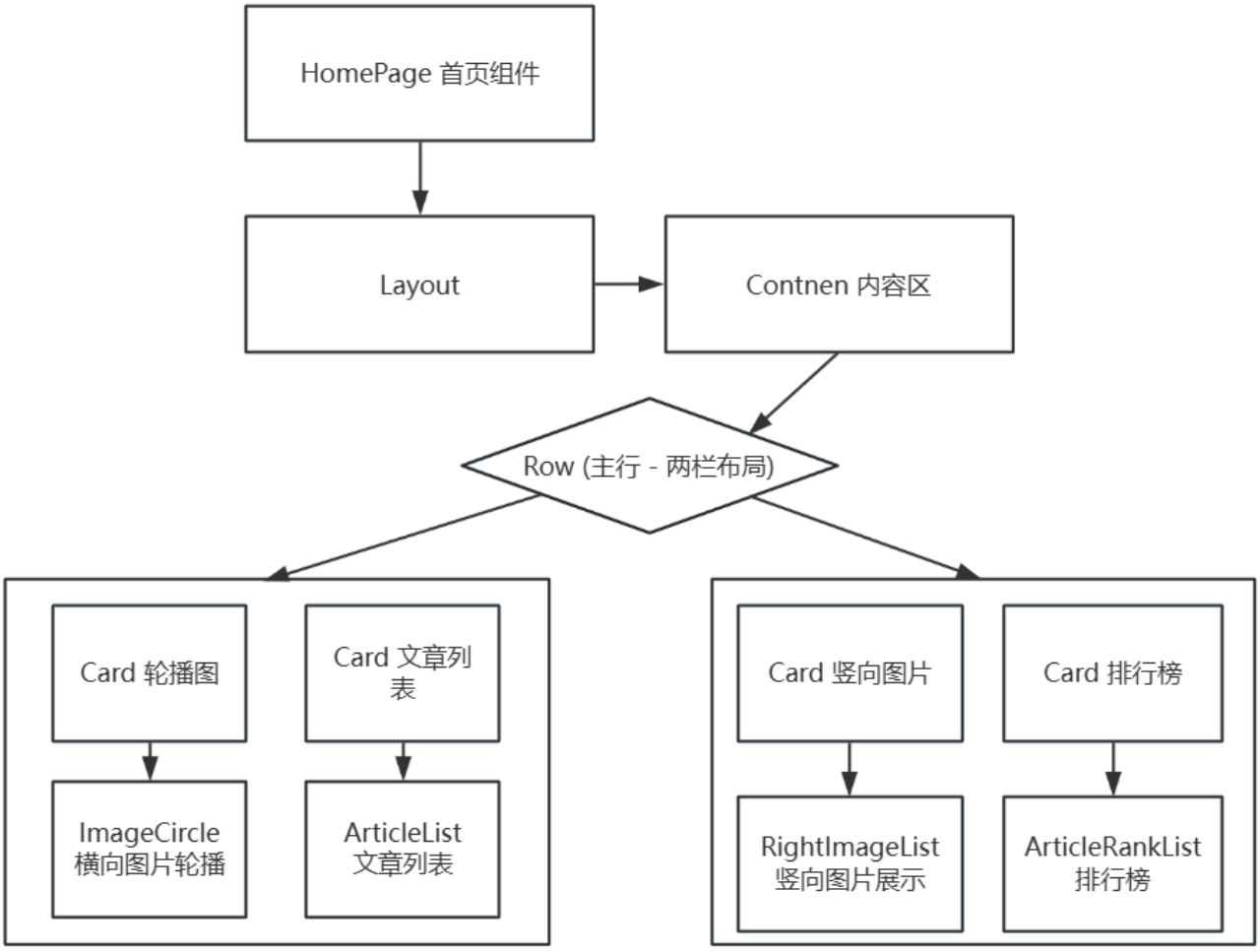

图 2.2 HomePage 结构图

2.3ArticleList 结构

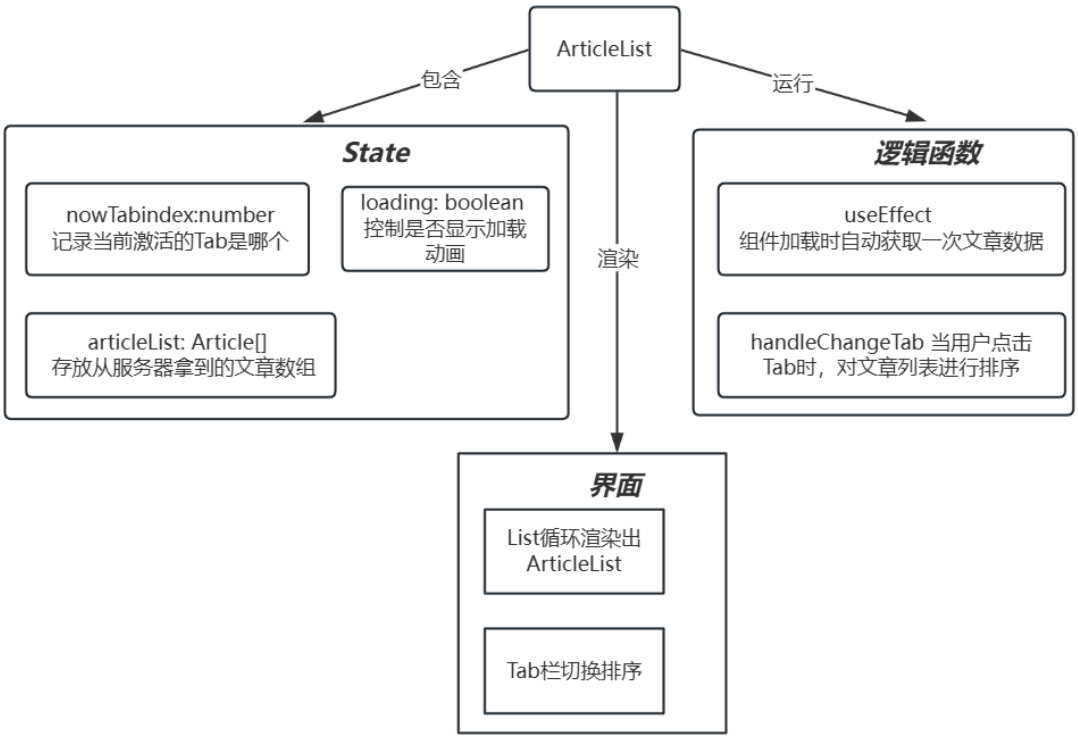

图 2.4ArticleList 结构图

讲一下 tab 切换和排序的逻辑，其实就是判断 当前的 key 等不等于点击的，关键代码是这个```${key === nowTabindex ? 'active' : ''}```通过三元运算符来判断是否添加类‘active’

排序的逻辑，其实就是通过 sort 来添加排序规则再通过展开运算符‘…’来创建新数组，由于 State 的更新是看地址，所以浅拷贝可行,关键代码如下```return [...list].sort((a, b) => {}```来返回一个新数组。

2.4ImageCircle 轮播图结构

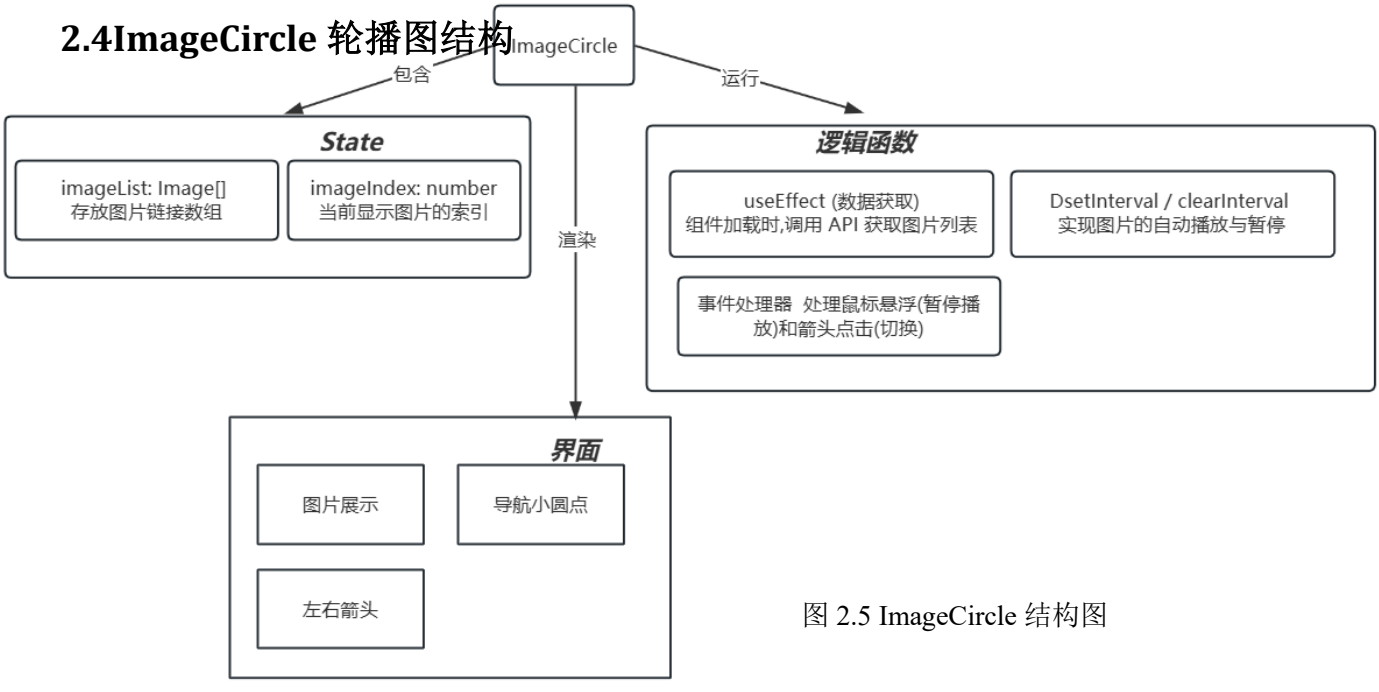

图 2.5 ImageCircle 结构图

小圆点的状态转化和 tab 类似，这里重点解释轮播图的定时器循环播放

最开始我以为是防抖，因为 React 在组件发生变化的时候会更新，会频繁触发定时器，会有很多的定时器，我发现不太行，还是会创建很多定时器，之后我想用全局变量来存储，但是全局变量只会有一个实例但是我的组件是可以复用的，会导致多个实例共享一个变量，会混乱，最后查询资料发现使用 useRef 来实现，定时器 id 的存储，不触发更新，持久性符合要求。关键代码如下

```
    const intervalIdRef = useRef<NodeJS.Timeout | null>(null);
      const beginImage = () => {
        if (intervalIdRef.current)
          clearInterval(intervalIdRef.current)
        intervalIdRef.current = setInterval(() => {
            setImageIndex(prevIndex => (prevIndex + 1) %
imageList.length);
        }, 1000);
      }
      const stopImage = () => {
        if (intervalIdRef.current)
          clearInterval(intervalIdRef.current);
      }
```

RightImageList 右侧图片列表

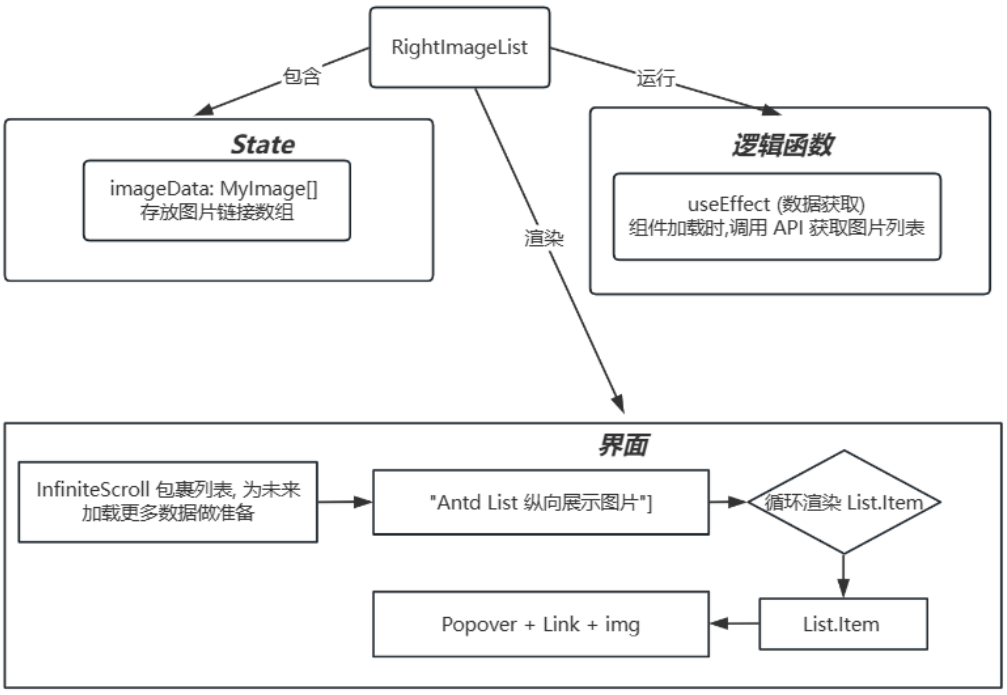

图 2.6 RightImageList 结构图

这个部分单纯的图片的展示，可能比较关键的是使用 InfiniteScroll 包裹列表，来实现滚动效果，Link 来实现点击，Popover 来实现悬浮，这个页面没有出现大问题。

2.5ArticleRankList 结构如下

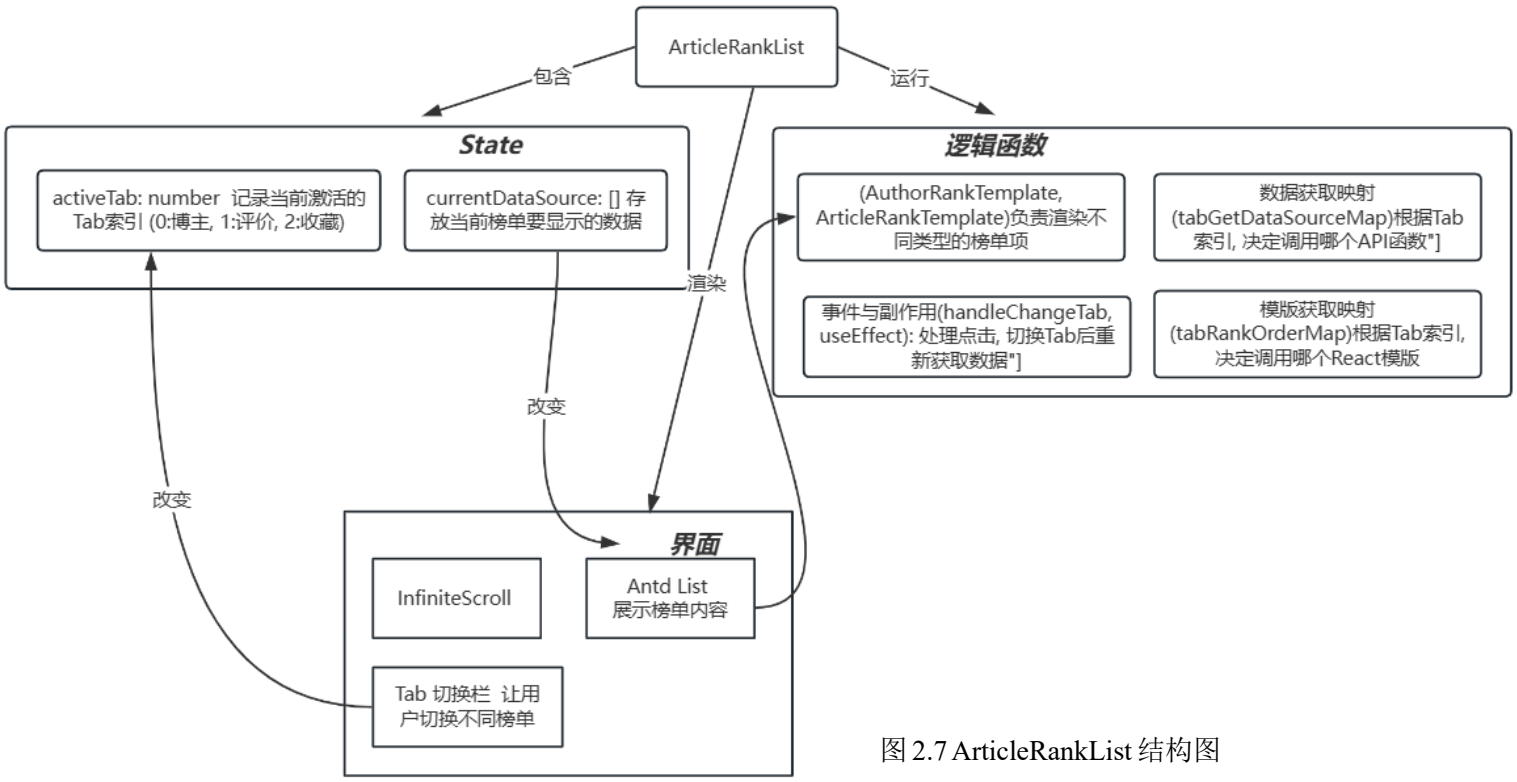

图 2.7 ArticleRankList 结构图

大致就是由两个组件一个 tab 栏控制组成，一个是博主的组件 const AuthorRankTemplate: React.FC，另一个是文章的组件 const ArticleRankTemplate，通过不同 tab 的 key 值然后改变 dataSource 来改变传入模版的 Props 来改变渲染内容。遇到的问题的，当 activeTab 发生改变的时候，渲染模版也发生了变化，导致数据源和模版不匹配，出现了 undefined，所以当时试了挺久的，用 useEffect，用同步，或是 sleep，都不行，最后是设置为空就可以了，相当于‘空一拍’，让数据跟上。关键代码如下：

```
    const handleChangeTab = (index: number) => {
        setActiveTab(index);
        setCurrentDataSource([]);//防止数据跟不上导致 undefined
中断程序运行导致数据获取不到}
```

还有个重要的就是使用 Map 来代替多重的 if else if 判断，一次成功的尝试，通过 activeTab→dateSource→Template ，通过 activeTab 来改变两者，关键代码如下：

```
const tabGetDataSourceMap = new Map<number, () => Promise<(Article
| AuthorRank)[]>>([
    [0, getAuthorRankList],
    [1, getArticleRankByCommentList],
    [2, getArticleRankByMarkList]
  ]);
  }
}
}
const tabRankOrderMap = new Map<number, React.FC<renderProp>>(
[[0,AuthorRankTemplate],
[1,ArticleRankTemplate],
[2,ArticleRankTemplate]]);
  useEffect(() => {
    const getDataFun = async () => {
      const data = await tabGetDataSourceMap.get(activeTab)!();
      setCurrentDataSource(data);
    }
    getDataFun();
  }, [activeTab])

 renderItem={(item, index) => {
             return (tabRankOrderMap.get(activeTab)!({ item: item,
index: index, orderKey: '' }))
}}
```

2.6Article 文章详情页结构

简单来说就两个部分，文章主体部分，有作者信息，文章内容，点赞收藏栏，还有评论区部分，发布评论，评论点赞。其中关键的是，更新评论，先进行本地更新，然后进行云端更新，云端更新完之后就会刷新。关键代码如下：

```
  await dispatch(addComment({author: author,
content: currentComment, articleId: article.id,
         })).unwrap();
         setCurrentComment('');
        message.success('评论发布成功! ');
```

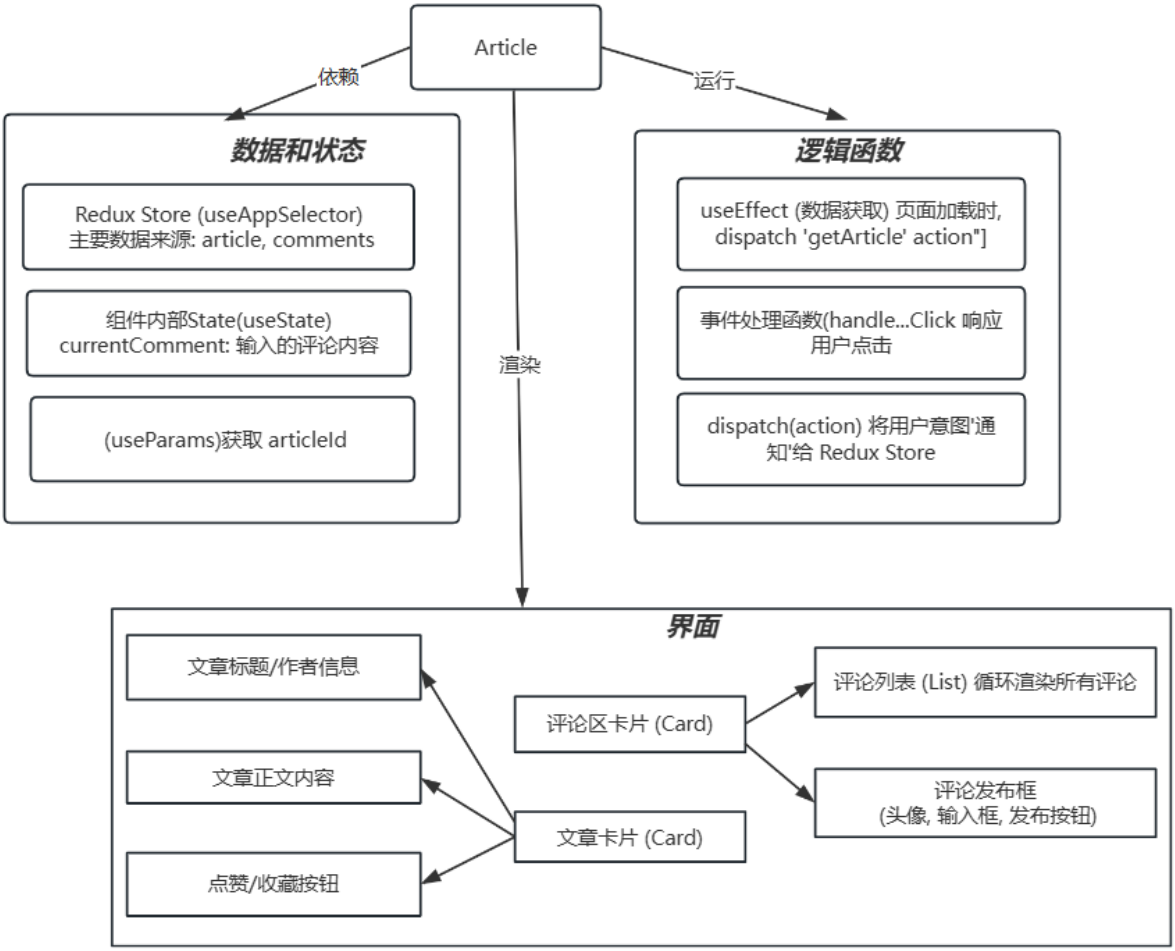

图 2.9Article 结构图