## **hdu-ai-web**

HDU AI平台（仿腾讯AI）

### **项目介绍**

基于vue.js的人工智能算法平台，样式部分大多直接复制腾讯AI开放平台的html和css代码并作一些修改，一部分采用iview的ui组件，功能部分采用vue .js参照腾讯AI开放平台的功能进行复现。该平台主要包含首页、技术引擎、文档中心、登录、注册、控制台首页、应用管理（创建应用、删除应用、应用概览、数据分析、应用信息）、能力库、接入能力等功能。

### **项目预览**

预览地址： http://106.54.192.188/hdu-ai-web/

测试账号：test 密码：123

### **启动项目**

注：需要先在开发机器上安装node.js(npm)

安装npm依赖

npm install

在开发环境启动项目, 以及打包项目

1. 使用命令行

# 启动项目

npm run serve

# 打包项目

npm run build

2. 使用vue ui（推荐）

# 安装最新版的vue-cli

npm install -g @vue/cli

# 检查vue-cli版本

vue -V

# 运行 vue ui

vue ui

# 运行vue ui之后，在ui界面进行相应的启动打包操作

### **部署项目**

本系统采用前后端分离的模式，可以考虑将前端项目单独部署到nginx并通过添加反向代理配置解决跨域问题。

接下来介绍如何将本项目部署到云服务器的docker nginx容器上

nginx容器的创建及相关配置文件可参考项目代码中的docker文件夹  
创建好nginx容器后，用xftp等工具连接云服务器，将打包生成的dist目录下的文件拖入对应目录

### **技术选型**

* MVVM框架：vue.js
* 前端路由框架：vue-router
* 全局状态管理框架：vuex
* 前端UI框架：iview（view ui）
* 前端HTTP框架：Axios
* 图表框架：Echarts

### **项目结构**

* api -- 请求后端数据用到的接口
* assets -- 静态资源，如图片等
* components -- 公共组件
  + footer -- 控制台的页脚（前台的页面暂未抽离也未修改）
  + header -- 前台顶栏和控制台顶栏
  + chart.vue -- 简单封装的echarts折线图
* libs -- 工具文件
  + axios.js -- 封装axios，用于api的http请求
  + dateUtil -- 日期格式转化
  + storage -- 封装localstorage，主要用于存储token
* mock -- 前期开发时写的模拟数据，后期有变动，以后端返回数据为准
* iview -- vue-cli插件形式安装iview
* router -- 前端路由vue-router
* store -- 全局状态管理框架vuex
* views -- 项目的页面文件
  + console -- 控制台相关页面
    - application -- 应用管理相关页面
      * create-app.vue -- 创建应用
      * detail.vue -- 应用详情主体布局
      * detail
        + data-analysis.vue -- 应用详情的数据分析
        + data-info.vue -- 应用详情的应用信息
        + overview.vue -- 应用详情的应用概览
    - capability -- 能力库相关页面
      * detail -- 能力详情页面
      * overview -- 能力库列表页面
    - home -- 控制台首页
    - user -- 账号信息页面
  + error -- 错误页面（暂时只用到了404）
  + front -- 前台相关页面
    - doc -- 文档中心
      * detail.vue -- 文档中心详情页
      * home.vue -- 文档中心首页
    - home -- 前台首页
    - product -- 技术引擎详情页
  + login -- 登录页面
  + regist -- 注册页面

### **项目功能**

本项目主要分为前台和控制台。其中，前台页面不需要登录即可访问，而控制台页面需要登录后才能访问。本项目未实现权限功能，仅通过api是否需要token来实现是否需要登录才能访问。如果某个页面的接口需要token，用户未登录时调用该接口，因为没有token或token过期，后台会返回401（和后台的约定）表示未授权，此时前端会清空localstorage里的token和userInfo，并跳转登录页面让用户重新登录。

对于用户来说，大概的使用流程如下：

* 通过前台部分了解和体验平台的算法
* 注册登录以便后续使用平台提供的算法
* 登录后进入控制台
* 创建应用
* 浏览能力库，为应用接入能力
* 对照文档，使用应用的appid和appkey调用已接入的能力
* 通过控制台管理应用，增加应用、删除应用、查看应用详情（能力调用情况、能力运行情况）

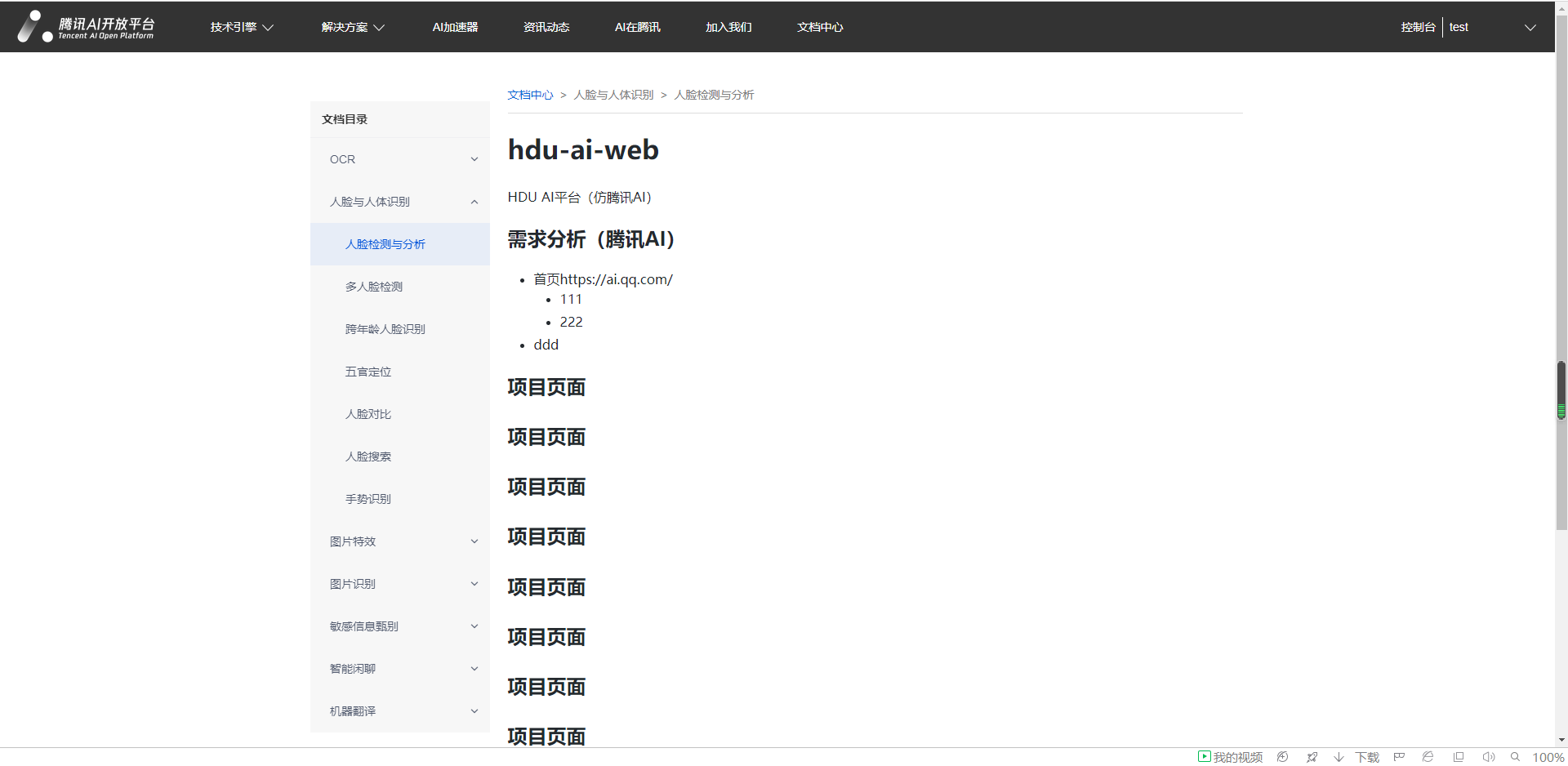
#### **前台部分**

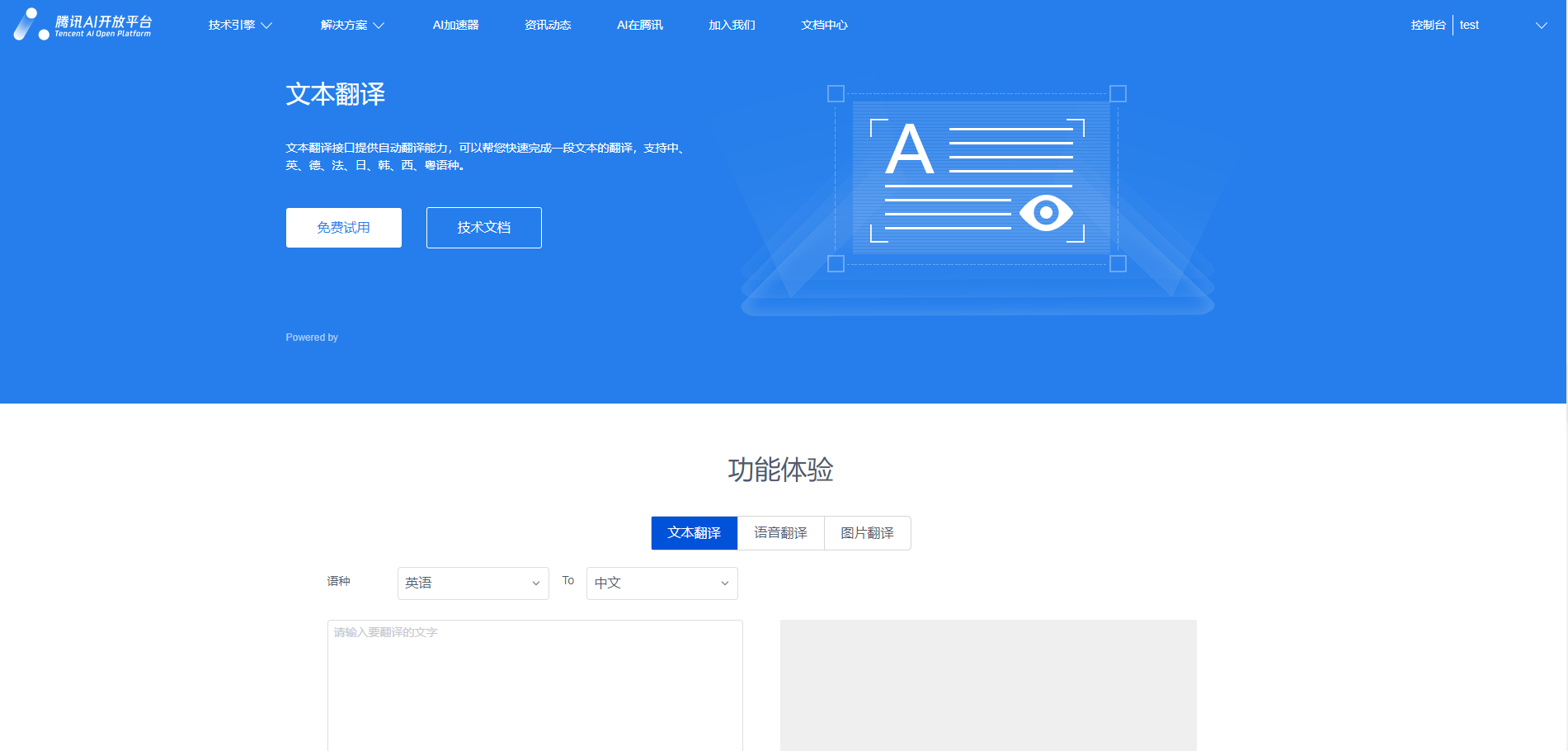
前台部分主要用于让用户了解我们平台提供的算法及相关文档。

前台部分主要包括首页、文档中心、技术引擎详情页以及前台每个页面都有的顶栏。  
首页主要有一个轮播图和技术引擎的目录。

  
文档中心包括文档目录和文档详情页，目前文档目录和技术引擎目录一致，文档详情页的具体内容还未填充。



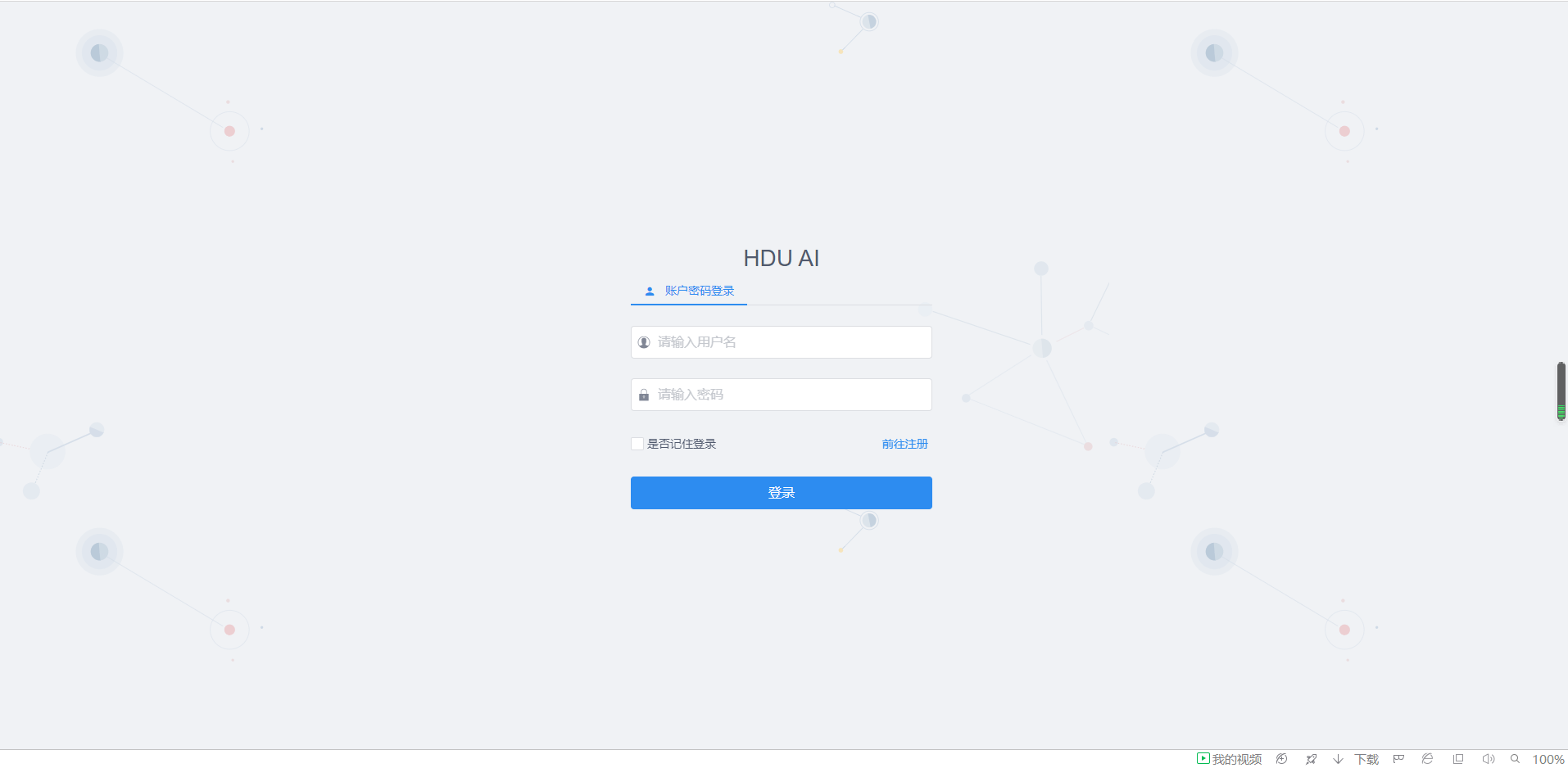
  
技术引擎详情页主要有各个算法的具体介绍和功能体验，其中功能体验目前只实现了机器翻译的文字翻译，后续只需要和后台做好约定，根据后台返回的目录中的关键字进行条件渲染对应的页面即可。

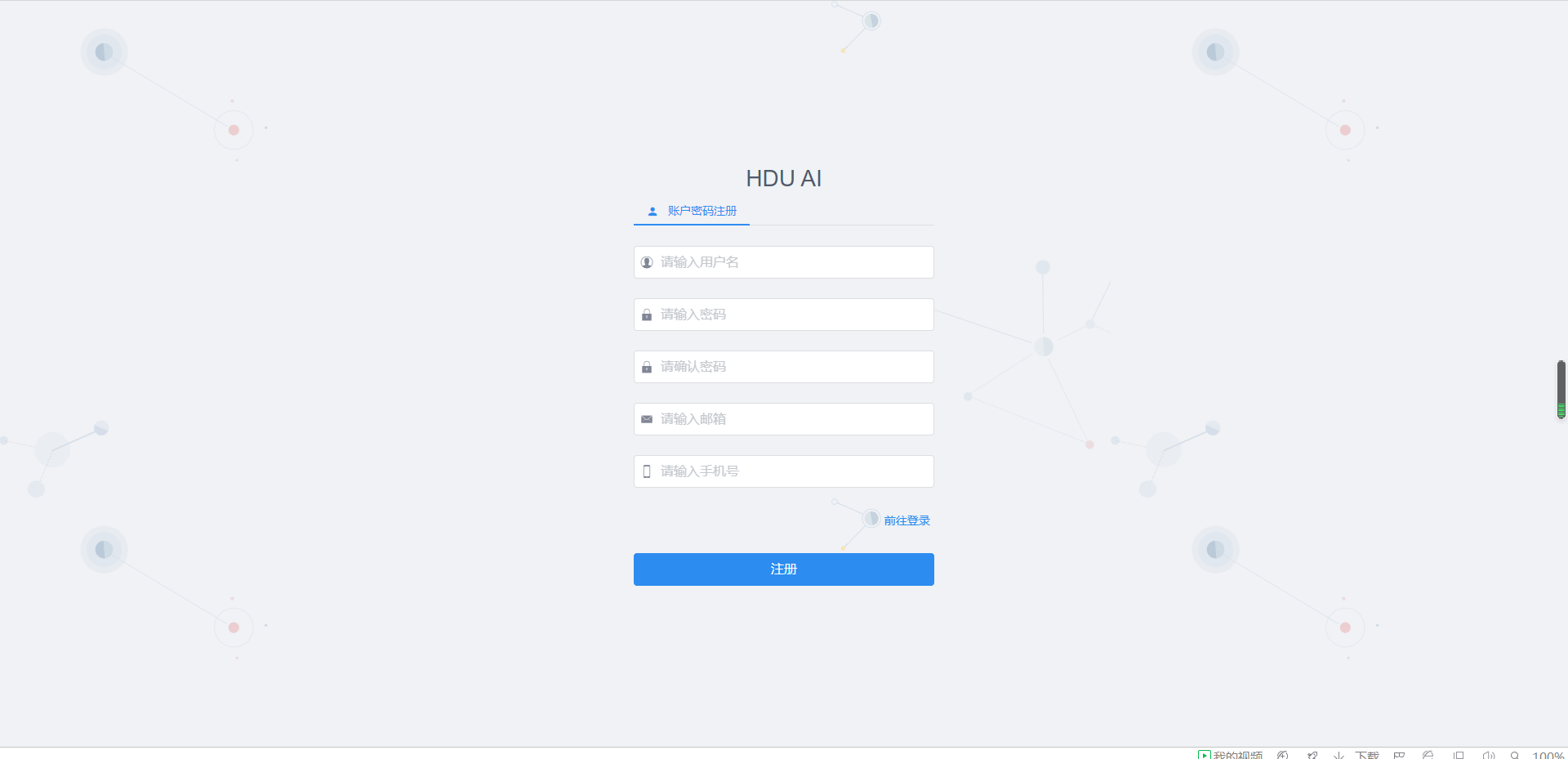
  
顶栏包含前台部分各个页面的入口，以及登录入口和控制台入口。



#### **登录注册**

本项目的账号验证方式采用最常见的账号密码形式。 其中，前端会对用户输入的密码进行一次md5加密后再传给后端。



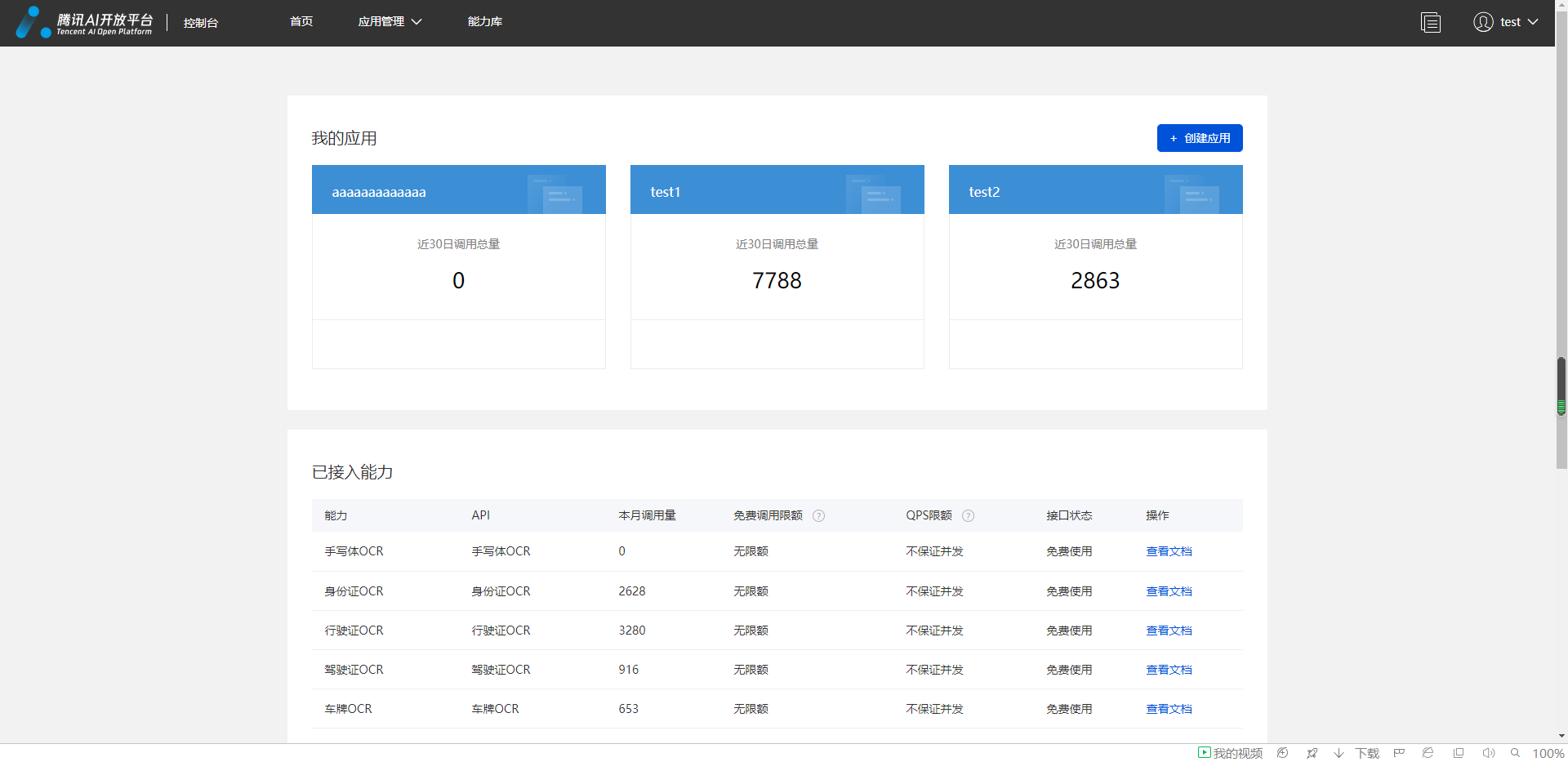


#### **控制台部分**

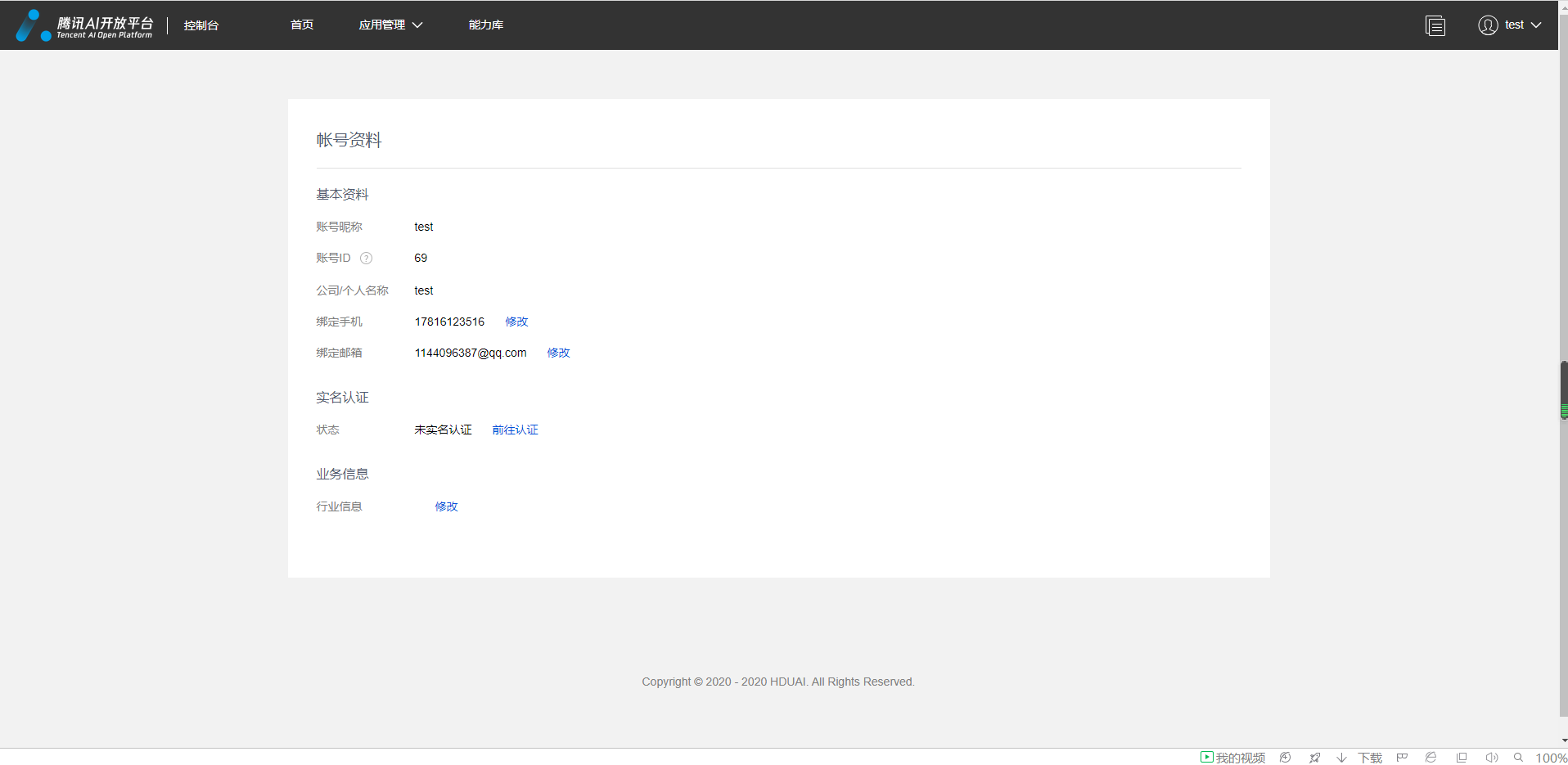
控制台部分用于让用户管理自己的应用和接入能力。

控制台部分主要包括控制台首页、账号信息、应用管理、能力库以及控制台每个页面都有的控制台顶栏。

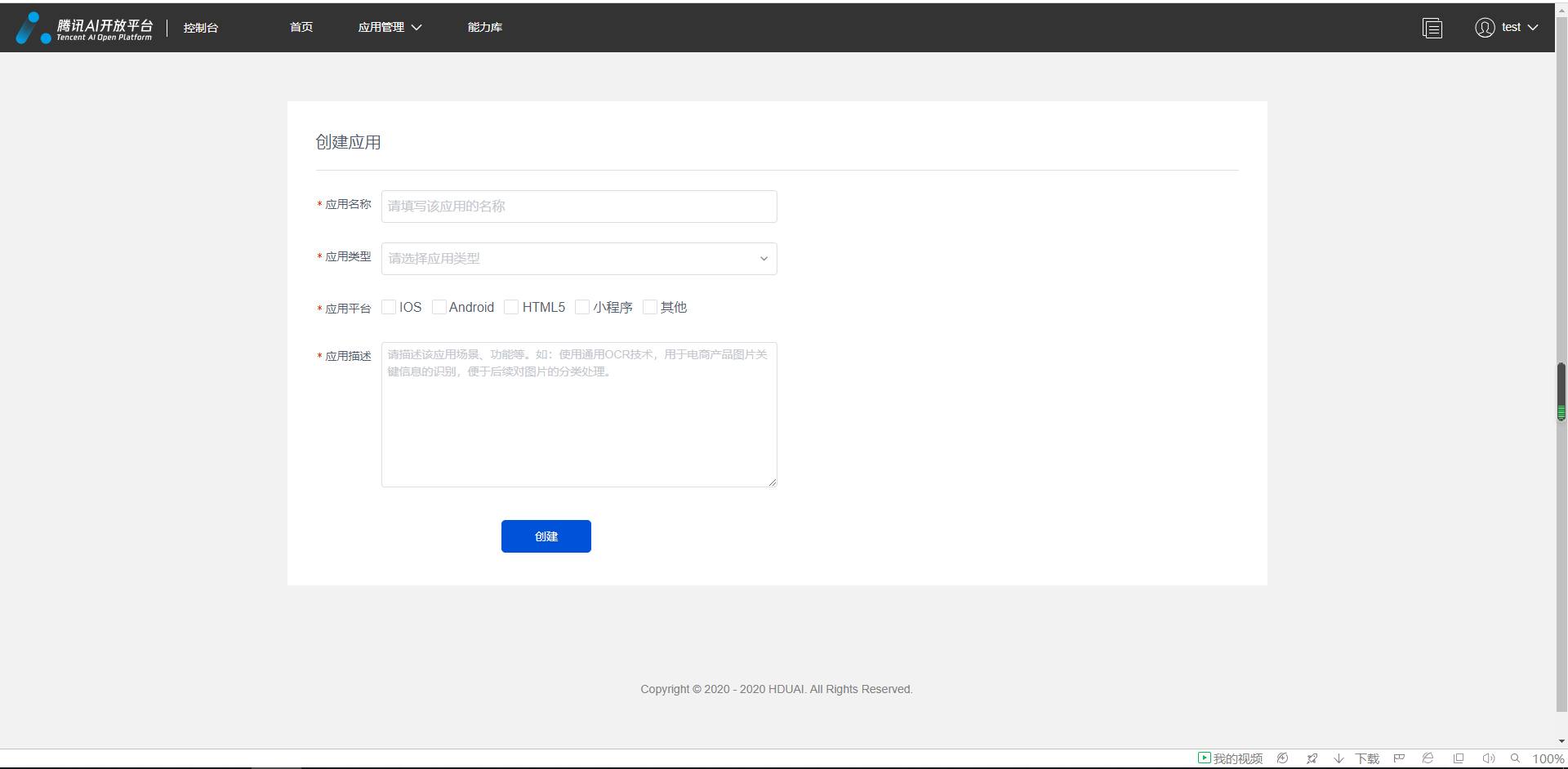
在控制台首页，用户能查看拥有的应用和总体的能力接入情况。

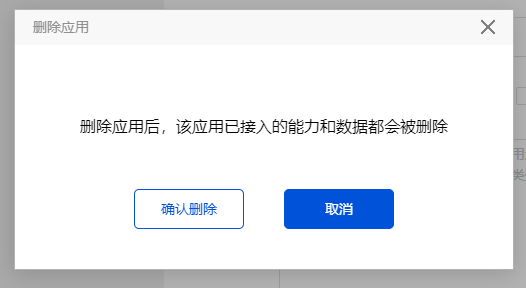


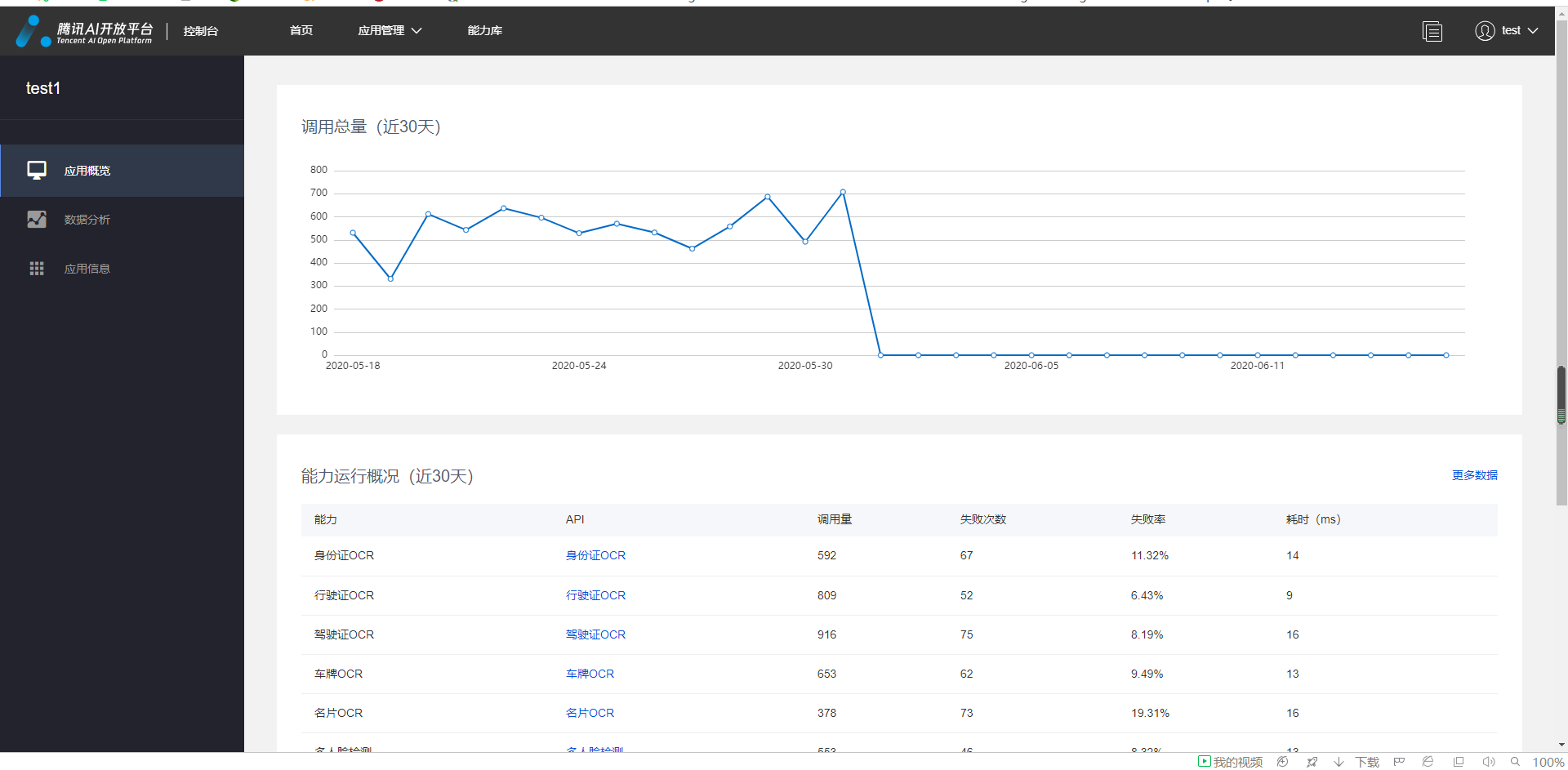
账号信息主要包括用户的id、绑定的手机和绑定的邮箱。

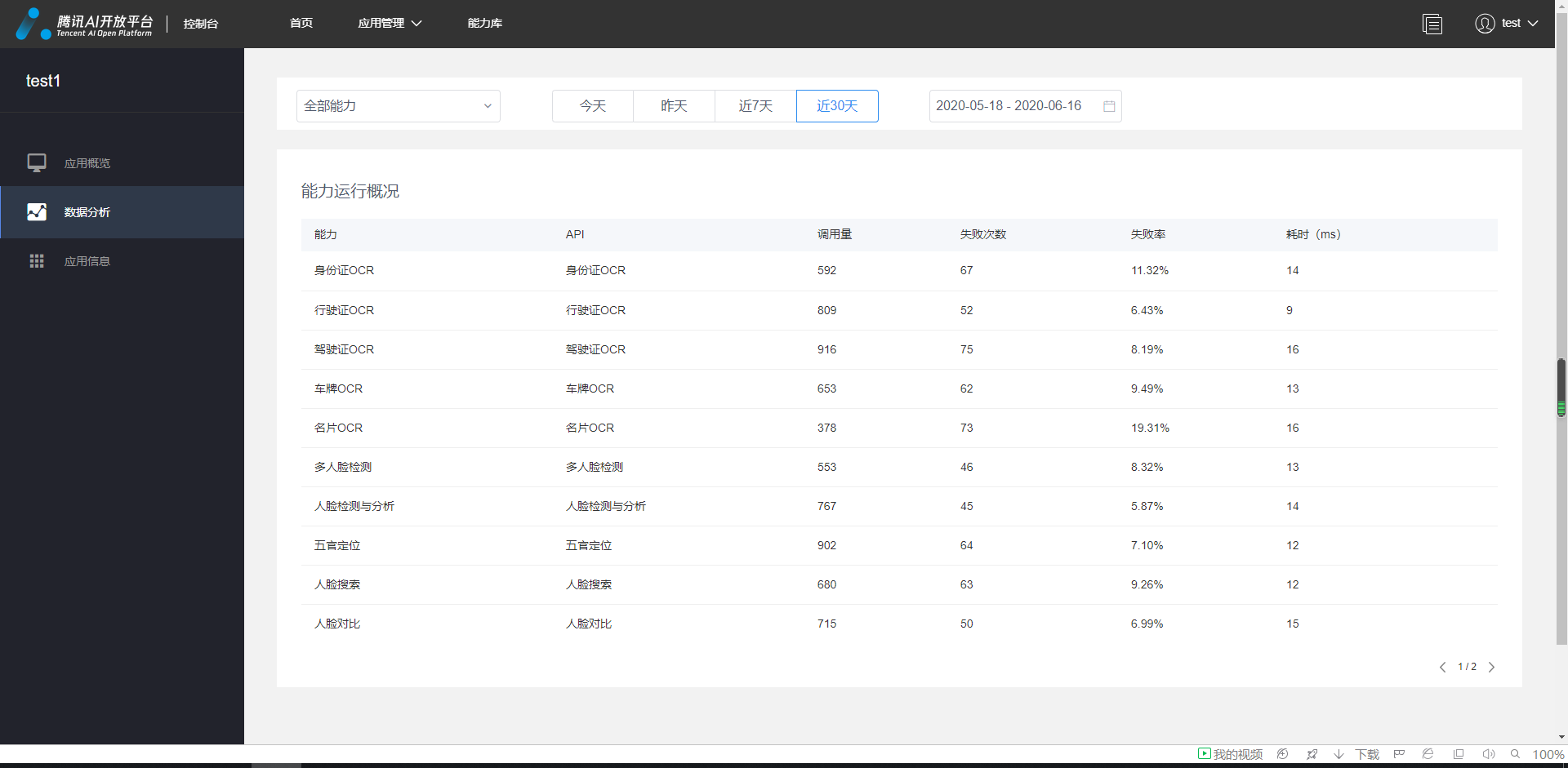


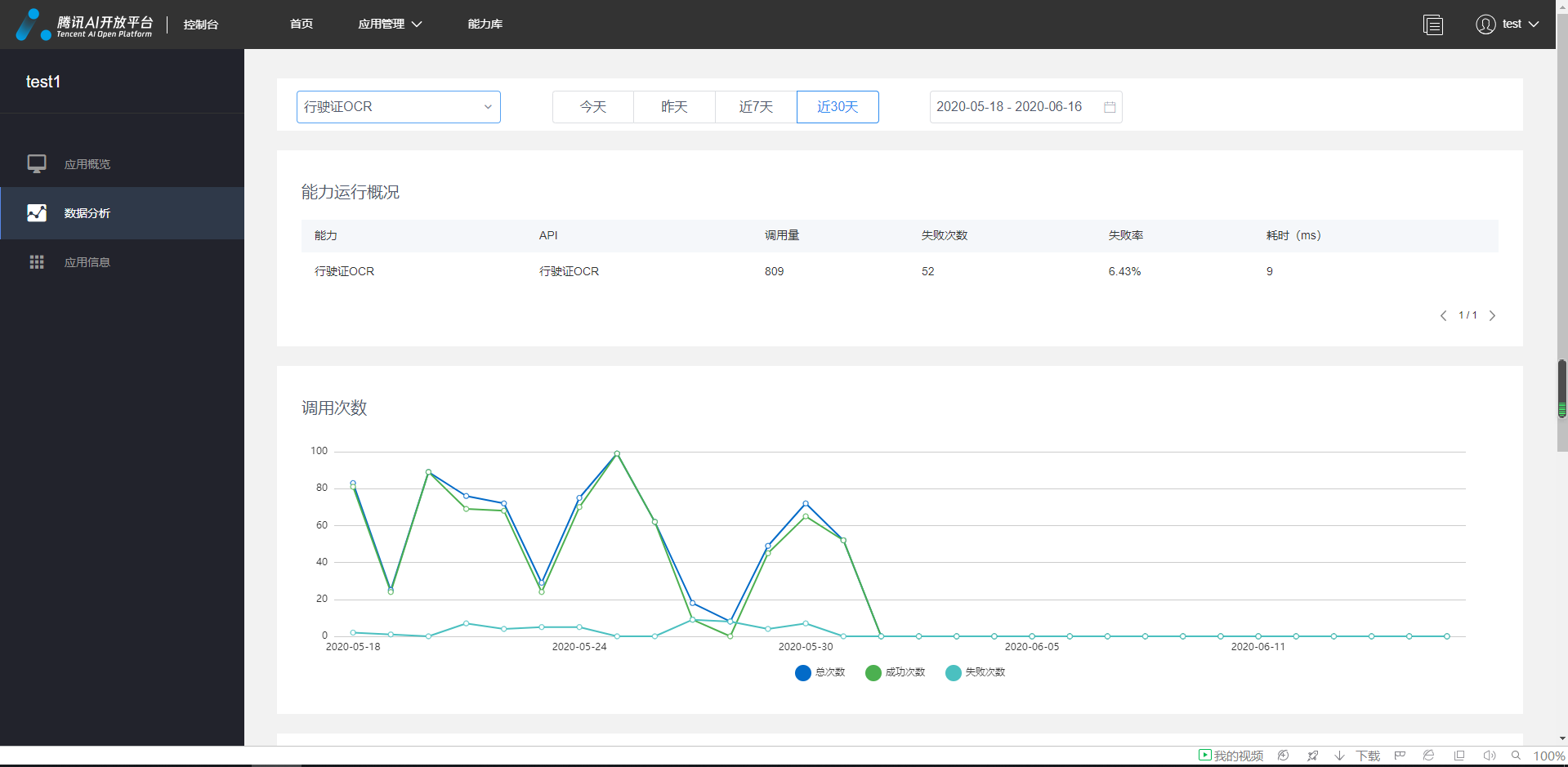
应用管理，主要有创建应用、删除应用和查看应用详情功能。在应用概览中，用户能查看该应用近30天的调用总量和能力运行概况。在数据分析中，用户可以通过选择能力和时间来查看相应的应用调用情况和能力运行情况。在应用信息中，用户能够查看应用的信息以及修改应用的部分信息，其中appid和appkey是用户调用能力的关键。

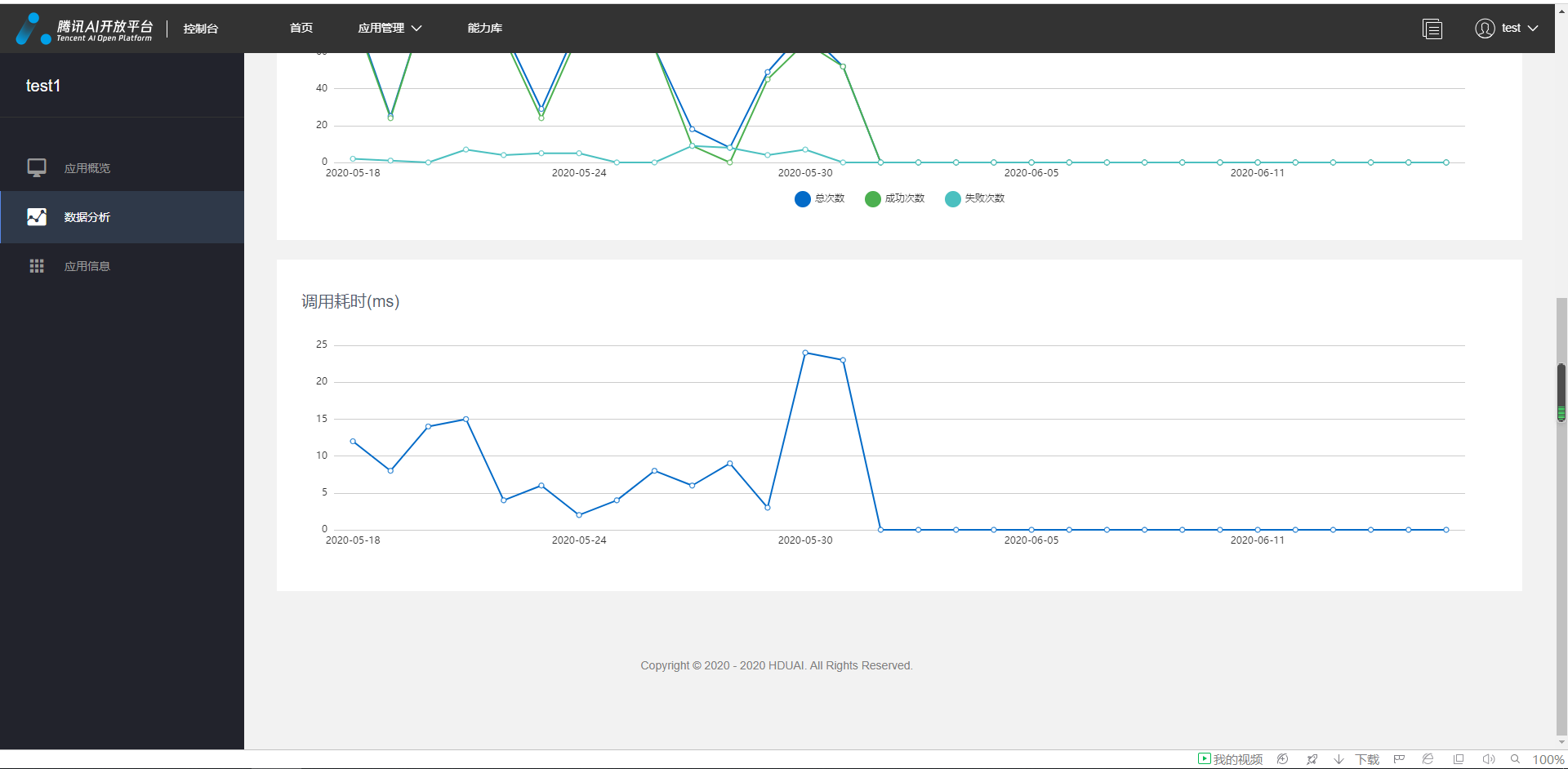


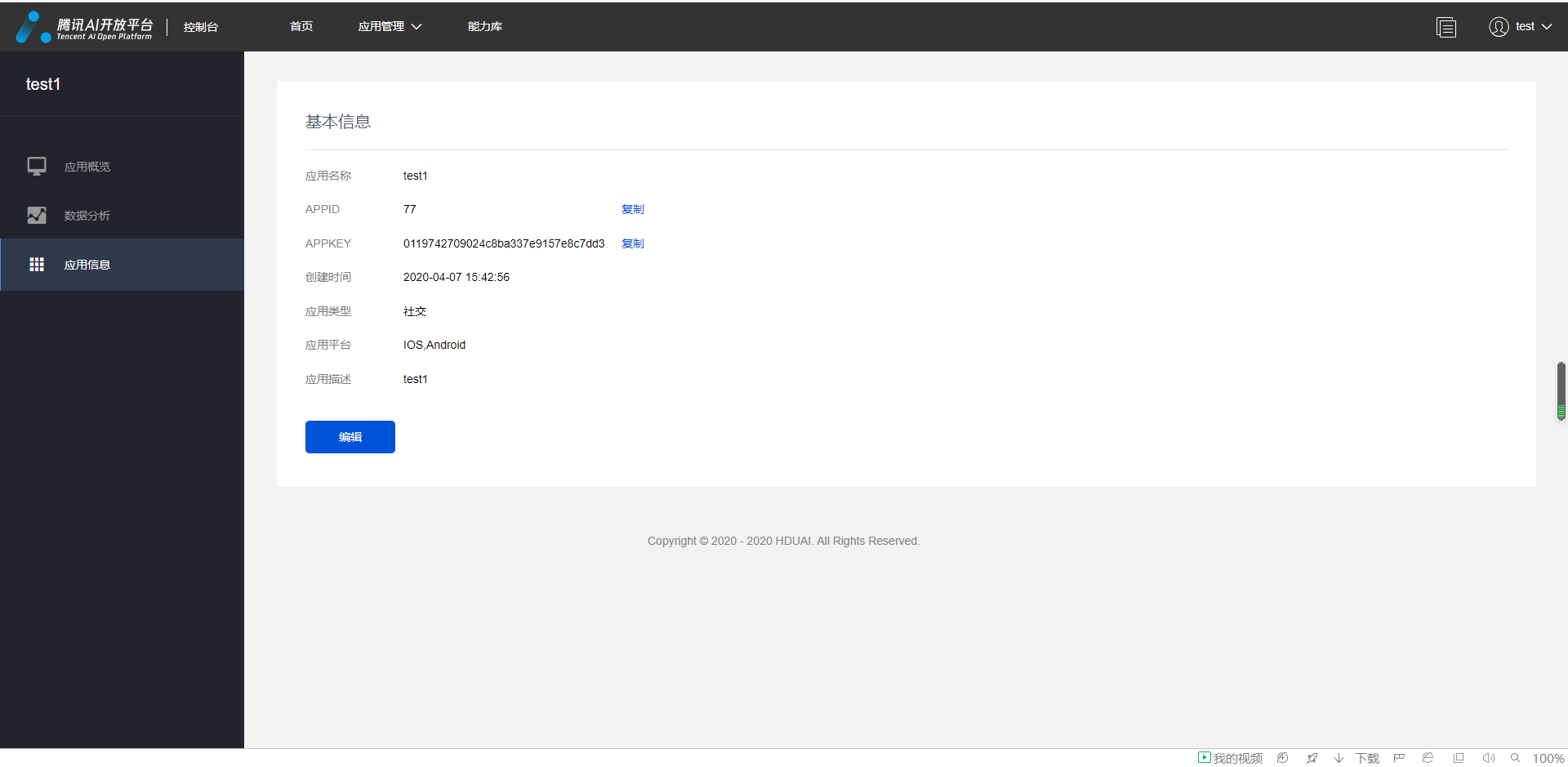


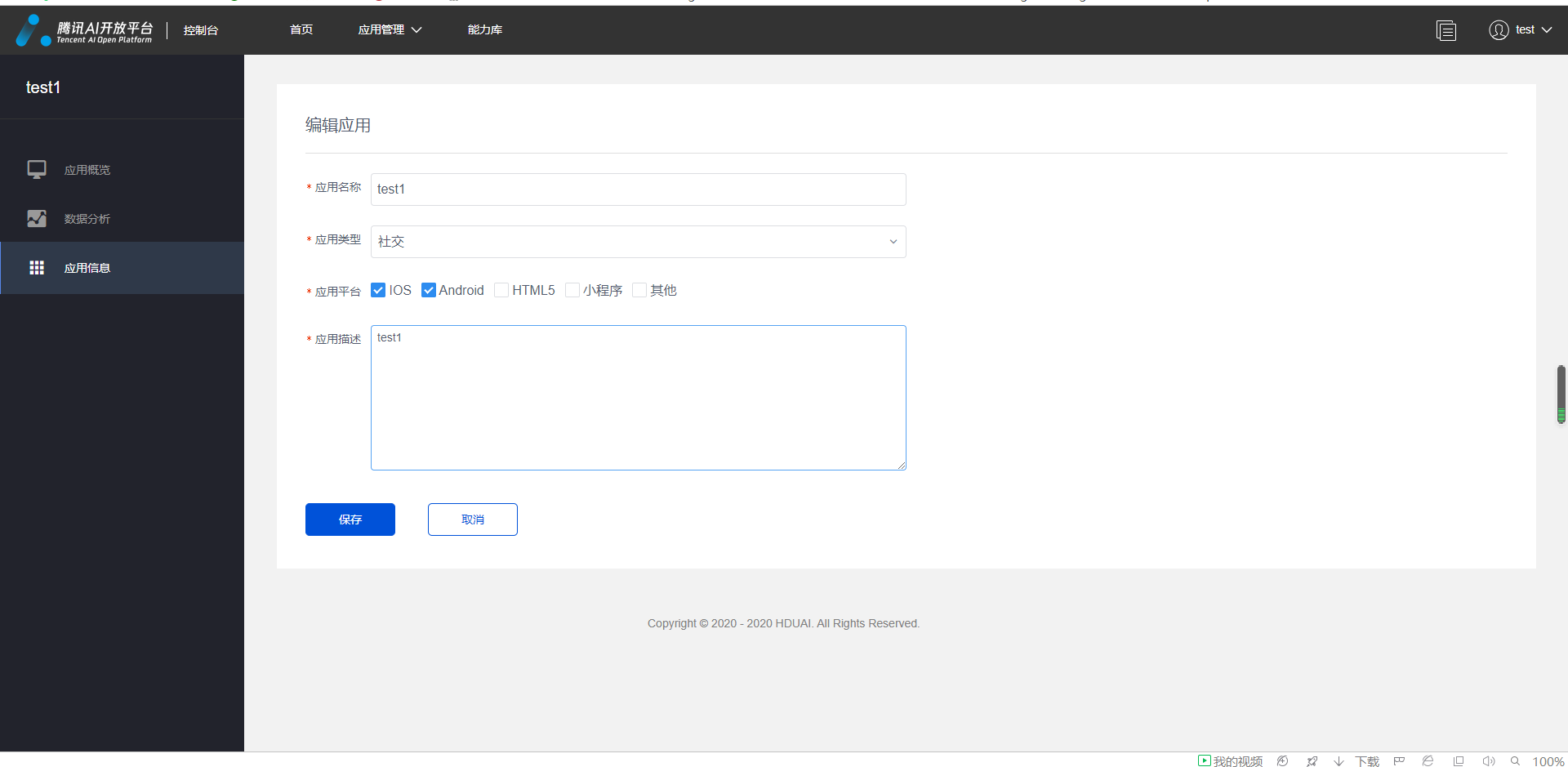




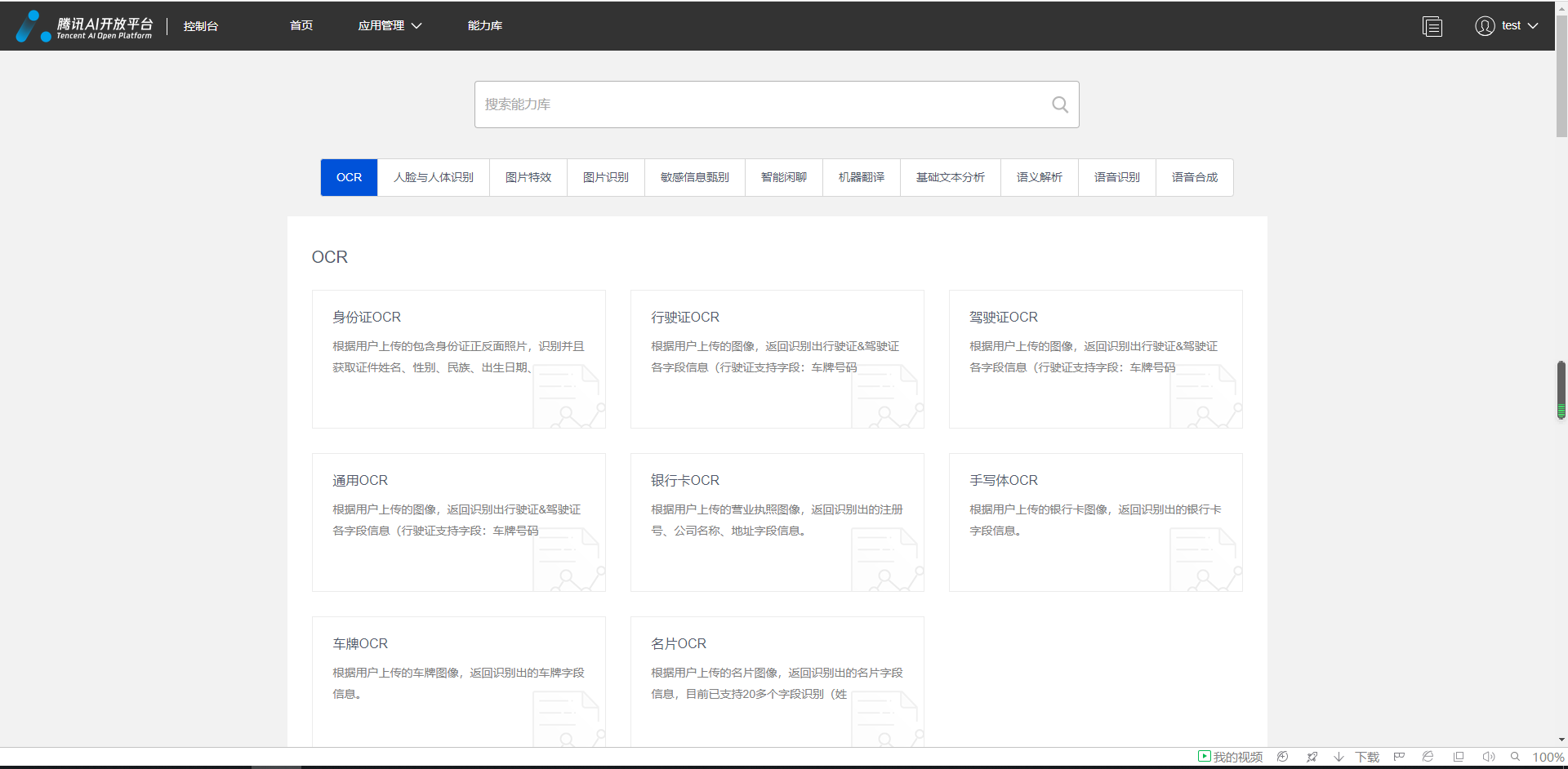


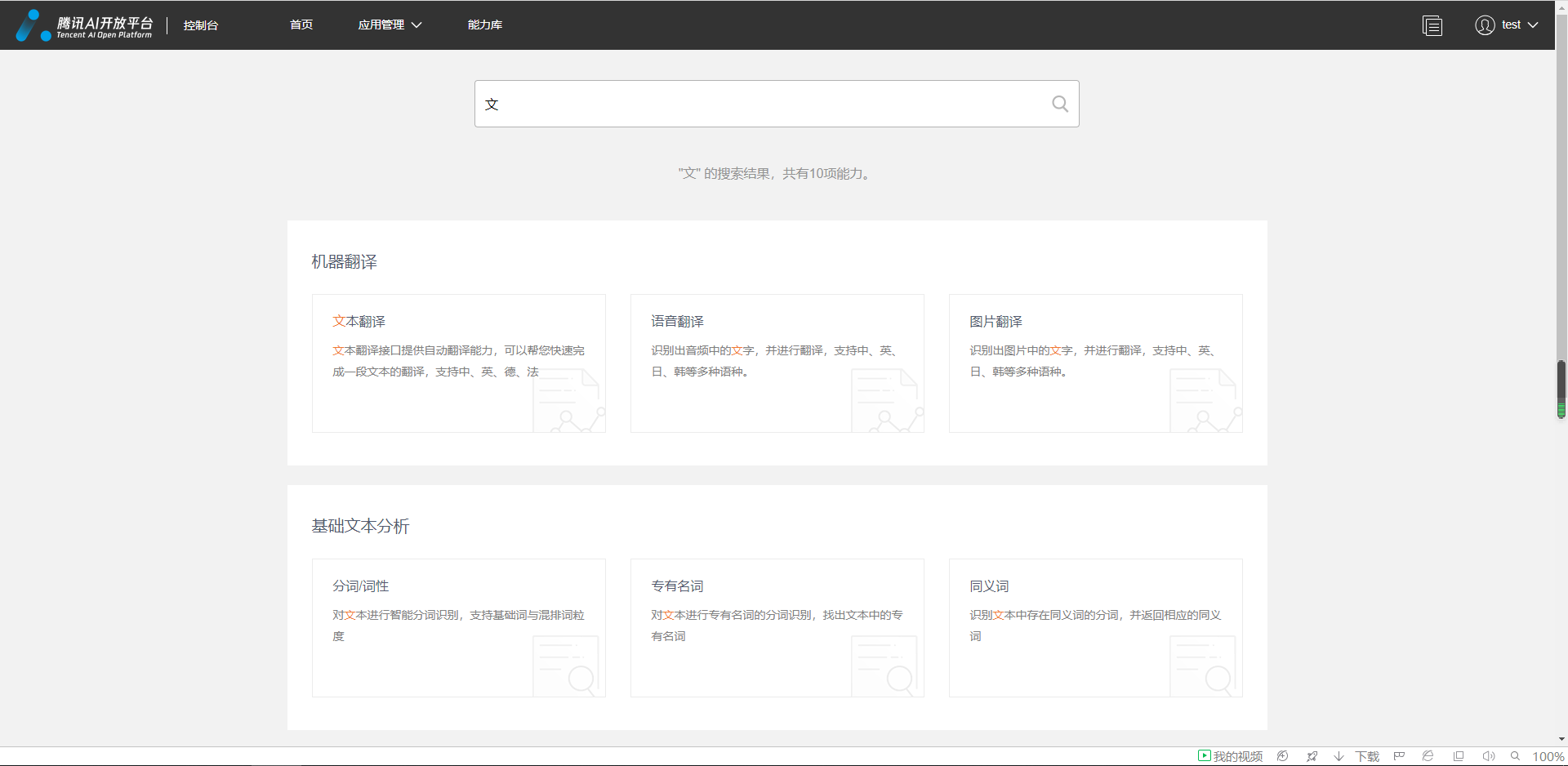


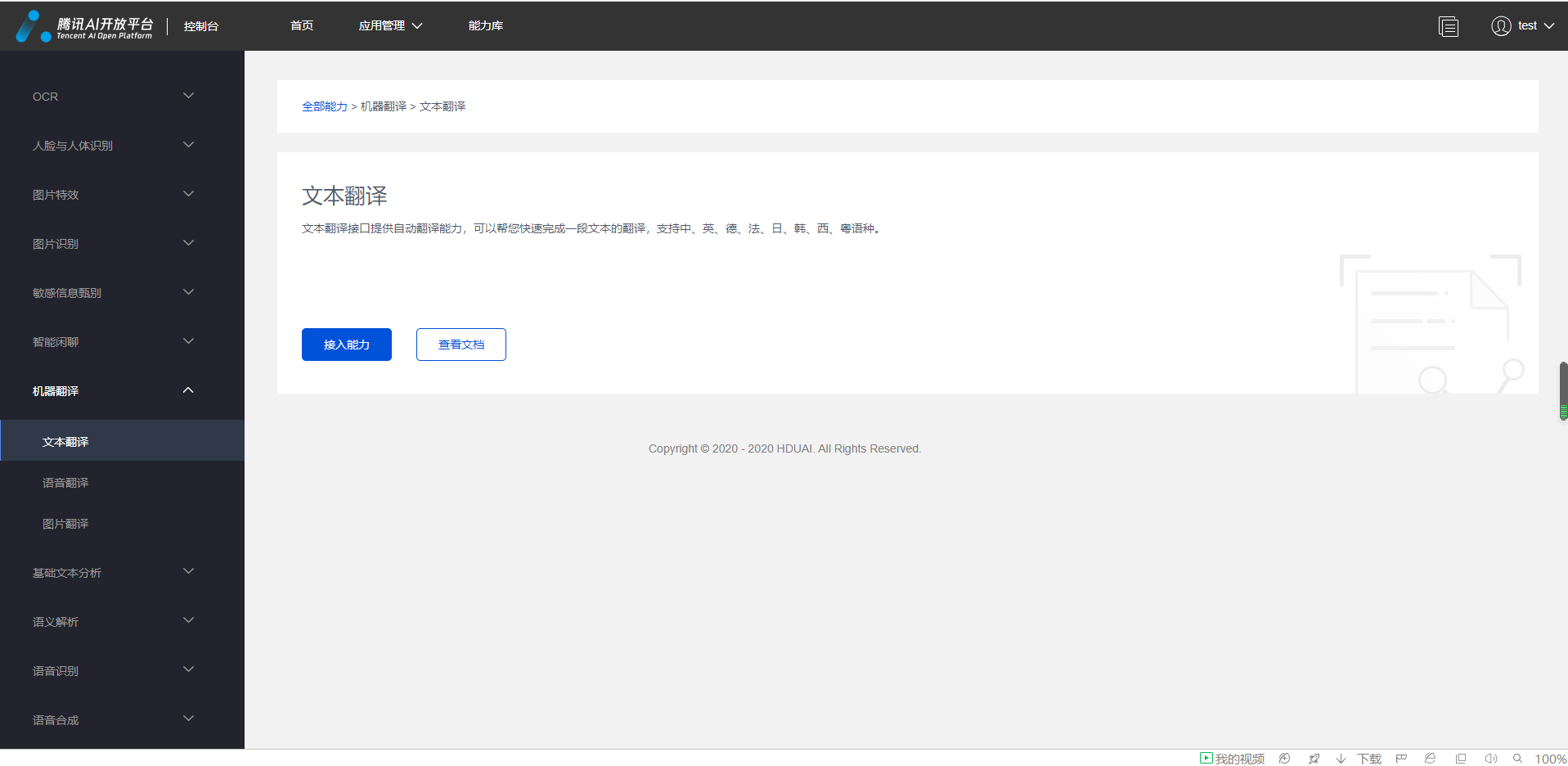


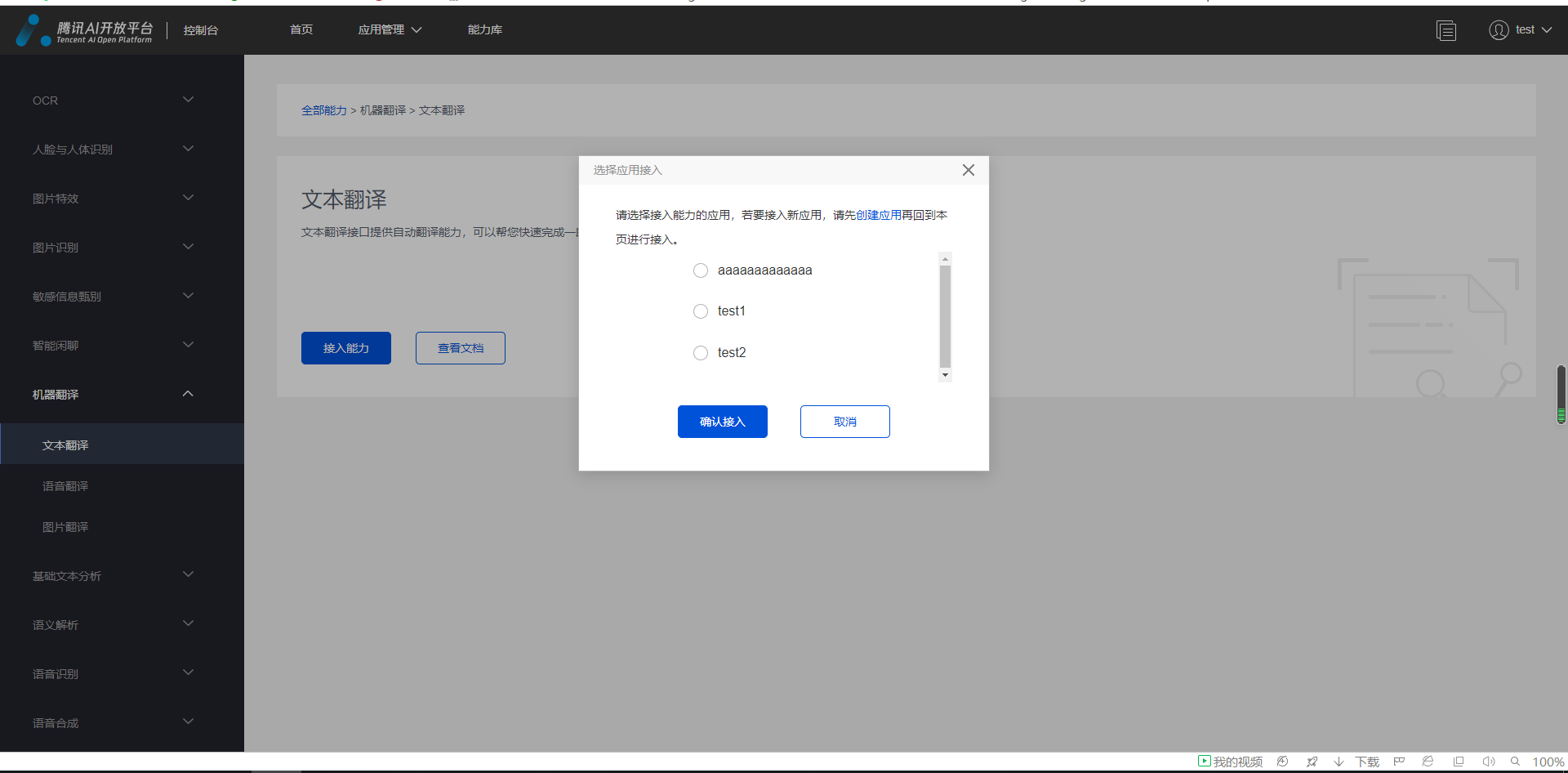


在能力库中，用户可以查看平台提供的所有能力（目前与技术引擎保持一致），用户可以通过模糊查询找到需要的能力。进入能力详情页可以查看能力的介绍以及将能力接入应用和查看文档。接入能力后，该应用即可调用该能力。









控制台顶栏主要包括各个页面的入口，以及退出登录。



### **核心代码介绍**

#### **封装axios**

具体代码见libs/axios.js

该文件基于axios对http请求的request和response进行了拦截。

对request进行了请求超时拦截，抛出异常信息。

对response进行了一些状态码的拦截，如果后台返回401状态码，表示未授权，清空本地的token和userInfo，并跳转登录页。

对常见的http请求方法，如get、post等进行了封装。需要向后端发送具体的请求时，只需要在api文件中导入这些方法并传入具体的url即可供具体的页面使用。

#### **Echarts**

简单封装了一个Echarts日期相关的折线图。见components/chart.vue  
其中views/console/application/detail/data-analysis.vue和views/console/application/detail/data-info.vue这两个页面使用了该组件  
使用方法可参考上述文件和文章https://blog.csdn.net/u013556477/article/details/105477430  
如有需要修改图表样式，请参照Echarts官方文档进行修改

#### **关于页面共享的数据**

本项目采用vuex进行全局状态管理。见store目录下的文件  
主要有用户信息，技术引擎目录（能力库和文档中心的目录也是这个）和用户应用列表。  
状态的定义在index.js文件中。  
mutations.js中为同步方法，用来修改index.js中定义的状态。  
actions.js中为异步方法，通过调用api获取后端数据并调用提交mutation来修改状态。  
getters.js类似compute计算属性，用于对index.js状态进行特殊处理，本项目中未使用。  
具体使用方法可参考各个使用了vuex中数据的页面以及官方文档：https://vuex.vuejs.org/zh/

#### **关于vue-router**

本项目的vue-router配置见router/index.js  
具体使用见官方文档：https://router.vuejs.org/zh/installation.html 需要注意的有两点：

1. 无路由匹配时跳转404页面。 需要在路由列表的最后添加以下代码。必须添加在最后，表示与之前路由规则都不匹配的所有路由。

// 无对应路由跳转404

{ path: '\*', component: () => import('@/views/error/404/404.vue')}

1. history模式 history模式利用了 HTML5 History Interface 中新增的 pushState() 和 replaceState() 方法。这两个方法能改变当前的url，但浏览器不会向后端发送对应的请求。  
   需要注意的是，当用户通过url打开某个页面或者进行刷新操作时，浏览器会像这个url发送http请求，但是实际上服务器中并没有对应的html资源，即返回404找不到资源。因此，使用history模式时，需要服务器进行支持。 之前我介绍了通过docker的nginx容器进行部署该项目，现在，我将以此为例介绍如何配置服务器解决上述问题。 请看docker/config/default.conf文件，这是nginx的配置文件。 请看以下代码和注释

# 表示匹配/hdu-ai-web请求

location /hdu-ai-web {

# 对应/hdu-ai-web请求，会以/usr/share/nginx/html为根目录，因此，如果是这样的配置，我们只需要在html目录下创建hdu-ai-web目录，并将打包后的文件拖入该目录即可，对应的，ip/hdu-ai-web/即为我们的平台首页

root /usr/share/nginx/html;

index index.html index.htm;

# 表示如果匹配不到资源，则会返回/hdu-ai-web/index.html文件

try\_files $uri $uri/ /hdu-ai-web/index.html;

}

# 表示匹配/api请求（/api是本项目的后台接口前缀）

location /api {

# 表示/api请求会转发到proxy\_pass对应的主机，即实现反向代理解决跨域问题（此处与history模式无关，讲到nginx配置顺带提一下反向代理解决跨域问题）

proxy\_pass http://47.99.47.230:8090;

}

更详细的nginx配置请自行搜索nginx配置相关的文章

#### **关于vue的响应式原理**

由于 JavaScript 的限制，Vue 不能检测数组和对象的变化。 假设有以下data

data:{

object:{

k1: 'v1',

},

arr:[1,2,3]

}

如果使用以下代码来更新这些data，将会出现数值被修改，而视图并未重新渲染的情况，即vue没有检测到数组和对象的变化。

this.object.k2 = 'v2';

this.arr[1] = 22;

因为对于对象，vue无法检测 property 的添加或移除。我们可以用以下代码来添加响应式的property

this.$set(this.object,'k2','v2');

// 或者创建新的对象赋值给this.object

this.object = Object.assign({}, this.object, { a: 1, b: 2 })

对于数组，Vue 不能检测以下数组的变动：

1. 当你利用索引直接设置一个数组项时，例如：vm.items[indexOfItem] = newValue
2. 当你修改数组的长度时，例如：vm.items.length = newLength

为解决这一问题可以使用

// Vue.set方法

this.$set(vm.items, indexOfItem, newValue)

// Array.prototype.splice方法

this.arr.splice(1, 1, 22)

// 对于问题二：修改数组长度，可以使用Array.prototype.splice(newLength)

// 长度变为1

this.arr.splice(1)

更详细的介绍可参考官方文档：https://cn.vuejs.org/v2/guide/reactivity.html

### **项目总结**

本项目的核心功能基本完成。  
后续开发可考虑完善以下功能：

* 技术引擎的功能体验：目前只有机器翻译-文字翻译的功能体验，后续可根据具体算法开发对应的功能体验
* 平台的图标和页脚信息等：目前图标和页脚信息基本直接使用了腾讯AI开放平台，后续可根据实际需要替换对应的图标和信息
* 文档详情页：目前文档详情页的内容为硬编码的md文件，后续可根据后端提供的接口替换成对应的md文件内容或者使用富文本的形式