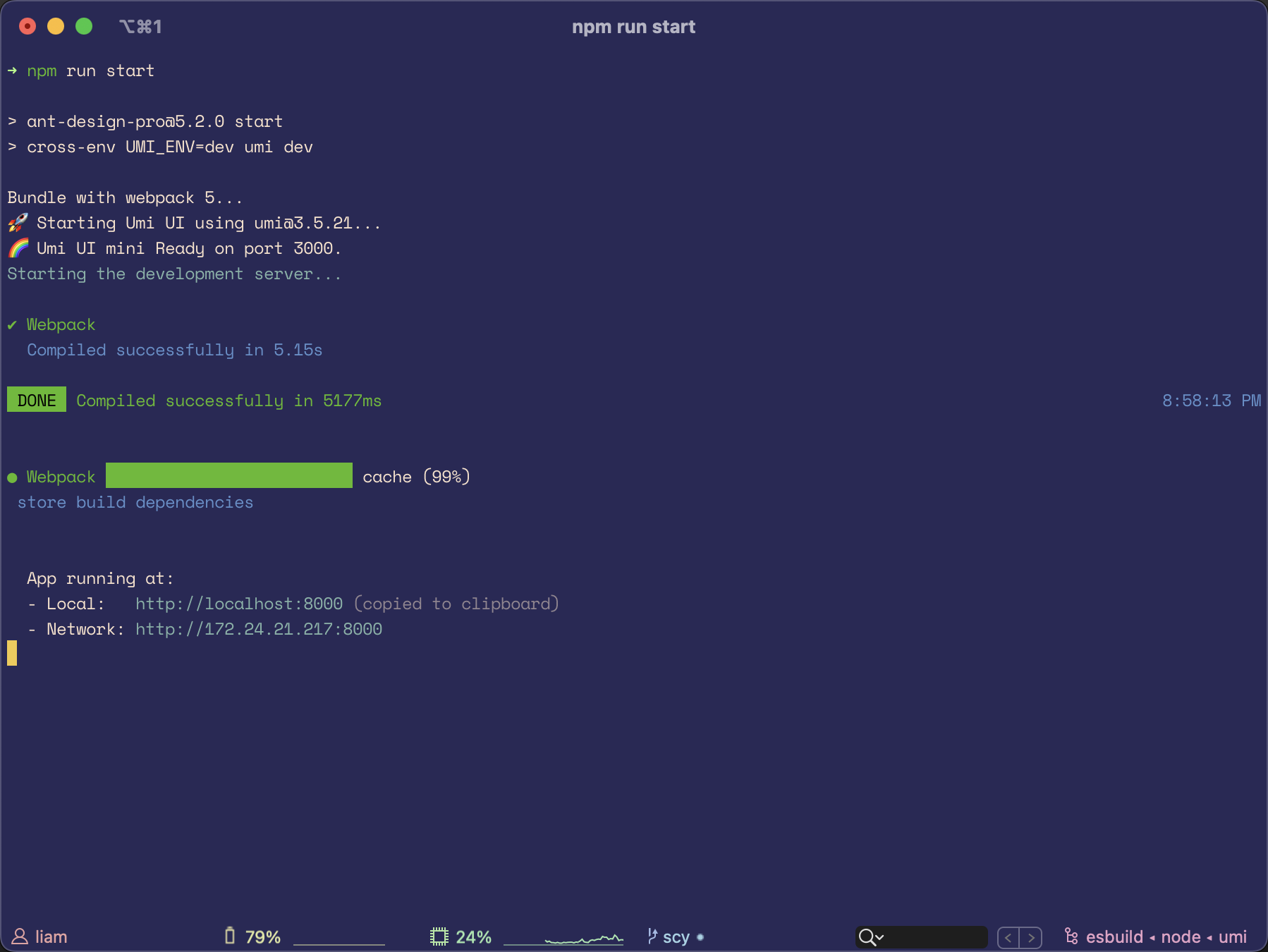
**[04/11]进度汇报**

**1 项目结构介绍**

项目基于React+Ant Design，使用全量Typescript开发，并基于Webpack和UmiJS构建。

|  |
| --- |
| Plain Text  ├── config # umi 配置，包含路由，构建等配置  ├── mock # 本地模拟数据  ├── public  │ └── favicon.png # Favicon  ├── src  │ ├── assets # 本地静态资源  │ ├── components # 业务通用组件  │ ├── e2e # 集成测试用例  │ ├── layouts # 通用布局  │ ├── models # 全局 dva model  │ ├── pages # 业务页面入口和常用模板  │ ├── services # 后台接口服务  │ ├── utils # 工具库  │ ├── locales # 国际化资源  │ ├── global.less # 全局样式  │ └── global.ts # 全局 JS  ├── tests # 测试工具  ├── README.md  └── package.json |

使用npm run start会启动服务：



在pages文件夹下，项目页面的主要结构如下：

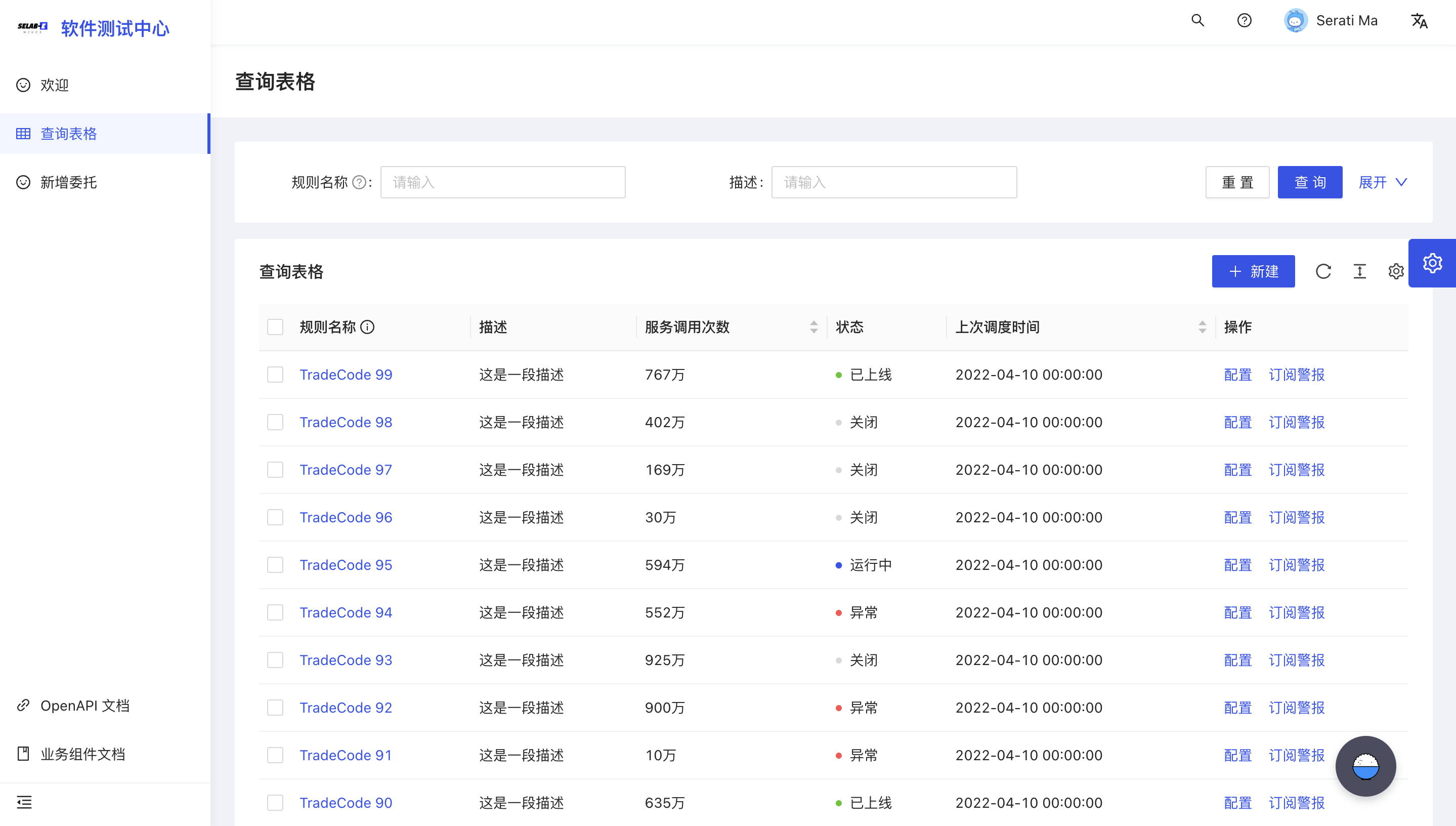
|  |
| --- |
| Plain Text  src  ├── components  └── pages  ├── Welcome // 路由组件  | ├── components  | ├── Form.tsx  | ├── index.tsx // 页面组件的代码  | └── index.less // 页面样式  ├── Order  | └── index.less  ├── user // 一系列页面推荐通过小写的单一字母做 group 目录  | ├── components // group 下公用的组件集合  | ├── Login // group 下的页面 Login  | ├── Register // group 下的页面 Register  | └── util.ts // 共用方法  └── \* // 其它页面组件代码 |

**2 研发进度汇总**

组内研发人员分别基于前期需求调研完成了一个小型demo，完成对框架的上手开发：

**2.1**

下面是基于mock.js生成数据并使用框架自带的dva工具进行数据fetch，最后使用组件进行数据渲染的例子：



mock.js可以方便地模拟数据，用于页面渲染的测试：

|  |
| --- |
| TypeScript  export default {  'GET /api/rule': getRule,  'POST /api/rule': postRule,  }; |

**2.2**

下面是基于ProForm组件实现的一个表单demo，包含了自定义的验证规则、填写项联动等，实现了《软件项目委托测试申请书》的内容：



React会实时进行组件render，在控制台输出填写情况如下：



**2.3**

实现框架的request请求拦截器，用于管理浏览器cookie和为所有请求生成包含状态信息的请求头。

**3 产品进度汇总**

@张嘉麒