多租户管理系统

源码文档

版本：V1.0

雷山碧阅科技有限公司

2024-7-11

File：src/pages/adminList/index.tsx

```tsx
import {PlusOutlined} from '@ant-design/icons';
import {Button, Select, message} from 'antd';
import React, {useEffect, useRef, useState} from 'react';
import {PageContainer} from '@ant-design/pro-layout';
import type {ActionType, ProColumns} from '@ant-design/pro-table';
import ProTable from '@ant-design/pro-table';
import ProForm, {ModalForm, ProFormSelect, ProFormText} from '@ant-design/pro-form';
import {
    addAdmin,
    adminList,
    roleSelect,
    updataPassword,
    updataStatus,
    updateAdmin,
} from './service';
import type {TableListItem, TableListPagination} from './data';
import {formatTime} from '@/utils';
import {checkPassword} from '@/utils/rolue';
import ImgUpload from '@/components/ossUpload';
import styles from './index.less';
import {history} from 'umi';
import {ProCard} from '@ant-design/pro-components';
import {listTenant} from '@/api/tenant';

/**
 * 添加节点
 *
 * @param fields
 */
const handleAdd = async (fields: any) => {
    const hide = message.loading('正在添加');
    try {
        await addAdmin({...fields});
        hide();
        message.success('添加成功');
        return true;
    } catch (error) {
        hide();
        return false;
    }
};
/**
 * 更新节点
 *
 * @param fields
 */
const handleUpdate = async (fields: any, currentRow?: TableListItem) => {
    const hide = message.loading('正在修改');
    try {
        await updateAdmin({
            guid: currentRow?.guid,
```

```
                avatar: currentRow?.currentRow,
                ...fields,
            });
            hide();
            message.success('修改成功');
            return true;
        } catch (error) {
            hide();
            return false;
        }
    };
    /**
     * 修改密码
     * @param fields
     * @param currentRow
     * @returns
     */
    const handlePasswordUpdate = async (fields: any, currentRow?: TableListItem) => {
        const hide = message.loading('正在修改');
        try {
            await updataPassword({
                guid: currentRow?.guid,
                ...fields,
            });
            hide();
            message.success('修改成功');
            return true;
        } catch (error) {
            hide();
            return false;
        }
    };
    // 添加或者修改密码表单
    const passWordBOX = () => (
        <>
            <ProFormText.Password
                // @ts-ignore
                rules={[
                    {
                        required: true,
                        message: '用户密码为必填项',
                    },
                    ({getFieldValue}: any) => ({
                        validator(role: any, value: string) {
                            const password_value = getFieldValue('passwords');
                            const data = checkPassword(value);
                            if (!data.status) {
                                return Promise.reject(data.message);
                            }
                            if (password_value && password_value !== value) {
                                return Promise.reject('两次密码不一致');
                            }
                            return Promise.resolve();
```

```
                    },
                }),
            ]}
            width="md"
            name="password"
            label="用户密码"
        />
        <ProFormText.Password
            rules={[
                {
                    required: true,
                    message: '请确认密码',
                },
                ({getFieldValue}: any) => ({
                    validator(role: any, value: string) {
                        const password_value = getFieldValue('password');
                        if (password_value && password_value !== value) {
                            return Promise.reject('两次密码不一致');
                        }
                        return Promise.resolve();
                    },
                }),
            ]}
            width="md"
            name="passwords"
            label="确认密码"
        />
    </>
);
// 修改或添加表单 name role avatar
// @ts-ignore
const EditBox = (tenant = 'master') => (
    <>
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '用户名称为必填项',
                },
            ]}
            width="md"
            name="nick_name"
            label="用户昵称"
        />
        <ProForm.Item
            name="avatar"
            label="头像"
            rules={[
                {
                    required: true,
                    message: '请上传头像',
                },
            ]}
```

```
            >
                <ImgUpload
                    needCrop={true}
                    //@ts-ignore
                    cropProps={{aspect: 1, rotate: true}}
                    maxSize={0.5}
                    pictureQuality={0.7}
                />
            </ProForm.Item>
            <ProFormSelect
                name="role_guid"
                label="角色名称"
                showSearch
                width="md"
                debounceTime={300}
                request={async ({keyWords}) => {
                    try {
                        const data = await roleSelect({name: keyWords, tenant_id: tenant});
                        return data.map((item: { guid: string; name: string }) => {
                            return {
                                value: item.guid,
                                label: item.name,
                            };
                        });
                    } catch (error) {
                        return [];
                    }
                }}
                placeholder="请选择角色,输入可搜索"
                rules={[{required: true, message: '请选择角色'}]}
            />
        </>
    );
    const TableList: React.FC = () => {
        const [query] = useState<any>(history.location.query);
        /** 新建窗口的弹窗 */
        const [createModalVisible, handleModalVisible] = useState<boolean>(false);
        const [tablePage, handleTablePage] = useState<string>('saas');
        const [updateModalVisible, handleUpdateModalVisible] = useState<boolean>(false);
        const [updatePasswordModalVisible, handleupdatePasswordModalVisible] = useState<boolean>(false);
        const actionRef = useRef<ActionType>();
        const [currentRow, setCurrentRow] = useState<TableListItem>();
        const [tenantList, setTenantList] = useState([]);
        const [currentTenant, setCurrentTenant] = useState<string>('');
        const columns: ProColumns<TableListItem>[] = [
            {
                title: 'guid',
                width: 150,
                dataIndex: 'guid',
                ellipsis: true,
                copyable: true,
            },

```

```
        {
            title: '账号',
            dataIndex: 'admin_name',
            ellipsis: true,
        },
        {
            title: '用户昵称',
            dataIndex: 'nick_name',
            ellipsis: true,
        },
        {
            title: '用户头像',
            dataIndex: 'avatar',
            search: false,
            render: (_: any, record: TableListItem) => [
                <img key="avatar" className={styles.avatarImg} src={record.avatar} alt="头像"/>,
            ],
        },
        {
            title: '角色',
            dataIndex: 'role_name',
            ellipsis: true,
        },
        {
            title: '状态',
            dataIndex: 'status',
            hideInForm: true,
            initialValue: '1',
            valueEnum: {
                1: {
                    text: '正常',
                    status: 'Processing',
                },
                2: {
                    text: '关闭',
                    status: 'Default',
                },
            },
        },
        {
            title: '创建时间',
            dataIndex: 'created_at',
            search: false,
            renderText: (val: number) => formatTime(val, ''),
        },
        {
            title: '最后登录时间',
            dataIndex: 'last_time',
            search: false,
            renderText: (val: number) => formatTime(val, ''),
        },
        {
            title: '操作',
```

```
            dataIndex: 'option',
            valueType: 'option',
            width: 250,
            render: (_: any, record: TableListItem) => [
                <Button
                    key="config"
                    type="link"
                    onClick={() => {
                        setCurrentRow(record);
                        handleUpdateModalVisible(true);
                    }}
                >
                    编辑
                </Button>,
                <Button
                    key="password"
                    type="link"
                    onClick={() => {
                        setCurrentRow(record);
                        handleupdatePasswordModalVisible(true);
                    }}
                >
                    修改密码
                </Button>,
                <Button
                    key="status"
                    type="link"
                    danger={record.status === 1}
                    onClick={() => {
                        updataStatus({tenant_id: currentTenant, guid: record.guid, status: record.status === 1 ? 2 : 1})
                            .then(() => {
                                message.success('修改成功');
                                if (actionRef.current) {
                                    actionRef.current.reload();
                                }
                            })
                            .catch(() => {
                            });
                    }}
                >
                    {record.status === 1 ? '禁止' : '启用'}
                </Button>,
            ],
        },
    ];
    const getTenantList = async () => {
        let data = await listTenant({});
        if (data) {
            data = data.data?.map((item: any) => {
                return {label: item.title, value: item.id}
            })
            setTenantList(data);
        }
```

```
};
useEffect(() => {
    getTenantList();
}, []);

return (
    <PageContainer
        tabList={[
            {
                tab: 'saas 管理员管理',
                key: 'saas',
            },
            {
                tab: '租户管理员管理',
                key: 'tenant',
            },
        ]}
        onTabChange={(value) => {
            handleTablePage(value)
        }
        }
    >
        {tablePage === 'saas' ? null : <ProCard bordered>
    <span
        style={{marginLeft: 48}}>
      当前租户 :
    </span>
            <Select
                value={currentTenant}
                style={{width: 280, marginLeft: 8}}
                onChange={async (value) => {
                    setCurrentTenant(value);
                    actionRef.current?.reload()
                }}
                options={tenantList}
            />

        </ProCard>}
        <ProTable<TableListItem, TableListPagination>
            key={tablePage}
            headerTitle="查询表格"
            actionRef={actionRef}
            // style={{ marginTop: 2 }}
            rowKey="guid"
            search={{
                labelWidth: 120,
            }}
            toolBarRender={() => [
                <Button
                    type="primary"
                    key="primary"
                    onClick={() => {
                        handleModalVisible(true);
```

```
                    }}
                >
                    <PlusOutlined/> 新建
                </Button>,
            ]}
            request={async (params) => {
                if (tablePage == 'saas') {
                    params.tenant_id = 'master';
                } else {
                    if (currentTenant == '') {
                        //@ts-ignore
                        setCurrentTenant(tenantList[0].value)
                        //@ts-ignore
                        params.tenant_id = tenantList[0].value;
                    } else {
                        params.tenant_id = currentTenant;
                    }
                }

                if (query.guid) {
                    params['a.guid'] = query.guid;
                }

                const msg = await adminList(params);
                return {
                    data: msg.list,
                    success: true,
                    total: msg.count,
                };
            }}
            columns={columns}
        />

        {/* 新建 modal */}
        <ModalForm
            title="新建"
            width="600px"
            visible={createModalVisible}
            onVisibleChange={handleModalVisible}
            modalProps={{
                destroyOnClose: true,
            }}
            onFinish={async (value) => {
                const success = await handleAdd({
                    tenant_id: tablePage == 'saas' ? 'master' : currentTenant,
                    avatar: '', ...value
                });
                if (success) {
                    handleModalVisible(false);
                    if (actionRef.current) {
                        actionRef.current.reload();
                    }
                }
```

```
        }}
    >
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '账号为必填项',
                },
            ]}
            width="md"
            name="admin_name"
            label="账号"
        />
        {/* 与编辑组件重复抽离 */}
        {EditBox(tablePage == 'saas' ? 'master' : currentTenant)}
        {/* 与密码组件重复抽离 */}
        {passWordBOX()}
    </ModalForm>
    {/* 编辑 */}
    <ModalForm
        title="编辑"
        width="600px"
        visible={updateModalVisible}
        onVisibleChange={handleUpdateModalVisible}
        modalProps={{
            destroyOnClose: true,
        }}
        initialValues={{
            ...currentRow,
        }}
        onFinish={async (value) => {
            value.tenant_id = currentTenant;
            const success = await handleUpdate(value, currentRow);
            if (success) {
                handleUpdateModalVisible(false);
                setCurrentRow(undefined);
                if (actionRef.current) {
                    await actionRef.current.reload();
                }
            }
        }}
    >
        {EditBox(tablePage == 'saas' ? 'master' : currentTenant)}
    </ModalForm>
    {/* 修改密码 */}
    <ModalForm
        title="编辑"
        width="600px"
        visible={updatePasswordModalVisible}
        onVisibleChange={handleupdatePasswordModalVisible}
        initialValues={{
            password: '',
            passwords: '',
```

```
                    }}
                    modalProps={{
                        destroyOnClose: true,
                    }}
                    onFinish={async (value) => {
                        value.tenant_id = currentTenant;
                        const success = await handlePasswordUpdate(value, currentRow);

                        if (success) {
                            handleupdatePasswordModalVisible(false);
                            setCurrentRow(undefined);
                            if (actionRef.current) {
                                actionRef.current.reload();
                            }
                        }
                    }}
                >
                    {passWordBOX()}
                </ModalForm>
            </PageContainer>
        );
};
export default TableList;

```
File: src/pages/case/index.tsx
```tsx
import {ModalForm, ProFormDependency, ProFormSelect, ProFormText} from "@ant-design/pro-form"
import {PageContainer} from "@ant-design/pro-layout";
import type { InputRef, TableColumnsType, TableColumnType} from "antd";
import {Button, Form, Input, message, Space, Table} from "antd";
import {useRef, useState} from "react";
import Highlighter from 'react-highlight-words';
import type {FilterDropdownProps} from "antd/es/table/interface";
import {SearchOutlined} from "@ant-design/icons";
import {deleteRedis} from "@/pages/case/service";

// 类别产品模版
export const productTemplate = [
  {
    label: '快递零售',
    value: 'entity_retail',
    status: {30: '待确认', 31: '待发货', 32: '待收货'},
  },
  {label: '酒店住宿', value: 'hotel_lodging', status: {41: '待确认', 42: '已确认'}},
  {label: '各类门票', value: 'travel_ticket', status: {51: '待出行'}},
  {label: '线下核销', value: 'catering', status: {61: '待使用'}},
  {label: '剧本/道具/大礼包', value: 'props', status: {71: '待发放'}},
  {label: '旅游线路', value: 'tourism', status: {81: '待出行'}},
];
const caseTypeList = [
  {
    value: 'productInfo',
```

```
    param：'产品 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'商品信息缓存',
    page："展示该商品相关的全部页面",
    effect：'存储商品的信息',
    textInput：true
  },
  {
    value：'product_status_search',
    label：'已上架商品缓存列表',
    page："全部商品相关的页面",
    effect：'存储全部已经上架的商品 id',
  },
  {
    value：'product_category_search',
    param：'类别 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'类别商品缓存',
    page："类别相关的页面，例如集市",
    effect：'存储该类别下全部商品 id',
    textInput：true
  },
  {
    value：'product_type_search',
    param：'开发设置的类型',
    example：'entity_retail',
    label：'类型商品缓存',
    page："产品类型的页面，例如集市、店铺详情中的商品列表",
    effect：'存储该类型下的全部商品 id',
    selectInput：productTemplate
  },
  {
    value：'product_integral_search',
    label：'积分商品缓存',
    page："积分商品列表页面",
    effect：'存储全部的积分商品 id',
  },
  {
    value：'product_hide_search',
    label：'隐藏商品缓存',
    page："涉及集市、店铺详情中的商品列表",
    effect：'存储全部的隐藏商品 id',

  },
  {
    value：'product_store_search',
    param：'店铺 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'商品店铺缓存',
    page：" 涉及店铺详情中的商品列表",
    effect：'存储该店铺下的全部商品 id',
    textInput：true
  },
```

```
  {
    value：'product_name_search',
    param：'搜索名称',
    example：'千户苗寨',
    label：'商品搜索信息缓存',
    page：" 涉及到产品搜索功能的页面，例如集市，产品搜索页面等",
    effect：'存储该搜索内容下全部相关的商品 id',
    textInput：true
  },

  {
    value：'product_weight_search',
    label：'商品权重缓存有序集合',
    page：" 涉及商品列表排序",
    effect：'存储全部商品的权重',
  },
  {
    value：'gameUserInfo',
    param：'用户 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'用户游戏相关的展示缓存（卡券数量，等级，经验，剧本等）',
    page：" 涉及我的和令牌页面",
    effect：'存储用户游戏相关数据',
    textInput：true
  },
  {
    value：'guidePropList',
    label：'引游人首页卡券列表缓存',
    page：" 涉及引游人推荐卡券列表",
    effect：'存储引游人卡券推荐列表页面数据',
  },
  {
    value：'guideGrantPropList',
    param：'引游人 id 或者合伙人 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'引游人或合伙人发放道具列表',
    page：" 涉及引游人和合伙人的已发放卡券列表",
    effect：'存储引游人或合伙人发放道具列表列表数据',
    textInput：true
  },
  {
    value：'hotelDayStock',
    param：'yyyy-mm-dd 格式的日期',
    example：'2024-09-01',
    label：'指定日期当天有库存的酒店的缓存',
    page：" 涉及全部酒店相关产品",
    effect：'存储当天有酒店库存的全部店铺 id',
    textInput：true
  },
  {
    value：'orderDetails',
    param：'订单 id',
    example：'298417635459072000',
```

```
    label：'订单信息缓存',
    page：'' 涉及订单详情页面'',
    effect：'存储订单详情',
    textInput：true
  },
  {
    value：'orderList',
    param：'用户 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'用户订单列表缓存',
    page：'' 涉及订单列表页面'',
    effect：'存储用户订单列表',
    textInput：true
  },
  {
    value：'userAccount',
    param：'用户 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'用户帐户金币缓存',
    page：'' 涉及我的页面'',
    effect：'存储用户金币信息',
    textInput：true
  },

  {
    value：'followList',
    param：'类型(guide_prop、product、store):用户 id',
    example：'guide_prop:25f2e9c4f1844f1faa154dd41ca40122',
    label：'收藏列表缓存',
    page：'' 涉及收藏相关的列表页面'',
    effect：'存储用户收藏列表',
    textInput：true
  },
  {
    value：'followUser',
    param：'用户 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'用户收藏缓存',
    page：'' 涉及用户是否收藏了某项内容'',
    effect：'存储用户收藏的全部内容',
    textInput：true
  },
  {
    value：'storeInfo',
    param：'店铺 id',
    example：'25f2e9c4f1844f1faa154dd41ca40122',
    label：'店铺信息缓存',
    page：'' 涉及店铺详情页面'',
    effect：'存储店铺详情',
    textInput：true
  },
  {
    value：'contentVillage',
```

```
    label：'首页村寨缓存',
    page：'' 涉及首页雷山地图'',
    effect：'存储雷山地图信息',
  },
  {
    value：'marketCategory',
    label：'集市类别缓存',
    page：'' 涉及集市类别数据'',
    effect：'存储集市类别下的类别树',
  },
]

interface DataType {
  value：string;
  label：string;
  param?：string,
  example?：string,
  page：string;
  effect：string;
}
type DataIndex = keyof DataType;
const CaseForm = () => {
  const [searchText, setSearchText] = useState('');
  const [searchedColumn, setSearchedColumn] = useState('');
  const searchInput = useRef<InputRef>(null);
  const [open, setOpen] = useState(false)
  const [form] = Form.useForm<{ type：string; param?：string }>();
  const handleSearch = (
    selectedKeys：string[],
    confirm：FilterDropdownProps['confirm'],
    dataIndex：DataIndex,
  ) => {
    confirm();
    setSearchText(selectedKeys[0]);
    setSearchedColumn(dataIndex);
  };

  const handleReset = (clearFilters：() => void) => {
    clearFilters();
    setSearchText('');
  };
  const getColumnSearchProps = (dataIndex：DataIndex)：TableColumnType<DataType> => ({
    filterDropdown：({setSelectedKeys, selectedKeys, confirm, clearFilters, close}) => (
      <div style={{padding：8}} onKeyDown={(e) => e.stopPropagation()}>
        <Input
          ref={searchInput}
          placeholder={`Search ${dataIndex}`}
          value={selectedKeys[0]}
          onChange={(e) => setSelectedKeys(e.target.value ? [e.target.value] ：[])}
          onPressEnter={() => handleSearch(selectedKeys as string[], confirm, dataIndex)}
          style={{marginBottom：8, display：'block'}}
        />
        <Space>
```

```
            <Button
              type="primary"
              onClick={() => handleSearch(selectedKeys as string[], confirm, dataIndex)}
              icon={<SearchOutlined/>}
              size="small"
              style={{width: 90}}
            >
              搜索
            </Button>
            <Button
              onClick={() => clearFilters && handleReset(clearFilters)}
              size="small"
              style={{width: 90}}
            >
              重置
            </Button>
            <Button
              type="link"
              size="small"
              onClick={() => {
                confirm({closeDropdown: false});
                setSearchText((selectedKeys as string[])[0]);
                setSearchedColumn(dataIndex);
              }}
            >
              过滤
            </Button>
            <Button
              type="link"
              size="small"
              onClick={() => {
                close();
              }}
            >
              关闭
            </Button>
          </Space>
        </div>
      ),
      filterIcon: (filtered: boolean) => (
        <SearchOutlined style={{color: filtered ? '#1677ff' : undefined}}/>
      ),
      onFilter: (value, record) => {
        //@ts-ignore
        return record[dataIndex]
          .toString()
          .toLowerCase()
          .includes((value as string).toLowerCase())
      },
      onFilterDropdownOpenChange: (visible) => {
        if (visible) {
          setTimeout(() => searchInput.current?.select(), 100);
        }
```

```
    },
    render: (text) =>
      searchedColumn === dataIndex ? (

        <Highlighter
          highlightStyle={{backgroundColor: '#ffc069', padding: 0}}
          searchWords={[searchText]}
          autoEscape
          textToHighlight={text ? text.toString() : ''}
        />
      ) : (
        text
      ),
  });

  const columns: TableColumnsType<DataType> = [
    {
      title: '名称',
      dataIndex: 'label',
      key: 'label',
      ...getColumnSearchProps('label'),
    },
    {
      title: '作用',
      dataIndex: 'effect',
      key: 'effect',
    },
    {
      title: '涉及页面',
      dataIndex: 'page',
      key: 'page',
    },
    {
      title: '所需参数',
      dataIndex: 'param',
      key: 'param',
      render: (_, record) => record?.param ?? '—'
    },
    {
      title: '例子',
      dataIndex: 'example',
      key: 'example',
      render: (_, record) => record?.example ?? '—'
    },
    {
      title: '操作',
      key: 'action',
      width: 150,
      render: (_, record) => (
        <Space size="middle">
          <a onClick={() => {
            // @ts-ignore
            form.setFieldValue('type', record.value)
```

```
                setOpen(true)

              }}>
                更新缓存
              </a>
            </Space>
        )
      }
  ];
  return <>
    <PageContainer title="缓存更新">
      <Table columns={columns} dataSource={caseTypeList}/>
      <ModalForm
        visible={open}
        onVisibleChange={setOpen}
        form={form}
        onFinish={async (values) => {
          // eslint-disable-next-line @typescript-eslint/no-shadow
          const data: string = await deleteRedis(values)
          if (data.includes('成功')) {
            message.success(data);
          }

          return true;
        }}
      >
        {/* eslint-disable-next-line react/jsx-no-undef */}
        <ProFormSelect
          options={caseTypeList}
          disabled
          required
          showSearch
          name="type"
          label="缓存类型"
        />
        {/* eslint-disable-next-line react/jsx-no-undef */}
        <ProFormDependency name={['type']}>
          {({type}) => {
            // eslint-disable-next-line @typescript-eslint/no-shadow
            const item = caseTypeList.find((item: any) => item.value === type);
            if (!item) {
              return null
            }
            if (item.textInput) {
              return (<>
                <p>参数要求：{item.param}</p>
                <p>示例：{item.example}</p>
                <ProFormText
                  name='param'
                  required
                  label="参数"
                  placeholder={item.param}
                /></>)
```

```
                }
                if (item.selectInput) {
                  return <ProFormSelect
                    options={item.selectInput}
                    width="md"
                    required
                    name="param"
                    label="产品类型"
                  />
                }
                return null
              }}
            </ProFormDependency>

          </ModalForm>
        </PageContainer>
      </>

}
export default CaseForm
```

File：src/pages/customer/user/list/index.tsx

```tsx
import {Avatar, Button, message, Modal, Select, Switch} from 'antd';
import {PageContainer} from '@ant-design/pro-layout';
import type {ActionType, ProColumns} from '@ant-design/pro-table';
import ProTable from '@ant-design/pro-table';
import {exportUserInfoExcel, getAccountAndPassword, updateUserStatus, userList} from '../service';
import type {TableListItem, TableListPagination} from './data';
import styles from './index.less';
import React, {useEffect, useRef, useState} from 'react';
import {FileExcelOutlined} from '@ant-design/icons';
import TagNames from '@/components/TagNames';
import DateFormat from '@/components/DateFormat';
import moment from 'moment';
import UserDetailForm from '../components/UserDetailForm';
import {history} from 'umi';
import UpdateUserInfoModal from '@/pages/customer/user/components/UpdateUserInfoModal';
import UpdatePasswordModal from '@/pages/customer/user/components/UpdatePasswordModal';
import {listTenant} from "@/api/tenant";
import {ProCard} from "@ant-design/pro-components";

const query: any = history.location.query;
const {confirm} = Modal;
const TableList: React.FC = () => {
    /** 新建窗口的弹窗 */
    const actionRef = useRef<ActionType>();
    const [currentRow, setCurrentRow] = useState<TableListItem>();
    const [updateUserVisible, handleUpdateUserVisible] = useState<boolean>(false);
    const [updatePasswordVisible, handleUpdatePasswordVisible] = useState<boolean>(false);

    //用户禁用状态
    const [switchDisabled, setSwitchDisabled] = useState<boolean>(false);
```

```
const [accountAndPassword, setAccountAndPassword] = useState<{
    username: undefined | string;
    password: undefined | string;
}>({
    username: undefined,
    password: undefined,
});
const [userModalVisible, updateUserModalVisible] = useState(false);
const [userGuid, updateUserGuid] = useState('');
const [selectedRowKeyList, setSelectedRowKeyList] = useState<any[]>([]);
const [tenantList, setTenantList] = useState<any[]>([]);
const [tabPageKey, handlerTabPageKey] = useState<string>('user');
const [currentTenant, setCurrentTenant] = useState<string>('');
const getTenantList = async () => {
    let data = await listTenant({});
    if (data) {
        data = data.data?.map((item: any) => {
            return {label: item.title, value: item.id}
        })
        setTenantList(data);
    }
};
useEffect(() => {
    if (query.guid) {
        updateUserGuid(query.guid);
        updateUserModalVisible(true);
    }

    getTenantList();

    return () => {
        updateUserGuid('');
    };
}, []);
const onSelectChange: any = (rowKeys: any[]) => {
    // 拿到之前选中的数据
    const temp: any = [...selectedRowKeyList];
    // 将当前选中的数据与之前选中的数据合并,去重
    const tempArr = [...temp, ...rowKeys];
    const tempSet = new Set(tempArr);
    const newArr = Array.from(tempSet);
    // 将合并后的数据赋值给选中的数据
    setSelectedRowKeyList(newArr);
};
const onEditClick = (record: any) => {
    getAccountAndPassword({guid: record.guid}).then((res) => {
        if (res.username) {
            setAccountAndPassword(res);
            setCurrentRow(record);
            handleUpdatePasswordVisible(true);
        } else {
            message.error('该用户没有账号,请先创建账户').then((r) => r);
        }
```

```
        });
    };
    const columns: ProColumns<TableListItem>[] = [
        {
            title: 'guid',
            dataIndex: 'guid',
            // width: 100,
            copyable: true,
            ellipsis: true,
        },
        {
            title: '用户头像',
            dataIndex: 'headImg',
            search: false,
            render: (_: any, record: TableListItem) => (
                <Avatar key="avatar" className={styles.avatarImg} src={record.headImg} alt="头像"/>
            ),
        },
        {
            title: '昵称',
            dataIndex: 'nickName',
            ellipsis: true,
        },
        {
            title: '性别',
            dataIndex: 'gender',
            valueType: 'select',
            valueEnum: {
                1: {
                    text: '男',
                },
                2: {
                    text: '女',
                },
            },
            search: false,
        },
        {
            title: '标签',
            // width: 150,
            search: false,
            dataIndex: 'tagNames',
            render: (_, row) => <TagNames tagNames={row.tagNames}/>,
        },
        {
            title: '手机号',
            dataIndex: 'mobilePhone',
            ellipsis: true,
            copyable: true,
        },
        {
            title: '是否是商户',
            dataIndex: 'isMerchant',
```

```
            valueType: 'select',
            valueEnum: {
                1: {
                    text: '否',
                },
                0: {
                    text: '是',
                },
            },
        },
        {
            title: '注册时间',
            dataIndex: 'createTime',
            search: false,
            render: (text: any) => <DateFormat date={text}/>,
        },
        {
            title: '注册时间',
            dataIndex: 'createTime',
            valueType: 'dateRange',
            hideInTable: true,
            search: {
                transform: (value) => {
                    return {
                        createStart: moment(value[0]).startOf('day').unix(),
                        createEnd: moment(value[1]).endOf('day').unix(),
                    };
                },
            },
        },
        {
            title: '状态',
            dataIndex: 'status',
            valueType: 'select',
            valueEnum: {
                '0': {text: '正常'},
                '1': {text: '禁用'},
            },
            render: (_, record) => (
                <Switch
                    disabled={switchDisabled}
                    onClick={() => {
                        confirm({
                            title: record.status === 1 ? '是否要启用该用户?' : '是否要禁用该用户?',
                            onOk() {
                                setSwitchDisabled(true);
                                updateUserStatus({
                                    guid: record.guid,
                                    status: record.status === 1 ? 0 : 1,
                                })
                                    .then(() => {
                                        setSwitchDisabled(false);
                                        message.success('修改成功').then((r) => r);
```

```
                                    Modal.destroyAll();

                                    if (actionRef.current) {
                                        actionRef.current.reload();
                                    }
                                })
                                .catch(() => {
                                    setSwitchDisabled(false);
                                    message.error('修改失败').then((r) => r);
                                    return false;
                                });
                        },
                    });
                }}
                checked={record.status === 0}
            />
        ),
    },
    {
        title: '操作',
        dataIndex: 'option',
        valueType: 'option',
        width: 250,
        render: (_: any, record: TableListItem) => [
            // <Access
            //     accessible={
            //         menuKeys.includes(path + userListAction.updatePwd) &&
            //         currentUser?.menu[path + userListAction.updatePwd]?.type == 2
            //     }
            // >
            <Button
                style={{padding: 0}}
                key="update"
                type="link"
                onClick={() => {
                    // const formValue = {
                    //   ...record
                    // }
                    setCurrentRow(record);
                    handleUpdateUserVisible(true);
                }}
            >
                编辑
            </Button>,
            // </Access>,
            // <Access
            //     accessible={
            //         menuKeys.includes(path + userListAction.edit) &&
            //         currentUser?.menu[path + userListAction.edit]?.type == 2
            //     }
            // >
            <Button
                style={{padding: 0}}
```

```
                        disabled={record.hasAccount == '1'}
                        key={'updatePassword'}
                        type={'link'}
                        onClick={() => onEditClick(record)}
                    >
                        修改密码
                    </Button>,
                    // </Access>,
                    <Button
                        style={{padding: 0}}
                        key="detail"
                        type="link"
                        onClick={() => {
                            // history.push("/customer/user/detail?guid=" + record.guid);
                            // 打开用户详情弹窗
                            updateUserGuid(record.guid);
                            updateUserModalVisible(true);
                        }}
                    >
                        详情
                    </Button>,
                ],
            },
        ];

        /**
         * 导出用户数据
         * @param selectedRowKeys
         */
        const exportUserData = (selectedRowKeys: (number | string)[]) => {
            return () => {
                const params = {
                    guids: selectedRowKeys,
                };
                if (selectedRowKeys.length > 0) {
                    // 请求接口,获取 excel 文件
                    exportUserInfoExcel(params)
                        .then((file: any) => {
                            const url = window.URL.createObjectURL(file);
                            const link = document.createElement('a');
                            link.href = url;
                            const filename = `用户信息_${moment().format('YYYY-MM-DD')}`;
                            link.setAttribute('download', filename);
                            link.click();

                            // It's important to revoke the object URL to avoid memory leaks
                            window.URL.revokeObjectURL(url);
                            message.success('导出成功!');
                        })
                        .catch((err) => {
                            message.error('导出失败');
                            console.error(err);
                        });
```

```
            } else {
                message.error('请先选择用户');
            }
        };
    };

    return (
        <PageContainer
            tabList={[
                {
                    key: 'user',
                    tab: '用户列表',
                },
                {
                    key: 'saas',
                    tab: 'SaaS 用户列表',
                },
            ]}
            onTabChange={(key: string) => {
                if (key === 'saas') {
                    setCurrentTenant(tenantList[0].value);
                } else {
                    setCurrentTenant('');
                }
                handlerTabPageKey(key);
                actionRef.current?.reload();
            }}
        >

            {tabPageKey == 'user' ? null : <ProCard bordered>
        <span
            style={{marginLeft: 48}}>
          当前租户 :
        </span>
                <Select
                    value={currentTenant}
                    style={{width: 280, marginLeft: 8}}
                    onChange={async (value) => {
                        setCurrentTenant(value);
                        actionRef.current?.reload();
                    }}
                    options={tenantList}
                />

            </ProCard>}

            <ProTable<TableListItem, TableListPagination>
                headerTitle="用户列表"
                actionRef={actionRef}
                rowKey="guid"
                search={{
                    labelWidth: 120,
                }}
```

```
pagination={{
    showSizeChanger: true,
    showTotal: (total) => `共 ${total} 条`,
}}
request={async (params: any) => {
    const msg = await userList({...params, tenantId: currentTenant});
    msg.data = msg.data.map((item: any) => {
        if (item.isPay) {
            item.isPay = item.isPay.toString();
        }
        if (item.isMerchant) {
            item.isMerchant = item.isMerchant.toString();
        }
        if (item.gender) {
            item.gender = item.gender.toString();
        }
        return item;
    });
    return {
        data: msg.data,
        success: true,
        total: msg.total,
    };
}}
columns={columns}
// 选择框
rowSelection={{
    defaultSelectedRowKeys: selectedRowKeyList,
    onChange: onSelectChange,
    preserveSelectedRowKeys: true,
    onSelectNone: () => {
        setSelectedRowKeyList([]);
    },
}}
tableAlertRender={({selectedRowKeys, onCleanSelected}) => {
    return (
        <div>
            已选择 <a style={{fontWeight: 600}}>{selectedRowKeys.length}</a>{' '}
            项  
            {selectedRowKeys.length > 0 && (
                <a onClick={onCleanSelected} style={{marginLeft: 24}}>
                    取消选择
                </a>
            )}
        </div>
    );
}}
// 下拉选项
tableAlertOptionRender={({selectedRowKeys}) => {
    return [
        <Button
            key={'export'}
            icon={<FileExcelOutlined/>}
```

```
                                type="primary"
                                style={{marginLeft: '20px'}}
                                onClick={exportUserData(selectedRowKeys)}
                            >
                                导出选中的用户
                            </Button>,
                        ];
                    }}
                />

                <UpdateUserInfoModal
                    updateUserVisible={updateUserVisible}
                    handleUpdateUserVisible={handleUpdateUserVisible}
                    currentRow={currentRow}
                    actionRef={actionRef}
                />

                <UpdatePasswordModal
                    updatePasswordVisible={updatePasswordVisible}
                    handleUpdatePasswordVisible={handleUpdatePasswordVisible}
                    currentRow={currentRow}
                    setCurrentRow={setCurrentRow}
                    accountAndPassword={accountAndPassword}
                    setAccountAndPassword={setAccountAndPassword}
                    actionRef={actionRef}
                />

                <UserDetailForm
                    modalVisible={userModalVisible}
                    updateModalVisible={updateUserModalVisible}
                    guid={userGuid}
                    updateGuid={updateUserGuid}
                />
            </PageContainer>
        );
};
export default TableList;
```

File: src/pages/database/fields/index.tsx

```tsx
import React, { useEffect, useRef, useState } from 'react';
import { PageContainer } from "@ant-design/pro-layout";
import type { ProColumns } from "@ant-design/pro-table";
import ProTable from "@ant-design/pro-table";
import { Button, Input, Modal, Select, message } from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { history } from "umi";
import type { ActionType } from "@ant-design/pro-table/lib/typing";
import { createFields, deleteFields, fieldsList, updateFields } from './service';
import { ModalForm, ProFormDigit, ProFormGroup, ProFormItem, ProFormSelect, ProFormSwitch, ProFormText } from
'@ant-design/pro-form';
import type { FieldsList } from './data';
import { FieldsTypeEnum } from './fieldsTypeEnum';
```

```
import { schemaList } from '../schema/service';
import type { SchemaList } from '../schema/data';
import { tableList } from '../table/service';
import type { TableList } from '../table/data';

/**
 * 添加和编辑
 */
const handleUpdate = async (fields: FieldsList, currentRow?: any) => {
    const hide = message.loading('正在修改');
    if (currentRow) {
        if (fields.isAutoIncrementTemp) {
            if (fields.fieldType !== 'bigint' && fields.fieldType !== 'integer') {
                hide();
                message.error("禁止将非 int4、int8 类型的字段设为自增");
                return;
            }
        }
        try {
            await updateFields({
                ...fields,
                isNullable: fields.isNullableTemp,
                isPrimaryKey: fields.isPrimaryKeyTemp,
                isAutoIncrement: fields.isAutoIncrementTemp,
            });
            hide();
            message.success('修改成功');
            return true;
        } catch (error) {
            hide();
            return false;
        }
    } else {
        if (fields.isAutoIncrement) {
            if (fields.fieldType !== 'bigint' && fields.fieldType !== 'integer') {
                hide();
                message.error("禁止将非 int4、int8 类型的字段设为自增");
                return;
            }
        }
        try {
            await createFields({
                ...fields,
            });
            hide();
            message.success('新建成功');
            return true;
        } catch (error) {
            console.log('error');
            console.log(error);
            hide();
            return false;
        }
```

```
    }
};

const TenantList: React.FC = () => {
    const schemaName = history.location.query?.schemaName;
    const tableName = history.location.query?.tableName;
    const actionRef = useRef<ActionType>(null);
    const { confirm } = Modal;
    let deleteValue = '';
    const [modalVisible, handleModalVisible] = useState<boolean>(false);
    const [currentRow, setCurrentRow] = useState<any>();
    const [schemaOptions, setSchemaOptions] = useState([]);
    const [tableOptions, setTableOptions] = useState([]);
    const handleModelDestroy = (field: boolean) => {
        if (!field) {
            setCurrentRow(undefined);
        }
        handleModalVisible(field);
    }
    const getSchemaList = async () => {
        const res = await schemaList({ current: 1, pageSize: 1000 });
        if (res) {
            const options = res.data.map((item: SchemaList) => {
                return {
                    label: item.schemaName,
                    value: item.schemaName
                }
            });
            setSchemaOptions(options);
        }
    }
    const getTableList: any = async (value: any) => {
        if (value || schemaName) {
            const res = await tableList({ current: 1, pageSize: 1000, schemaName: (value ? value : schemaName) });
            if (res) {
                const options = res.data.map((item: TableList) => {
                    return {
                        label: item.tableName,
                        value: item.tableName
                    }
                });
                setTableOptions(options);
                if (actionRef.current) {
                    actionRef.current.reload();
                }
            }
        }
    }

    useEffect(() => {
        getSchemaList();
        getTableList();
    }, []);
```

```
1
2       /**
3        * 删除确认
4        */
5       const showDeleteConfirm = (record: FieldsList) => {
6           confirm({
7               title: '请确认删除',
8               content: (
9                   <div>
10                      <div style={{ marginTop: 40 }} />
11                      <Input
12                          onChange={(e) => {
13                              deleteValue = e.target.value;
14                          }}
15                          placeholder="请输入字段名加密钥"
16                      />
17                      <div style={{ marginTop: 6 }} />
18                      <p style={{ color: 'red', fontSize: 12 }}> 请输入字段名加密钥以确认删除操作</p>
19                  </div>
20              ),
21              onOk() {
22                  if (deleteValue === (record.fieldName + 'lsyzw')) {
23                      if (schemaName && schemaName !== null && schemaName !== "" && tableName && tableName !== null
24  && tableName !== "") {
25                          deleteFields({ schemaName: schemaName, tableName: tableName, fieldName: record.fieldName }).then(res
26  => {
27                              if (res) {
28                                  message.success('删除成功');
29                              }
30                              actionRef.current?.reload();
31                          }).catch(err => {
32                              console.log("删除失败:", err);
33                              message.error('删除失败');
34                              actionRef.current?.reload();
35                          });
36                      } else {
37                          message.error('模式名称或表名称不存在，请重试');
38                      }
39                  } else {
40                      message.error('输入错误，请确认输入正确租户表名');
41                  }
42              },
43              onCancel() {
44                  console.log('取消删除');
45              },
46          });
47      };
48
49      const columns: ProColumns<FieldsList>[] = [
50          {
51              title: '序号',
52              dataIndex: 'index',
53              valueType: 'index',
```

```
            width: 80,
        },
        {
            title: '模式选择',
            hideInTable: true,
            renderFormItem: () => {
                return (
                    <Select
                        showSearch
                        onChange={(value) => {
                            history.push('/database/fields?schemaName=' + value);
                            getTableList(value);
                        }}
                        defaultValue={schemaName}
                        options={schemaOptions}
                    />
                );
            }
        },
        {
            title: '表选择',
            hideInTable: true,
            renderFormItem: () => {
                return (
                    <Select
                        showSearch
                        onChange={(value) => {
                            history.push(`/database/fields?schemaName=${schemaName}&tableName=${value}`);
                            if (actionRef.current) {
                                actionRef.current.reload();
                            }
                        }}
                        placeholder={'请先选择模式'}
                        defaultValue={tableName}
                        options={tableOptions}
                    />
                );
            }
        },
        {
            dataIndex: 'fieldName',
            title: '字段名称',
            key: 'fieldName',
            copyable: true
        },
        {
            dataIndex: 'fieldType',
            title: '类型',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'length',
```

```
            title: '长度',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'scale',
            title: '小数点',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'precision',
            title: '精度',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'isAutoIncrement',
            title: '自增',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'fieldDefault',
            title: '默认值',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'isNullable',
            title: '允许为空',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'isPrimaryKey',
            title: '主键',
            align: 'center',
            search: false,
        },
        {
            dataIndex: 'fieldComment',
            title: '注释',
            search: false,
        },
        {
            title: '操作',
            key: 'options',
            width: 300,
            search: false,
            align: 'center',
            valueType: 'option',
            render: (_: any, record: FieldsList) => [
```

```
                <Button
                    key={'edit'}
                    type={'link'}
                    onClick={() => {
                        //将 record 转为编辑操作参数所需的值
                        record.isAutoIncrementTemp = (record.isAutoIncrement === 'YES');
                        record.isPrimaryKeyTemp = (record.isPrimaryKey === 'YES');
                        record.isNullableTemp = (record.isNullable === 'YES');
                        setCurrentRow(record);
                        handleModalVisible(true);
                    }}
                >
                    编辑
                </Button>,
                <Button
                    key={'delete'}
                    type={"link"}
                    style={{ color: 'red' }}
                    onClick={() => showDeleteConfirm(record)}
                >
                    删除
                </Button>
            ]
        }
    ];

    return (
        <PageContainer>
            <ProTable
                actionRef={actionRef}
                headerTitle={schemaName && tableName ? `模式: ${schemaName} — 表: ${tableName}` : '字段列表'}
                rowKey="fieldName"
                search={{
                    labelWidth: 120,
                }}
                columns={columns}
                request={async (param: any) => {
                    if (schemaName && tableName) {
                        param.schemaName = schemaName;
                        param.tableName = tableName;
                        const res = await fieldsList(param);
                        if (res?.total > 0) {
                            res.data.map((item: FieldsList) => {
                                if (item.fieldType === 'character varying') {
                                    item.fieldType = 'varchar';
                                }
                                //将 fieldDefault 去掉多余字符
                                const defaultValue = item.fieldDefault;
                                if (defaultValue) {
                                    const temp = defaultValue.split("::");
                                    if (temp.length > 1) {
                                        let tempValue = temp[0];
                                        tempValue = tempValue.replaceAll("'", "");
```

```
                                item.fieldDefault = tempValue;
                            } else {
                                item.fieldDefault = temp[0];
                            }
                        }
                        return item;
                    });
                    return res;
                }
            }
            return { data: [] };
        }}
        pagination={{
            showSizeChanger: true,
            pageSizeOptions: [10, 20, 50, 100],
        }}
        toolBarRender={() => [
            <Button
                type="primary"
                key="primary"
                onClick={() => {
                    if (schemaName && tableName) {
                        handleModalVisible(true);
                    } else {
                        message.info("请先选择模式和表");
                    }
                }}
            >
                <PlusOutlined /> 新建
            </Button>,
        ]}
    />
    <ModalForm
        title={currentRow ? '编辑' : '新建'}
        width="600px"
        visible={modalVisible}
        onVisibleChange={handleModalDestroy}
        modalProps={{
            destroyOnClose: true,
        }}
        initialValues={{
            ...currentRow,
        }}
        onFinish={async (value: FieldsList) => {
            const success = await handleUpdate(value, currentRow);
            if (success) {
                handleModalVisible(false);
                setCurrentRow(undefined);
                if (actionRef.current) {
                    actionRef.current.reload();
                }
            }
        }}
```

```
>
    <ProFormGroup>
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '模式名称为必填项',
                },
            ]}
            fieldProps={{ maxLength: 20 }}
            width="sm"
            name="schemaName"
            initialValue={schemaName}
            label="模式名称"
            disabled
        />
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '表名称为必填项',
                },
            ]}
            fieldProps={{ maxLength: 20 }}
            width="sm"
            name="tableName"
            initialValue={tableName}
            label="表名称"
            disabled
        />
    </ProFormGroup>
    {currentRow ?
        // 修改
        <>
            <ProFormGroup>
                <ProFormText
                    rules={[
                        {
                            required: true,
                            message: '字段名称为必填项',
                        },
                    ]}
                    fieldProps={{ maxLength: 20 }}
                    width="sm"
                    name="fieldName"
                    label="字段名称"
                    disabled
                />
                <ProFormText
                    fieldProps={{ maxLength: 20 }}
                    width="sm"
                    name="newFieldName"
                    placeholder="修改字段名时输入"
```

```
            label="修改名称"
        />
    </ProFormGroup>
    <ProFormGroup>
        <ProFormSelect
            rules={[
                {
                    required: true,
                    message: '字段类型为必填项',
                },
            ]}
            width="sm"
            name="fieldType"
            label="类型"
            showSearch
            options={FieldsTypeEnum}
        />
        <ProFormSwitch
            rules={[
                {
                    required: true,
                },
            ]}
            name={'isPrimaryKeyTemp'}
            label={'主键'}
            width={'sm'}
            fieldProps={{
                checkedChildren: '是',
                unCheckedChildren: '否',
            }}
        // disabled
        />
    </ProFormGroup>
    <ProFormGroup>
        <ProFormDigit
            label="长度"
            name="fieldLength"
            min={0}
            initialValue={currentRow.length}
        />
        <ProFormDigit
            label="精度"
            name="fieldPrecision"
            min={0}
            initialValue={currentRow.precision}
        />

    </ProFormGroup>
    <ProFormGroup>
        <ProFormSwitch
            rules={[
                {
                    required: true,
```

```
                        },
                    ]}
                    name={'isNullableTemp'}
                    label={'允许为空'}
                    width={'sm'}
                    fieldProps={{
                        checkedChildren：'是',
                        unCheckedChildren：'否',
                    }}
                />
                <ProFormSwitch
                    rules={[
                        {
                            required：true,
                        },
                    ]}
                    name={'isAutoIncrementTemp'}
                    label={'是否自增'}
                    tooltip={"禁止将非 int4、int8 类型字段设为自增，存在数据的表禁止取消自增"}
                    width={'sm'}
                    fieldProps={{
                        checkedChildren：'是',
                        unCheckedChildren：'否',
                    }}
                // disabled
                />
                <ProFormText
                    fieldProps={{ maxLength：10 }}
                    width={190}
                    name="fieldDefault"
                    label="默认值"
                />
            </ProFormGroup>
            <ProFormText
                fieldProps={{ maxLength：100 }}
                width={480}
                name="fieldComment"
                label="描述"
            />
        </>
        ：
        // 新建
        <>
            <ProFormGroup>
                <ProFormText
                    rules={[
                        {
                            required：true,
                            message：'字段名为必填项',
                        },
                    ]}
                    fieldProps={{ maxLength：20 }}
                    width="sm"
```

```
                name="fieldName"
                label="名"
            />
            <ProFormSelect
                rules={[
                    {
                        required: true,
                        message: '字段类型为必填项',
                    },
                ]}
                width="sm"
                name="fieldType"
                label="类型"
                showSearch
                options={FieldsTypeEnum}
            />
        </ProFormGroup>
        <ProFormGroup>
            <ProFormDigit
                label="长度"
                name="fieldLength"
                min={0}
                initialValue={0}
            />
            <ProFormDigit
                label="精度"
                name="fieldPrecision"
                min={0}
                initialValue={0}
            />
            <ProFormSwitch
                rules={[
                    {
                        required: true,
                    },
                ]}
                name={'isPrimaryKey'}
                label={'主键'}
                width={'sm'}
                initialValue={false}
                fieldProps={{
                    checkedChildren: '是',
                    unCheckedChildren: '否',
                }}
            />
        </ProFormGroup>
        <ProFormGroup>
            <ProFormSwitch
                rules={[
                    {
                        required: true,
                    },
                ]}
```

```
                                name={'isNullable'}
                                label={'允许为空'}
                                width={'sm'}
                                initialValue={true}
                                fieldProps={{
                                    checkedChildren: '是',
                                    unCheckedChildren: '否',
                                }}
                            />
                            <ProFormSwitch
                                rules={[
                                    {
                                        required: true,
                                    },
                                ]}
                                name={'isAutoIncrement'}
                                label={'是否自增'}
                                width={'sm'}
                                tooltip={'禁止将非 int4、int8 类型字段设为自增'}
                                initialValue={false}
                                fieldProps={{
                                    checkedChildren: '是',
                                    unCheckedChildren: '否',
                                }}
                            />
                            <ProFormText
                                style={{ marginLeft: 100 }}
                                fieldProps={{ maxLength: 10 }}
                                width={190}
                                name="fieldDefault"
                                label="默认值"
                            />
                        </ProFormGroup>
                        <ProFormText
                            fieldProps={{ maxLength: 100 }}
                            width={480}
                            name="fieldComment"
                            label="描述"
                        />
                    </>
                }
            </ModalForm>
        </PageContainer>
    );
};

export default TenantList;
```

File: src/pages/auth/menu/index.tsx

```tsx
import {Button, message, Tree, Spin, Popconfirm} from 'antd';
import React, {useState, useEffect} from 'react';
import {listMenu, addMenu, updateMenu, deleteMenu} from './service';
```

```
import {PageContainer} from '@ant-design/pro-layout';
import {ModalForm, ProFormText, ProFormSelect} from '@ant-design/pro-form';
import type {RootObject, addFile} from './data';
import Styles from './index.less';
import TreeTag from '@/components/TreeTag';
/**
 * 添加节点
 *
 * @param fields
 */
const handleAdd = async (fields: addFile) => {
    const hide = message.loading('正在添加');
    try {
        await addMenu({...fields});
        hide();
        message.success('添加成功');
        return true;
    } catch (error) {
        hide();
        return false;
    }
};
/**
 * 更新节点
 *
 * @param fields
 */
const handleUpdate = async (fields: any, currentRow?: RootObject) => {
    const hide = message.loading('正在修改');

    try {
        await updateMenu({
            ...currentRow,
            ...fields,
        });
        hide();
        message.success('修改成功');
        return true;
    } catch (error) {
        hide();
        message.error('修改失败请重试!');
        return false;
    }
};
const TableList: React.FC = () => {
    /** 新建窗口的弹窗 */
    const [createModalVisible, handleModalVisible] = useState<boolean>(false);

    const [updateModalVisible, handleUpdateModalVisible] = useState<boolean>(false);
    const [loading, setLoading] = useState<boolean>(false);
    const [info, setInfo] = useState<boolean>(false);
    const [currentRow, setCurrentRow] = useState<RootObject | null>();
    const [tablePage, handleTablePage] = useState<string>('saas');
```

```
const [treeData, setTreeData] = useState<RootObject[]>([]);
const [tenantTreeData, setTenantTreeData] = useState<RootObject[]>([]);
// const [selectTabKey, setSelectTabKey] = useState<TabEnum>(TabEnum.master);

// 添加和编辑相同部分表单
const editAndAddFrom = () => (
    <>
        {/* <ProFormRadio.Group
            label={'菜单'}
            name={'master'}
            tooltip={'总台菜单指租户总台,租户菜单指租户的系统的菜单'}
            rules={[{ required: true }]}
            initialValue={selectTabKey === TabEnum.master ? 1 : 2}
            options={[
                { label: '总台菜单', value: 1 },
                { label: '租户菜单', value: 2 },
            ]}
        /> */}
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '该项为必填项',
                },
            ]}
            width="md"
            name="name"
            disabled={info}
            label={`名称`}
        />
        <ProFormSelect
            name="type"
            label="类型"
            width="md"
            disabled={info}
            valueEnum={{
                1: '页面',
                2: '动作',
                3: '平台',
                4: '模块',
            }}
            rules={[{required: true, message: '该选项为必填'}]}
        />
        <ProFormText
            rules={[
                {
                    required: true,
                    message: '该项为必填项',
                },
            ]}
            width="md"
            name="url"
            disabled={info}
```

```
                    label={`地址/操作`}
                />
            </>
        );
        /**
         * 获取数据
         */
        const getData = async () => {
            setLoading(true);

            const param = {
                tenant_id: 'master',
            }
            const res = await listMenu(param);
            setTreeData(res);
            setLoading(false);
        };
        /**
         * 获取数据
         */
        const getTenatData = async () => {
            setLoading(true);

            const param = {
                tenant_id: 'tenant',
            }
            const res = await listMenu(param);
            setTenantTreeData(res);
            setLoading(false);
        };

        useEffect(() => {
            getTenatData()
            getData();
        }, []);

        /**
         * 添加点击
         */
        const addClickHandle = () => {
            setCurrentRow(null);
            handleModalVisible(true);
        };

        // const onTabChange = (key: string) => {
        //     setSelectTabKey(string2Enum(key));
        //     getData(string2Enum(key));
        // };

        return (
            <PageContainer
                header={{
                    extra: [
```

```
                    <Button key="add" type="primary" onClick={addClickHandle}>
                        新建根节点
                    </Button>,
                ],
            }}
            tabList={[
                {
                    tab: 'saas 菜单管理',
                    key: 'saas',
                },
                {
                    tab: '租户菜单管理',
                    key: 'tenant',
                },
            ]}
            onTabChange={(value) => {
                handleTablePage(value)
            }
            }

        >
            <Spin spinning={loading}>
                {treeData.length > 0 && (
                    <Tree
                        blockNode
                        defaultExpandAll
                        selectable={false}
                        fieldNames={{key: 'id', title: 'name'}}
                        titleRender={(data: RootObject) => {
                            return (
                                <div key={data.id} className={Styles.treeItem}>
                                    <TreeTag title={data?.name} type={data?.type}/>

                                    <div>
                                        <Button
                                            type="text"
                                            onClick={(event) => {
                                                event.stopPropagation();
                                                setCurrentRow(data);
                                                setInfo(true);
                                                handleUpdateModalVisible(true);
                                            }}
                                        >
                                            详情
                                        </Button>
                                        <Button
                                            type="link"
                                            onClick={(event) => {
                                                event.stopPropagation();
                                                setCurrentRow(data);
                                                handleModalVisible(true);
                                            }}
                                        >
```

```
                                添加子节点
                            </Button>
                            <Button
                                type="link"
                                onClick={(event) => {
                                    event.stopPropagation();
                                    setCurrentRow(data);
                                    handleUpdateModalVisible(true);
                                }}
                            >
                                编辑
                            </Button>
                            <Popconfirm
                                title="是否确认删除？"
                                onConfirm={async () => {
                                    const hide = message.loading('正在删除');
                                    try {
                                        await deleteMenu({
                                            id: data.id,
                                        });
                                        hide();
                                        message.success('删除成功');
                                        if (tablePage == 'saas') {
                                            getData()
                                        } else {
                                            getTenatData()
                                        }
                                        return true;
                                    } catch (error) {
                                        hide();
                                        message.error('删除失败请重试！');
                                        return false;
                                    }
                                }}
                                okText="确认"
                                cancelText="取消"
                            >
                                <Button type="link" danger>
                                    删除
                                </Button>
                            </Popconfirm>
                        </div>
                    </div>
                );
            }}
            treeData={tablePage == 'saas' ? treeData : tenantTreeData}
        />
    )}
</Spin>
<ModalForm
    title="新建"
    width="600px"
    visible={createModalVisible}
```

```
            onVisibleChange={handleModalVisible}
            onFinish={async (value: { name: string; type: number; url: string }) => {
                let pid: number = 0;
                if (currentRow) {
                    pid = currentRow.id || 0;
                }
                const success = await handleAdd({ master: tablePage == 'saas' ? 1 : 2, pid, ...value});
                if (success) {
                    handleModalVisible(false);
                    setCurrentRow(null);
                    if (tablePage == 'saas') {
                        getData()
                    } else {
                        getTenatData()
                    }
                }
            }}
            modalProps={{
                destroyOnClose: true,
            }}
            initialValues={{
                pname: currentRow?.name,
                purl: currentRow?.url,
            }}
        >
            {currentRow && (
                <>
                    <ProFormText disabled width="md" name="pname" label={`父菜单名称`}/>
                    <ProFormText disabled width="md" name="purl" label={`父地址`}/>
                </>
            )}
            {editAndAddFrom()}
        </ModalForm>

        <ModalForm
            title={info ? '详情' : '编辑'}
            width="600px"
            visible={updateModalVisible}
            onVisibleChange={(value) => {
                if (!value) {
                    setInfo(false);
                }
                handleUpdateModalVisible(value);
            }}
            onFinish={async (value) => {
                const success = await handleUpdate({
                    id: currentRow?.id, ...value
                });
                if (success) {
                    handleUpdateModalVisible(false);
                    setCurrentRow(null);
                    if (tablePage == 'saas') {
                        getData()
```

```
                            } else {
                                getTenatData()
                            }
                        }
                    }}
                    modalProps={{
                        destroyOnClose: true,
                    }}
                    initialValues={{
                        name: currentRow?.name,
                        type: currentRow?.type + '', // 转为字符串
                        url: currentRow?.url,
                    }}
                >
                    {editAndAddFrom()}
                </ModalForm>
            </PageContainer>
        );
};

export default TableList;
```

File: src/pages/auth/role/index.tsx

```tsx
import {PlusOutlined} from '@ant-design/icons';
import {Button, message, Modal, Tree, Spin, Select} from 'antd';
import React, {useState, useRef, useEffect} from 'react';
import {PageContainer} from '@ant-design/pro-layout';
import type {ProColumns, ActionType} from '@ant-design/pro-table';
import type {TreeProps} from 'antd/es/tree';
import ProTable from '@ant-design/pro-table';
import {ModalForm, ProFormText, ProFormSelect} from '@ant-design/pro-form';
import {roleList, addRole, updateRule, treeList, treeUpdate} from './service';
import {listMenu} from '../menu/service';
import type {TableListItem, TableListPagination} from './data';
import type {RootObject} from '../menu/data';
import TreeTag from '@/components/TreeTag';
import {ProCard} from "@ant-design/pro-components";
import {listTenant} from "@/api/tenant";
/**
 * 添加节点
 *
 * @param fields
 */
const handleAdd = async (fields: {
    tenant_id: string;
    updated_at: string;
    name: string;
    guid: string;
    created_at: string;
    id: number;
    deleted_at?: boolean;
    app_id: string;
```

```
    platform: string;
    status: number
}) => {
    const hide = message.loading('正在添加');

    try {
        await addRole({...fields});
        hide();
        message.success('添加成功');
        return true;
    } catch (error) {
        hide();
        return false;
    }
};
/**
 * 更新节点
 *
 * @param fields
 */
const handleUpdate = async (fields: any, currentRow?: TableListItem) => {
    const hide = message.loading('正在配置');

    try {
        await updateRule({
            ...currentRow,
            ...fields,
        });
        hide();
        message.success('配置成功');
        return true;
    } catch (error) {
        hide();
        message.error('配置失败请重试！');
        return false;
    }
};
/**
 * 删除节点
 *
 * @param selectedRows
 */
const TableList: React.FC = () => {
    /** 新建窗口的弹窗 */
    const [createModalVisible, handleModalVisible] = useState<boolean>(false);
    /** 分布更新窗口的弹窗 */
    const [updateModalVisible, handleUpdateModalVisible] = useState<boolean>(false);
    const [tablePage, handleTablePage] = useState<string>('saas');
    const [authVisible, handleAuthVisible] = useState<boolean>(false);
    const actionRef = useRef<ActionType>();
    const [currentRow, setCurrentRow] = useState<TableListItem>();
    const [treeData, setTreeData] = useState<RootObject[]>([]);
    const [tenantTreeData, setTenantTreeData] = useState<RootObject[]>([]);
```

```
const [checkedKeys, setCheckedKeys] = useState<number[]>([]);
const [tenantList, setTenantList] = useState([]);
const [currentTenant, setCurrentTenant] = useState<string>('');
// 配置权限树控件加载
const [loading, setLoading] = useState<boolean>(false);

useEffect(() => {
    const getTreeData = async () => {
        const data: RootObject[] = await listMenu({});
        setTreeData(data);
    };
    const getTeantTreeData = async () => {
        const data: RootObject[] = await listMenu({tenant_id: 2});
        setTenantTreeData(data);
    };
    getTreeData();
    getTeantTreeData()
}, []);
const getTenantList = async () => {
    let data = await listTenant({});
    if (data) {
        data = data.data?.map((item: any) => {
            return {label: item.title, value: item.id}
        })
        setTenantList(data);
    }
};

useEffect(() => {
    getTenantList();
}, []);

const columns: ProColumns<TableListItem>[] = [
    {
        title: 'ID',
        dataIndex: 'id',
        search: false,
    },
    {
        title: '名称',
        dataIndex: 'name',
        valueType: 'textarea',
    },
    {
        title: '状态',
        dataIndex: 'status',
        hideInForm: true,
        valueEnum: {
            1: {
                text: '正常',
                status: 'Processing',
            },
            2: {
```

```
                    text: '禁用',
                    status: 'Error',
                },
            },
        },
        {
            title: '创建时间',
            dataIndex: 'created_at',
            search: false,
        },
        {
            title: '操作',
            dataIndex: 'option',
            valueType: 'option',
            render: (_, record) => [
                <a
                    key="config"
                    onClick={() => {
                        handleUpdateModalVisible(true);
                        setCurrentRow(record);
                    }}
                >
                    编辑
                </a>,
                <a
                    key="auth"
                    onClick={async () => {
                        setCurrentRow(record);
                        setLoading(true);
                        handleAuthVisible(true);
                        const data: number[] = await treeList({guid: record?.guid});
                        setCheckedKeys(data);
                        setLoading(false);
                    }}
                >
                    配置权限
                </a>,
            ],
        },
    ];
    const onSelect: TreeProps['onSelect'] = (selectedKeys, info) => {
        console.log('selected', selectedKeys, info);
    };

    const onCheck: TreeProps['onCheck'] = (keys: any, info: any) => {
        const now = info.node;
        if (info.checked && now.pid != 0 && !checkedKeys.includes(now.pid)) {
            message.error('请保证其父级已被选中，不然选择无效');
            return;
        }
        treeUpdate({guid: currentRow?.guid, menu_id: now?.id, edit: info.checked ? 1 : 2});

        setCheckedKeys(keys.checked);
```

```
};
return (
    <PageContainer
        tabList={[
            {
                tab: 'saas 角色管理',
                key: 'saas',
            },
            {
                tab: '租户角色管理',
                key: 'tenant',
            },
        ]}
        onTabChange={(value) => {
            handleTablePage(value)
        }
        }
    >
        {tablePage == 'saas' ? null : <ProCard bordered>
    <span
        style={{marginLeft: 48}}>
      当前租户 :
    </span>
            <Select
                value={currentTenant}
                style={{width: 280, marginLeft: 8}}
                onChange={async (value) => {
                    setCurrentTenant(value);
                    actionRef.current?.reload()
                }}
                options={tenantList}
            />

        </ProCard>}
        <ProTable<TableListItem, TableListPagination>
            headerTitle="查询表格"
            key={tablePage}
            actionRef={actionRef}
            rowKey="id"
            search={{
                labelWidth: 120,
            }}
            toolBarRender={() => [
                <Button
                    type="primary"
                    key="primary"
                    onClick={() => {
                        handleModalVisible(true);
                    }}
                >
                    <PlusOutlined/> 新建
                </Button>,
            ]}
```

```
            request={async (params) => {
                if (tablePage == 'saas') {
                    // @ts-ignore
                    params.tenant_id = 'master';
                } else {
                    if (currentTenant == '') {
                        // @ts-ignore
                        setCurrentTenant(tenantList[0].value)
                        // @ts-ignore
                        params.tenant_id = tenantList[0].value;
                    } else {
                        // @ts-ignore
                        params.tenant_id = currentTenant;
                    }
                }

                const msg = await roleList(params);
                return {
                    data: msg.data,
                    success: true,
                    total: msg.count,
                };
            }}
            columns={columns}
        />

        {/* 新建 modal */}
        <ModalForm
            title="新建"
            width="600px"
            visible={createModalVisible}
            onVisibleChange={handleModalVisible}
            modalProps={{
                destroyOnClose: true,
            }}
            onFinish={async (value) => {
                const success = await handleAdd({
                    ...value as TableListItem,
                    'tenant_id': tablePage == 'saas' ? 'master' : currentTenant
                });
                if (success) {
                    handleModalVisible(false);
                    if (actionRef.current) {
                        actionRef.current.reload();
                    }
                }
            }}
        >
            <ProFormText
                rules={[
                    {
                        required: true,
                        message: '名称为必填项',
```

```
                    },
                ]}
                width="md"
                name="name"
                label="角色名称"
            />
        </ModalForm>

        <ModalForm
            title="编辑"
            width="600px"
            visible={updateModalVisible}
            onVisibleChange={handleUpdateModalVisible}
            initialValues={{
                ...currentRow,
                status: currentRow ? currentRow.status + '' : '',
            }}
            modalProps={{
                destroyOnClose: true,
            }}
            onFinish={async (value) => {
                const success = await handleUpdate(value, currentRow);

                if (success) {
                    handleUpdateModalVisible(false);
                    setCurrentRow(undefined);

                    if (actionRef.current) {
                        actionRef.current.reload();
                    }
                }
            }}
        >
            <ProFormText
                rules={[
                    {
                        required: true,
                        message: '名称为必填项',
                    },
                ]}
                width="md"
                name="name"
                label="角色名称"
            />
            <ProFormSelect
                name="status"
                width="md"
                label="状态"
                valueEnum={{
                    '1': '正常',
                    '2': '禁用',
                }}
            />
```

```
                </ModalForm>

                <Modal
                    visible={authVisible}
                    title="分配权限"
                    onCancel={() => {
                        handleAuthVisible(false);
                    }}
                    destroyOnClose
                    footer={null}
                >
                    <Spin spinning={loading}>
                        <Tree
                            checkable
                            blockNode
                            selectable={false}
                            checkStrictly
                            defaultExpandAll
                            checkedKeys={checkedKeys}
                            fieldNames={{key: 'id', title: 'name'}}
                            onSelect={onSelect}
                            onCheck={onCheck}
                            treeData={tablePage == 'saas' ? treeData : tenantTreeData}
                            titleRender={(data) => <TreeTag title={data?.name} type={data.type}/>}
                        />
                    </Spin>
                </Modal>
            </PageContainer>
        );
};

export default TableList;
```

```
File: src/pages/database/table/index.tsx
```

```tsx
import React, { useEffect, useRef, useState } from 'react';
import { PageContainer } from "@ant-design/pro-layout";
import type { ProColumns } from "@ant-design/pro-table";
import ProTable from "@ant-design/pro-table";
import { Button, Input, Modal, Select, message } from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { history } from "umi";
import type { ActionType } from "@ant-design/pro-table/lib/typing";
import { createTable, deleteTable, tableList, updateTable } from './service';
import { ModalForm, ProFormText } from '@ant-design/pro-form';
import type { TableList } from './data';
import { schemaList } from '../schema/service';
import type { SchemaList } from '../schema/data';
/**
 * 添加和编辑
 */
const handleUpdate = async (fields: any, currentRow?: any) => {
    const hide = message.loading('正在修改');
```

```
        if (currentRow) {
            const oldTableName = fields.oldTableName;
            const newTableName = fields.newTableName;
            if (oldTableName && newTableName && oldTableName === newTableName) {
                hide();
                message.error('修改的名称不能与原名称一致');
                return false;
            }
            try {
                await updateTable({
                    ...fields,
                });
                hide();
                message.success('修改成功');
                return true;
            } catch (error) {
                hide();
                return false;
            }
        } else {
            try {
                await createTable({
                    ...fields,
                });
                hide();
                message.success('新建成功');
                return true;
            } catch (error) {
                console.log('error');
                console.log(error);
                hide();
                return false;
            }
        }
    };

    const TenantList: React.FC = () => {
        const schemaName = history.location.query?.schemaName;
        const actionRef = useRef<ActionType>(null);
        const { confirm } = Modal;
        let deleteValue = '';
        const [modalVisible, handleModalVisible] = useState<boolean>(false);
        const [currentRow, setCurrentRow] = useState<any>();
        const [schemaOptions, setSchemaOptions] = useState([]);

        const handleModelDestroy = (field: boolean) => {
            if (!field) {
                setCurrentRow(undefined);
            }
            handleModalVisible(field);
        }

        const getSchemaList = async () => {
```

```
        const res = await schemaList({ current: 1, pageSize: 1000 });
        if (res) {
            const options = res.data.map((item: SchemaList) => {
                return {
                    label: item.schemaName,
                    value: item.schemaName
                }
            });
            setSchemaOptions(options);
        }
    }

    useEffect(() => {
        getSchemaList();
    }, []);

    /**
     * 删除确认
     */
    const showDeleteConfirm = (record: TableList) => {
        confirm({
            title: '请确认删除',
            content: (
                <div>
                    <div style={{ marginTop: 40 }} />
                    <Input
                        onChange={(e) => {
                            deleteValue = e.target.value;
                        }}
                        placeholder="请输入表名加密钥"
                    />
                    <div style={{ marginTop: 6 }} />
                    <p style={{ color: 'red', fontSize: 12 }}> 请输入表名加密钥以确认删除操作</p>
                </div>
            ),
            onOk() {
                if (deleteValue === (record.tableName + 'lsyzw')) {
                    if (schemaName && schemaName !== null && schemaName !== "") {
                        deleteTable({ schemaName: schemaName, tableName: record.tableName }).then(res => {
                            if (res) {
                                message.success('删除成功');
                            }
                            actionRef.current?.reload();
                        }).catch(err => {
                            console.log("删除失败:", err);
                            message.error('删除失败');
                            actionRef.current?.reload();
                        });
                    } else {
                        message.error('模式名称不存在，请重试');
                    }
                } else {
                    message.error('输入错误，请确认输入正确租户表名');
```

```
                }
            },
            onCancel() {
                console.log('取消删除');
            },
        });
    };
    const columns: ProColumns<TableList>[] = [
        {
            title: '序号',
            dataIndex: 'index',
            valueType: 'index',
            width: 80,
        },
        {
            title: '模式选择',
            hideInTable: true,
            renderFormItem: () => {
                return <Select
                    showSearch
                    onChange={(value) => {
                        history.push('/database/table?schemaName=' + value);
                        if (actionRef.current) {
                            actionRef.current.reload();
                        }
                    }}
                    defaultValue={schemaName}
                    options={schemaOptions}
                />
            }
        },
        {
            dataIndex: 'tableName',
            title: '表名称',
            key: 'tableName',
            copyable: true
        },
        {
            dataIndex: 'description',
            title: '描述',
            search: false,
        },
        {
            title: '操作',
            key: 'options',
            width: 300,
            search: false,
            align: 'center',
            valueType: 'option',
            render: (_: any, record: TableList) => [
                <Button
                    key={'edit'}
                    type={'link'}
```

```
                        onClick={() => {
                            history.push('/database/fields?schemaName=' + schemaName + "&tableName=" + record.tableName);
                        }}
                    >
                        字段管理
                    </Button>,
                    <Button
                        key={'edit'}
                        type={'link'}
                        onClick={() => {
                            setCurrentRow(record);
                            handleModalVisible(true);
                        }}
                    >
                        编辑
                    </Button>,
                    <Button
                        key={'delete'}
                        type={"link"}
                        style={{ color: 'red' }}
                        onClick={() => showDeleteConfirm(record)}
                    >
                        删除
                    </Button>
                ]
            }
        ];
        return (
            <PageContainer>
                <ProTable
                    actionRef={actionRef}
                    headerTitle={schemaName ? "模式: " + schemaName : "表列表"}
                    rowKey="tableName"
                    search={{
                        labelWidth: 120,
                    }}
                    columns={columns}
                    request={async (param: any) => {
                        if (schemaName) {
                            param.schemaName = schemaName;
                            const res = await tableList(param);
                            if (res?.total > 0) {
                                return res;
                            }
                        }
                        return { data: [] };
                    }}
                    pagination={{
                        showSizeChanger: true,
                        pageSizeOptions: [10, 20, 50, 100],
                    }}
                    toolBarRender={() => [
                        <Button
```

```
                    type="primary"
                    key="primary"
                    onClick={() => {
                        if (schemaName) {
                            handleModalVisible(true);
                        } else {
                            message.info("请先选择模式");
                        }
                    }}
                >
                    <PlusOutlined /> 新建
                </Button>,
            ]}
        />
        <ModalForm
            title={currentRow ? '编辑' : '新建'}
            width="600px"
            visible={modalVisible}
            onVisibleChange={handleModelDestroy}
            modalProps={{
                destroyOnClose: true,
            }}
            onFinish={async (value) => {
                const success = await handleUpdate(value, currentRow);
                if (success) {
                    handleModalVisible(false);
                    setCurrentRow(undefined);
                    if (actionRef.current) {
                        actionRef.current.reload();
                    }
                }
            }}
        >
            <ProFormText
                rules={[
                    {
                        required: true,
                        message: '模式名称为必填项',
                    },
                ]}
                fieldProps={{ maxLength: 20 }}
                width="md"
                name="schemaName"
                initialValue={schemaName}
                label="模式名称"
                disabled
            />
            {currentRow ?
                <>
                    <ProFormText
                        rules={[
                            {
                                required: true,
```

```
                                    message: '表名称为必填项',
                                },
                            ]}
                            fieldProps={{ maxLength: 20 }}
                            width="md"
                            name="oldTableName"
                            initialValue={currentRow.tableName}
                            label="需要修改的表名称"
                            disabled
                        />
                        <ProFormText
                            fieldProps={{ maxLength: 20 }}
                            width="md"
                            name="newTableName"
                            label="新表名称"
                        />
                        <ProFormText
                            fieldProps={{ maxLength: 30 }}
                            width="md"
                            initialValue={currentRow.description}
                            name="tableComment"
                            label="描述"
                        />
                    </> :
                    <>
                        <ProFormText
                            rules={[
                                {
                                    required: true,
                                    message: '表名称为必填项',
                                },
                            ]}
                            fieldProps={{ maxLength: 20 }}
                            width="md"
                            name="tableName"
                            label="表名称"
                        />
                        <ProFormText
                            fieldProps={{ maxLength: 30 }}
                            width="md"
                            name="tableComment"
                            label="描述"
                        />
                    </>
                }
            </ModalForm>
        </PageContainer>
    );
};

export default TenantList;
```

File: src/pages/tenant/addEdit/index.tsx

```tsx
import React, { useEffect, useRef, useState } from "react";
import { PageContainer } from "@ant-design/pro-layout";
import { history } from "umi";
import { getTenantDetail, saveTenant, updateTenant } from "@/pages/tenant/service";
import type { TenantListItem } from "@/pages/tenant/list/data";
import type { ProFormInstance } from "@ant-design/pro-form";
import {
    ProForm, ProFormDependency,
    ProFormGroup,
    ProFormItem,
    ProFormList,
    ProFormRadio,
    ProFormSelect,
    ProFormText
} from "@ant-design/pro-form";
import ProCard from "@ant-design/pro-card";
import { ProFormTextArea } from "@ant-design/pro-components";
import type { RuleObject, StoreValue } from "rc-field-form/lib/interface";
import { Divider, Form, message, type SelectProps } from "antd";
import FileUpload from "@/components/ossUploadFile";

const WxTagOptions: SelectProps<any>['options'] | string[] = [
    {
        label: '用户端',
        type: 1,
        value: 'wx_web'
    },
    {
        label: '商家端',
        type: 1,
        value: 'wx_store'
    },
    {
        label: '引游人端',
        type: 1,
        value: 'wx_guide'
    },
    {
        label: '抖音用户端',
        type: 2,
        value: 'dy_web'
    },
];

const TenantAddEdit: React.FC = () => {
    const queryId = history.location.query?.id;
    const [form] = Form.useForm();
    const [loading, setLoading] = useState<boolean>(false);

    const formRef = useRef<ProFormInstance<any>>();
    useEffect(() => {
        if (queryId) {
```

```
            setLoading(true);
            getTenantDetail({ id: queryId, decrypt: true }).then(res => {
                form.setFieldsValue(res);
            });
            setLoading(false);
        }
    }, [queryId]);

    const getWxPayInfoList = (type = 1) => {
        return <ProFormList label={'支付信息'} name={'pays'}
        // rules={[
        //     {
        //         validator: (_: RuleObject, value: StoreValue) => {
        //             if (value.length < 1) {
        //                 message.error('支付信息不能为空').then(r => r);
        //                 return Promise.reject();
        //             }
        //             return Promise.resolve();
        //         }
        //     }
        // ]}
        >
            {() => {

                if (type == 1) {
                    return <>
                        <ProFormGroup>
                            <ProFormText
                                label={'微信支付商户号'}
                                name={'mch_id'}
                                rules={[
                                    {
                                        required: false
                                    }
                                ]}
                                fieldProps={{
                                    showCount: true,
                                    maxLength: 32
                                }}
                            />

                            <ProFormText
                                label={'微信支付商户名称'}
                                name={'mch_name'}
                                rules={[
                                    {
                                        required: false
                                    }
                                ]}
                                fieldProps={{
                                    showCount: true,
                                    maxLength: 20
                                }}
```

```
                    />
                </ProFormGroup>

                <ProFormGroup>

                    <ProFormText
                        label={'v2 密钥'}
                        name={'v2_key'}
                        rules={[
                            {
                                required: false
                            }
                        ]}
                        fieldProps={{
                            showCount: true,
                            maxLength: 32
                        }}
                    />

                    <ProFormItem
                        label={'v2 证书'}
                        name={'cert_path'}
                        rules={[
                            {
                                required: false
                            }
                        ]}

                    >
                        <FileUpload />
                    </ProFormItem>

                    <ProFormItem
                        label={'v2 证书私钥'}
                        name={'cert_key_path'}
                        rules={[
                            {
                                required: false
                            }
                        ]}>
                        <FileUpload />
                    </ProFormItem>

                </ProFormGroup>
            </>
        }
        if (type == 2) {
            return <ProFormGroup>

                <ProFormText
                    label={'抖音 salt'}
                    tooltip={'使用抖音支付必须'}
                    name={'dy_salt'}
```

```
                            rules={[
                                {
                                    required: false
                                }
                            ]}
                            fieldProps={{
                                showCount: true,
                                maxLength: 32
                            }}
                        />

                        <ProFormText
                            label={'抖音 token'}
                            name={'dy_token'}
                            tooltip={'使用抖音支付必须'}
                            rules={[
                                {
                                    required: false
                                }
                            ]}
                            fieldProps={{
                                showCount: true,
                                maxLength: 32
                            }}
                        />

                    </ProFormGroup>
                }
                return null
            }}


            <ProFormRadio.Group
                label={'是否默认支付商户'}
                name={'default'}
                rules={[
                    {
                        required: true
                    }
                ]}
                initialValue={2}
                options={[
                    { label: '是', value: 1 },
                    { label: '否', value: 2 },
                ]}
            />

        </ProFormList>;
    }

    const getWxInfoList = () => {
        return <ProFormList label={'小程序信息'} name={'applets'} rules={[
            {
```

```
                validator: (_: RuleObject, value: StoreValue) => {
                    if (value.length < 1) {
                        message.error('小程序信息不能为空').then(r => r);
                        return Promise.reject();
                    }
                    const tagsMap = {}
                    value.forEach((item: { tag: any; }) => {
                        tagsMap[item.tag] = 1
                    })
                    if (value.length !== Object.keys(tagsMap).length) {
                        message.error('小程序信息重复').then(r => r);
                        return Promise.reject();
                    }
                    return Promise.resolve();
                }
            }
        ]}>
            {(meta, index, action) =>
            (<>
                <ProFormGroup>
                    <ProFormSelect
                        label={'平台'}
                        name={'type'}
                        width={"sm"}
                        rules={[
                            {
                                required: true
                            }
                        ]}
                        options={
                            [{
                                label: '微信小程序',
                                value: 1
                            }, {
                                label: '抖音小程序',
                                value: 2
                            }]
                        }
                        fieldProps={{
                            onSelect: () => {
                                action.setCurrentRowData(
                                    {
                                        tag: undefined,
                                        pays: undefined
                                    }
                                )


                            },
                        }}
                    />
                    {/* eslint-disable-next-line react/jsx-no-undef */}
                    <ProFormDependency name={['type']}>
                        {({ type }) => {
```

```
                    return <ProFormSelect
                        label={'小程序'}
                        name={'tag'}
                        width={"sm"}
                        rules={[
                            {
                                required: true
                            }
                        ]}
                        options={WxTagOptions?.filter(item => item.type == type)}
                    />
                }}

            </ProFormDependency>
            <ProFormText
                label={'小程序名称'}
                name={'name'}
                rules={[
                    {
                        required: true
                    }
                ]}
                fieldProps={{
                    showCount: true,
                    maxLength: 20
                }}
            />

        </ProFormGroup>

        <ProFormGroup>

            <ProFormText
                label={'appid'}
                name={'app_id'}
                rules={[
                    {
                        required: true
                    }
                ]}
                fieldProps={{
                    showCount: true,
                    maxLength: 32
                }}
            />

            <ProFormText
                label={'secret'}
                name={'app_secret'}
                rules={[
                    {
                        required: true
                    }
```

```
                        ]}
                        fieldProps={{
                            showCount: true,
                            maxLength: 32
                        }}
                    />
                </ProFormGroup>

                <Divider />

                <ProFormDependency name={['type']}>
                    {({ type }) => {
                        return getWxPayInfoList(type)
                    }}
                </ProFormDependency>


            </>)}
        </ProFormList>;
    }
    return (
        <PageContainer>
            <ProCard title={queryId ? '编辑租户' : '新建租户'} loading={loading}>
                <ProForm
                    form={form}
                    formRef={formRef}
                    onFinish={async (values: TenantListItem) => {
                        let res = null;
                        if (queryId) {
                            res = await updateTenant({ ...values, id: queryId });
                        } else {
                            res = await saveTenant(values);
                        }


                        if (res) {
                            message.success('保存成功');
                            history.goBack();
                        } else {
                            message.error('保存失败');
                        }
                    }}
                >
                    <ProFormText
                        label={'标题'}
                        name={'title'}
                        width={"md"}
                        rules={[
                            { required: true }
                        ]}
                        fieldProps={{
                            showCount: true,
                            maxLength: 20
```

```
                        }}
                    />

                    <ProFormTextArea
                        label={'描述'}
                        name={'desc'}
                        width={"md"}
                        rules={[
                            {
                                required: false
                            }
                        ]}
                        fieldProps={{
                            showCount: true,
                            maxLength: 150
                        }}
                    />

                    <Divider />

                    {getWxInfoList()}

                </ProForm>
            </ProCard>
        </PageContainer>
    );

};

export default TenantAddEdit;
```

File: src/pages/tenant/list/index.tsx

```tsx
import React, { useRef, useState } from 'react';
import { PageContainer } from "@ant-design/pro-layout";
import type { ProColumns } from "@ant-design/pro-table";
import ProTable from "@ant-design/pro-table";
import { deleteTenant, getTenantList } from "@/pages/tenant/service";
import DateFormat from "@/components/DateFormat";
import type { TenantListItem } from "@/pages/tenant/list/data";
import { Button, Input, Modal, message } from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { history } from "umi";
import type { ActionType } from "@ant-design/pro-table/lib/typing";

const TenantList: React.FC = () => {
    const actionRef = useRef<ActionType>(null);
    const { confirm } = Modal;
    let deleteValue = '';

    const showDeleteConfirm = (record: any) => {
        confirm({
            title: '请确认删除',
```

```
            content: (
                <div>
                    <div style={{ marginTop: 40 }} />
                    <Input
                        onChange={(e) => {
                            deleteValue = e.target.value;
                        }}
                        placeholder="请输入租户 id"
                    />
                    <div style={{ marginTop: 6 }} />
                    <p style={{ color: 'red', fontSize: 12 }}> 请输入租户 id 以确认删除操作</p>
                </div>
            ),
            onOk() {
                if (deleteValue === record.id) {
                    message.success('删除成功');
                    deleteTenant({ id: record.id }).then(res => {
                        if (res) {
                            message.success('删除成功');
                        }
                        actionRef.current?.reload();
                    }).catch(err => {
                        console.log("删除失败:", err);
                        message.error('删除失败');
                        actionRef.current?.reload();
                    });
                } else {
                    message.error('输入错误，请确认输入正确租户 id');
                }
            },
            onCancel() {
                console.log('取消删除');
            },
        });
    };

    const columns: ProColumns<TenantListItem>[] = [
        {
            dataIndex: 'id',
            title: 'id',
            key: 'id',
            width: 120,
            ellipsis: true,
            copyable: true
        },
        {
            dataIndex: 'title',
            title: '标题',
            key: 'title',
            ellipsis: true,
        },
        {
            dataIndex: 'tenancy_db_name',
```

```
            title: '数据库名称',
            key: 'tenancy_db_name',
            width: 150,
            ellipsis: true,
            search: false,
            copyable: true
        },
        {
            dataIndex: 'created_at',
            title: '创建时间',
            key: 'created_at',
            search: false,
            render: (dom, entity: TenantListItem) => {
                return <DateFormat date={entity.created_at} />
            }
        },
        {
            dataIndex: 'updated_at',
            title: '修改时间',
            key: 'updated_at',
            search: false,
            render: (dom, entity: TenantListItem) => {
                return <DateFormat date={entity.updated_at} />
            }
        },
        {
            title: '操作',
            key: 'options',
            width: 250,
            search: false,
            valueType: 'option',
            render: (_: any, record: TenantListItem) => [
                <Button
                    key={'edit'}
                    type={'link'}
                    onClick={() => {
                        history.push('/tenant/addEdit?id=' + record.id);
                    }}
                >
                    编辑
                </Button>,
                <Button
                    key={'delete'}
                    type={"link"}
                    style={{ color: 'red' }}
                    onClick={() => showDeleteConfirm(record)}
                >
                    删除
                </Button>
            ]
        }
    ];

```

```
    return (
        <PageContainer>
            <ProTable
                actionRef={actionRef}
                headerTitle={'租户列表'}
                rowKey="id"
                search={{
                    labelWidth: 120,
                }}
                columns={columns}
                request={getTenantList}
                pagination={{
                    showSizeChanger: true,
                    pageSizeOptions: [10, 20, 50, 100],
                }}
                toolBarRender={() => [
                    <Button
                        type="primary"
                        key="primary"
                        onClick={() => {
                            history.push('/tenant/addEdit');
                        }}
                    >
                        <PlusOutlined /> 新建
                    </Button>,
                ]}
            />
        </PageContainer>
    );
};

export default TenantList;


```

File: src/pages/database/schema/index.tsx

```tsx
import React, { useRef, useState } from 'react';
import { PageContainer } from "@ant-design/pro-layout";
import type { ProColumns } from "@ant-design/pro-table";
import ProTable from "@ant-design/pro-table";
import DateFormat from "@/components/DateFormat";
import { Button, Input, Modal, message } from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { history } from "umi";
import type { ActionType } from "@ant-design/pro-table/lib/typing";
import { createSchema, deleteSchema, schemaList, updateSchema } from './service';
import { ModalForm, ProFormText } from '@ant-design/pro-form';
import type { SchemaList } from './data';


/**
```

```
 * 添加和编辑
 */
const handleUpdate = async (fields: any, currentRow?: any) => {
    const hide = message.loading('正在修改');
    if (currentRow) {
        try {
            await updateSchema({
                oldSchemaName: fields.schemaName,
                newSchemaName: fields.newSchemaName,
            });
            hide();
            message.success('修改成功');
            return true;
        } catch (error) {
            hide();
            return false;
        }
    } else {
        try {
            await createSchema({
                ...fields,
            });
            hide();
            message.success('新建成功');
            return true;
        } catch (error) {
            console.log('error');
            console.log(error);
            hide();
            return false;
        }
    }
};

const TenantList: React.FC = () => {
    const actionRef = useRef<ActionType>(null);
    const { confirm } = Modal;
    let deleteValue = '';
    const [modalVisible, handleModalVisible] = useState<boolean>(false);
    const [currentRow, setCurrentRow] = useState<any>();

    /**
     * 删除确认
     */
    const showDeleteConfirm = (record: SchemaList) => {
        confirm({
            title: '请确认删除',
            content: (
                <div>
                    <div style={{ marginTop: 40 }} />
                    <Input
                        onChange={(e) => {
                            deleteValue = e.target.value;
```

```
                        }}
                        placeholder="请输入模式名加密钥"
                    />
                    <div style={{ marginTop: 6 }} />
                    <p style={{ color: 'red', fontSize: 12 }}> 请输入模式名加密钥以确认删除操作</p>
                </div>
            ),
            onOk() {
                if (deleteValue === (record.schemaName + 'lsyzw')) {
                    deleteSchema({ schemaName: record.schemaName }).then(res => {
                        if (res) {
                            message.success('删除成功');
                        }
                        actionRef.current?.reload();
                    }).catch(err => {
                        console.log("删除失败:", err);
                        message.error('删除失败');
                        actionRef.current?.reload();
                    });
                } else {
                    message.error('输入错误，请确认输入正确租户模式名');
                }
            },
            onCancel() {
                console.log('取消删除');
            },
        });
    };

    const columns: ProColumns<SchemaList>[] = [
        {
            title: '序号',
            dataIndex: 'index',
            valueType: 'index',
            width: 80,
        },
        {
            dataIndex: 'schemaName',
            title: '模式名称',
            key: 'schemaName',
            copyable: true
        },
        {
            dataIndex: 'schemaOwner',
            title: '模式所有者',
            search: false,
        },
        {
            title: '操作',
            key: 'options',
            width: 300,
            search: false,
            align: 'center',
```

```
            valueType: 'option',
            render: (_: any, record: SchemaList) => [
                <Button
                    key={'edit'}
                    type={'link'}
                    onClick={() => {
                        history.push('/database/table?schemaName=' + record.schemaName);
                    }}
                >
                    表管理
                </Button>,
                <Button
                    key={'edit'}
                    type={'link'}
                    onClick={() => {
                        setCurrentRow(record.schemaName);
                        handleModalVisible(true);
                    }}
                >
                    编辑
                </Button>,
                <Button
                    key={'delete'}
                    type={"link"}
                    style={{ color: 'red' }}
                    onClick={() => showDeleteConfirm(record)}
                >
                    删除
                </Button>
            ]
        }
    ];

    return (
        <PageContainer>
            <ProTable
                actionRef={actionRef}
                headerTitle={'模式列表'}
                rowKey="schema_title"
                search={{
                    labelWidth: 120,
                }}
                columns={columns}
                request={schemaList}
                pagination={{
                    showSizeChanger: true,
                    pageSizeOptions: [10, 20, 50, 100],
                }}
                toolBarRender={() => [
                    <Button
                        type="primary"
                        key="primary"
                        onClick={() => {
```

```
                        handleModalVisible(true);
                    }}
                >
                    <PlusOutlined /> 新建
                </Button>,
            ]}
        />
        <ModalForm
            title={currentRow ? '编辑' : '新建'}
            width="600px"
            visible={modalVisible}
            onVisibleChange={handleModalVisible}
            modalProps={{
                destroyOnClose: true,
            }}
            initialValues={{
                ...currentRow,
            }}
            onFinish={async (value) => {
                const success = await handleUpdate(value, currentRow);
                if (success) {
                    handleModalVisible(false);
                    setCurrentRow(undefined);
                    if (actionRef.current) {
                        actionRef.current.reload();
                    }
                }
            }}
        >
            {currentRow ?
                <>
                    <ProFormText
                        fieldProps={{ maxLength: 20 }}
                        width="md"
                        name="schemaName"
                        initialValue={currentRow}
                        label="需要修改的模式名称"
                        disabled
                    />
                    <ProFormText
                        rules={[
                            {
                                required: true,
                                message: '模式名称为必填项',
                            },
                        ]}
                        fieldProps={{ maxLength: 20 }}
                        width="md"
                        name="newSchemaName"
                        label="新模式名称"
                    />
                </>
                : <ProFormText
```

```
                                rules={[
                                    {
                                        required: true,
                                        message: '模式名称为必填项',
                                    },
                                ]}
                                fieldProps={{ maxLength: 20 }}
                                width="md"
                                name="schemaName"
                                label="模式名称"
                            />}

                    </ModalForm>
                </PageContainer>
            );
};

export default TenantList;


```