link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=130 sentences=543, words=3371

1.Ant Design Pro

- Ant Design Pro 是一个企业级中后台前端/设计解决方案，我们秉承 Ant Design 的设计价值观，致力于在设计规范和基础组件的基础上，继续向上构建，提炼出典型模板/业务组件/配套设计资源，进一步提升企业级中后台产品设计研发过程中的『用户』和『设计者』的体验。
- pro.ant.design (https://pro.ant.design)
- beta-pro.ant.design (https://beta-pro.ant.design/index-cn/)
- procomponents.ant.design (https://procomponents.ant.design/components/table/)
- getting-started-cn (https://beta-pro.ant.design/docs/getting-started-cn)

1.1 启动项目

1.1.1 安装

- 新建一个空的文件夹作为项目目录，并在目录下执行
- python-380 (https://www.python.org/downloads/release/python-380/)

```
yarn create umi
```

1.1.2 目录结构

- 我们已经为你生成了一个完整的开发框架，提供了涵盖中后台开发的各类功能和坑位，下面是整个项目的目录结构。

```
├─config # umi 配置，包含路由，构建等配置
├─mock   # 本地模拟数据
├─public
│  └─icons
├─src
│  ├─components # 业务通用组件
│  │  ├─Footer
│  │  ├─HeaderDropdown
│  │  ├─HeaderSearch
│  │  ├─NoticeIcon
│  │  └─RightContent
│  ├─e2e       # 集成测试用例
│  ├─locales   # 国际化资源
│  │  ├─en-US
│  │  ├─id-ID
│  │  ├─pt-BR
│  │  ├─zh-CN
│  │  └─zh-TW
│  ├─pages     # 业务页面入口和常用模板
│  │  ├─ListTableList
│  │  │  └─components
│  │  └─user
│  │      └─login
│  ├─services # 后台接口服务
│  └─utils    # 工具库
└─tests       # 测试工具
```

1.1.3 本地开发

```
npm install
npm run dev
```

2. 用户登录

- plugin-access (https://umijs.org/zh-CN/plugins/plugin-access)

2.2.1 config\proxy.ts

config\proxy.ts

```
export default {
  dev: {
    '/api/': {
+      target: 'http://localhost:3000/',
      changeOrigin: true,
      pathRewrite: { '^': '' },
    },
  },
```

2.2.2 src\services\API.d.ts

src\services\API.d.ts

```
  export interface LoginStateType {
    status?: 'ok' | 'error';
    type?: string;
+    token?:string;
  }
```

2.2.3 login\index.tsx

src\pages\user\login\index.tsx

```
  const handleSubmit = async (values: LoginParamsType) => {
    setSubmitting(true);
    try {
      // 登录
      const msg = await fakeAccountLogin({ ...values, type });
+      if (msg.status === 'ok' && msg.token) {
+        localStorage.setItem('token',msg.token);
        message.success('登录成功！');
        await fetchUserInfo();
        goto();
        return;
      }
      // 如果失败去设置用户错误信息
      setUserLoginState(msg);
    } catch (error) {
      message.error('登录失败，请重试！');
    }
    setSubmitting(false);
  };
```

2.2.4 app.tsx

src\app.tsx

```
export const request: RequestConfig = {
  errorHandler,
+   headers: {
+    Authorization: `Bearer ${localStorage.getItem('token')}`,
+  },
};
```

2.2.5 user.ts

src\services\user.ts

```
export async function queryCurrent() {
  return request('/api/currentUser',
+  {
+    headers: {
+      Authorization: `Bearer ${localStorage.getItem('token')}`,
+    },
+  }
 );
}
```

2.2.6 API.d.ts

src\services\API.d.ts

```
  export interface CurrentUser {
    avatar?: string;
+    username?: string;
    title?: string;
    group?: string;
    signature?: string;
    tags?: {
      key: string;
      label: string;
    }[];
    userid?: string;
    access?: 'user' | 'guest' | 'admin';
    unreadCount?: number;
  }
```

2.2.7 AvatarDropdown.tsx

src\components\RightContent\AvatarDropdown.tsx

```
/**
 * 退出登录，并且将当前的 url 保存
 */
const loginOut = async () => {
  await outLogin();
+ localStorage.removeItem('token');
  const { query, pathname } = history.location;
  const { redirect } = query;
  // Note: There may be security issues, please note
  if (window.location.pathname !== '/user/login' && !redirect) {
    history.replace({
      pathname: '/user/login',
      search: stringify({
        redirect: pathname,
      }),
    });
  }
};

const AvatarDropdown: React.FC = ({ menu }) => {
  const { initialState, setInitialState } = useModel('@@initialState');
  const { currentUser } = initialState;
+  if (!currentUser || !currentUser.username) {
    return loading;
  }
  return (

+        {currentUser.username}

  );
};

export default AvatarDropdown;
```

3. 新增用户和用户列表

3.1 config\routes.ts

config\routes.ts

```
  {
    path: '/admin',
    name: 'admin',
    icon: 'crown',
    access: 'canAdmin',
    routes: [
      {
+       path: '/admin/user',
+       name: 'user',
+       icon: 'smile',
+       component: './admin/User',
      },
    ],
  },
```

src\locales\zh-CN\menu.ts

```
export default {
  'menu.welcome': '欢迎',
  'menu.more-blocks': '更多区块',
  'menu.home': '首页',
  'menu.admin': '管理页',
+ 'menu.admin.user': '用户管理',
```

3.3 data.d.ts

src\pages\admin\user\data.d.ts

```
export interface TableListItem {
+  id?:string;
+  username?:string;
+  email?:string;
+  password?: string;
+  avatar?: string;
+  updatedAt?: Date;
+  createdAt?: Date;
}

export interface TableListParams {
+  id?:string;
+  username?:string;
+  password?: string;
+  email?:string;
+  avatar?: string;
+  updatedAt?: Date;
+  createdAt?: Date;
  pageSize?: number;
  currentPage?: number;
  filter?: { [key: string]: any[] };
  sorter?: { [key: string]: any };
}
```

3.4 service.ts

src\pages\admin\user\service.ts

```
+export async function removeUser(params: { key: string[] }) {
  return request('/api/user', {
+   method: 'DELETE',
    data: {
      ...params
    },
  });
}

+export async function addUser(params: TableListItem) {
  return request('/api/user', {
    method: 'POST',
    data: {
      ...params
    },
  });
}

+export async function updateUser(params: TableListParams) {
  return request('/api/user', {
+    method: 'PUT',
    data: {
      ...params
    },
  });
}
```

3.5 user\index.tsx

src\pages\admin\user\index.tsx

```
import { PlusOutlined } from '@ant-design/icons';
+import { Button, message, Drawer ,Avatar} from 'antd';
import React, { useState, useRef } from 'react';
import { PageContainer, FooterToolbar } from '@ant-design/pro-layout';
import ProTable, { ProColumns, ActionType } from '@ant-design/pro-table';
import ProDescriptions from '@ant-design/pro-descriptions';
import CreateForm from './components/CreateForm';
import UpdateForm, { FormValueType } from './components/UpdateForm';
import { TableListItem } from './data.d';
+import { queryUser, updateUser, addUser, removeUser } from './service';
+import moment from 'moment';
/**
 * 添加节点
 * @param fields
```

```
 */
const handleAdd = async (fields: TableListItem) => {
  const hide = message.loading('正在添加');
  try {
+    await addUser({ ...fields });
    hide();
    message.success('添加成功');
    return true;
  } catch (error) {
    hide();
    message.error('添加失败请重试！');
    return false;
  }
};

/**
 * 更新节点
 * @param fields
 */
+const handleUpdate = async (id:string,fields: FormValueType) => {
  const hide = message.loading('正在配置');
  try {
    await updateUser({
+      id,
+      username: fields.username,
+      password: fields.password,
+      email: fields.email
    });
    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 handleRemove = async (selectedRows: TableListItem[]) => {
  const hide = message.loading('正在删除');
  if (!selectedRows) return true;
  try {
+    await removeUser({
      key: selectedRows.map((row:TableListItem) => row.id!),
    });
    hide();
    message.success('删除成功，即将刷新');
    return true;
  } catch (error) {
    hide();
    message.error('删除失败，请重试');
    return false;
  }
};

const TableList: React.FC = () => {
  const [createModalVisible, handleModalVisible] = useState(false);
  const [updateModalVisible, handleUpdateModalVisible] = useState(false);
+  const [stepFormValues, setStepFormValues] = useState({});
  const actionRef = useRef();
  const [row, setRow] = useState();
  const [selectedRowsState, setSelectedRows] = useState([]);
+  const columns: ProColumns[] = [
+    {
+      title: '用户名',
+      dataIndex: 'username',
+      tip: '用户名是唯一的',
+      render: (dom, entity) => {
+        return  setRow(entity)}>{dom};
+      }
+    },
+    {
+      title: '密码',
+      dataIndex: 'password',
+      hideInDescriptions:true,
+      tip: '密码',
+      hideInTable:true
+    },
+    {
+      title: '角色',
+      dataIndex: 'access',
+      search: false,
+      filters: [
+        { text: '普通用户', value: 'user' },
+        { text: '管理员', value: 'admin' },
+      ],
+      valueEnum: {
+        'user': { text: '普通用户'},
+        'admin': { text: '管理员'},
+      }
+    },
+    {
+      title: '邮箱',
+      dataIndex: 'email',
+      hideInTable:true
+    },
+    {
+      title: '头像',
+      dataIndex: 'avatar',
```

```
+      search: false,
+      hideInForm: true,
+      render: (dom,entity) => {
+        return
+      }
+    },
+    {
+      title: '更新时间',
+      dataIndex: 'updatedAt',
+      sorter: true,
+      hideInForm: true,
+      search: false,
+      renderText: (val: string) => {
+        if(!val) return '';
+        return  moment(val).fromNow();
+      }
+    },
+    {
+      title: '创建时间',
+      dataIndex: 'createdAt',
+      sorter: true,
+      hideInForm: true,
+      valueType: 'dateTime'
+    },
+    {
+      title: '操作',
+      dataIndex: 'option',
+      valueType: 'option',
+      render: (_, record) => (
+        <>
+
+            onClick={() => {
+              handleUpdateModalVisible(true);
+              setStepFormValues(record);
+            }}
+          >
+            修改
+
+        </>
+      ),
+    },
+  ];

  return (

       headerTitle={'用户管理'}
       actionRef={actionRef}
+       rowKey="id"
       search={{
         labelWidth: 120,
       }}
+       pagination={{defaultPageSize:5}}
       toolBarRender={() => [
          handleModalVisible(true)}>
            新建
          ,
       ]}
       request={(params, sorter, filter) => queryUser({ ...params, sorter, filter })}
       columns={columns}
       rowSelection={{
         onChange: (_, selectedRows) => setSelectedRows(selectedRows),
       }}
     />
     {selectedRowsState?.length > 0 && (

             已选择{' '}
             {selectedRowsState.length}{' '}
             项

         }
       >
          {
             await handleRemove(selectedRowsState);
             setSelectedRows([]);
             actionRef.current?.reloadAndRest?.();
           }}
         >
           批量删除

     )}
      handleModalVisible(false)} modalVisible={createModalVisible}>

         onSubmit={async (value) => {
           const success = await handleAdd(value);
           if (success) {
             handleModalVisible(false);
             if (actionRef.current) {
               actionRef.current.reload();
             }
           }
         }}
+        rowKey="id"
         type="form"
         columns={columns}
       />

     {stepFormValues && Object.keys(stepFormValues).length ? (
       {
+          const success = await handleUpdate(stepFormValues.id!,value);
           if (success) {
             handleUpdateModalVisible(false);
             setStepFormValues({});
             if (actionRef.current) {
               actionRef.current.reload();
```

```
              }
            }
          }}
          onCancel={() => {
            handleUpdateModalVisible(false);
            setStepFormValues({});
          }}
          updateModalVisible={updateModalVisible}
          values={stepFormValues}
        />
      ) : null}

       {
          setRow(undefined);
        }}
        closable={false}
      >
        {row?.username && (

            column={2}
+            title={row?.username}
            request={async () => ({
              data: row || {},
            })}
            params={{
+              id: row?.username,
            }}
            columns={columns}
          />
        )}

  );
};

export default TableList;
```

3.6 pages.ts

src\\locales\zh-CN\pages.ts

```
  'pages.admin.subPage.title': ' 这个页面只有 admin 权限才能查看',
+ 'pages.admin.user.createForm.newUser': '增加用户',
```

3.7 CreateForm.tsx

src\pages\admin\user\components\CreateForm.tsx

```
const CreateForm: React.FC = (props) => {
  const { modalVisible, onCancel } = props;
  const intl = useIntl();
  return (
    +        id: 'pages.searchTable.createForm.newUser',
+        defaultMessage: '增加用户',
      })}
      visible={modalVisible}
      onCancel={() => onCancel()}
      footer={null}
    >
      {props.children}

  );
};
```

4. 修改用户

4.1 UpdateForm.tsx

src\pages\admin\user\components\UpdateForm.tsx

```
import React from 'react';
import { Modal } from 'antd';
+import ProForm,{ProFormText} from '@ant-design/pro-form';
import { TableListItem } from '../data.d';

+export interface FormValueType extends Partial {}

export interface UpdateFormProps {
  onCancel: (flag?: boolean, formVals?: FormValueType) => void;
  onSubmit: (values: FormValueType) => Promise;
  updateModalVisible: boolean;
  values: Partial;
}

const UpdateForm: React.FC = (props) => {
  return (
+
+    destroyOnClose
+    title="更新用户"
+    width={420}
+    visible={props.updateModalVisible}
+    onCancel={() => props.onCancel()}
+    footer={null}
+  >
+
+      initialValues={props.values}
+      onFinish={async (values: any) => {
+        props.onSubmit(values);
+      }}
+    >
+
+        label="用户名"
+        name="username"
+      />
+
+        label="密码"
+        name="password"
+      />
+
+        label="邮箱"
+        name="email"
+      />
+
+
  );
};

export default UpdateForm;
```

1.初始化项目

- 下载mongodb (https://www.mongodb.com/try/download/community)

1.1 安装

```
cnpm i express body-parser mongoose jsonwebtoken http-status-codes -S
```

2.用户注册登录

2.1 app.js

```
let express = require("express");
let bodyParser = require("body-parser");
let user = require("./routes/user");
let app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use("/api", user);
app.listen(3000, () => {
    console.log('服务器在3000端口启动!');
});
```

2.2 config.js

config.js

```
module.exports = {
    secret: 'pro5',
    dbUrl: "mongodb://localhost:27017/pro5"
}
```

2.3 model.js

model.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
let config = require('./config');
const conn = mongoose.createConnection(config.dbUrl, { useNewUrlParser: true, useUnifiedTopology: true });

let UserSchema = new Schema({
    email: { type: String },
    username: { type: String },
    password: { type: String, required: true },
    avatar: { type: String, required: true },
    access: { type: String, required: true,default:'user' }
},{timestamps:true,toJSON:{
    transform(doc,ret){
        ret.id=ret._id;
        delete ret._id;
        delete ret.__v;
        delete ret.password;
        return ret;
    }
}});

const UserModel = conn.model('User', UserSchema);

module.exports = {
    UserModel
}
```

2.4 routes\user.js

routes\user.js

```
let express=require('express');
let {UserModel} = require('../model');
let jwt = require('jsonwebtoken');
let config = require('../config');
const router = express.Router();
router.post('/register', async (req, res) => {
    let user = req.body;
    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
    user = await UserModel.create(user);
    res.send({ status: 'ok',access: user.toJSON().access});
});
router.get('/user', async (req, res) => {
    let users = await UserModel.find();
    let dataSource = users.map(item=>item.toJSON());
    res.send({ success: true,data:dataSource});
});

router.post('/login/account', async (req, res) => {
    let query = {username:req.body.username,password:req.body.password};
    let dbUser = await UserModel.findOne(query);
    if (dbUser) {
        let user = dbUser.toJSON();
        let token = jwt.sign(user, config.secret,{expiresIn:'5d'});
        return res.send({ status: 'ok', token, type: 'account', access: user.access });
    } else {
        return res.send({
            status: 'error',
            type: 'account',
            access: 'guest'
        });
    }
});

router.get('/currentUser', async (req, res) => {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            let user = jwt.verify(authorization.split(' ')[1], config.secret);
            res.json(user);
        } catch (err) {
            res.status(401).send({});
        }
    } else {
        res.status(401).send({});
    }
});
router.get('/login/outLogin', async (req, res) => {
    res.send({ data: {}, success: true });
});
module.exports = router;
```

3.增加用户

3.1 routes\user.js

routes\user.js

```
let express=require('express');
let {UserModel} = require('../model');
let jwt = require('jsonwebtoken');
let config = require('../config');
const router = express.Router();
+//增加
+router.post('/user', async (req, res) => {
+    let user = req.body;
+    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
+    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
+    user = await UserModel.create(user);
+    res.send({ status: 'ok',access: user.toJSON().access});
+});
+//删除
+router.delete('/user',async (req, res) => {
+    console.log('req.body',req.body);
+    await UserModel.remove({_id:req.body.key});
+    return res.json({success:true});
+});
+//修改
+router.put('/user', async (req, res) => {
+    let user = req.body;
+    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
+    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
+    user = await UserModel.findByIdAndUpdate(user.id,user,{useFindAndModify:false});
+    res.send({ status: 'ok',access: user.toJSON().access});
+});
+//查询
+router.get('/user',async (req, res) => {
+    let { current = 1, pageSize = 10,sorter,filter={},...query} = req.query;
+    if(sorter){
+        sorter = sorter?JSON.parse(sorter):{};
+        for(let key in sorter){
+            sorter[key]= sorter[key]==='ascend'?1:-1;
+        }
+    }
+    if(filter){
+        filter = filter?JSON.parse(filter):{};
+        for(let key in filter){
+            if(filter[key])
+             query[key]=filter[key];
+        }
+    }
+    let total = await UserModel.countDocuments(query);
+    let users = await UserModel.find(query)
+    .sort(sorter).skip((current-1)*pageSize).limit(pageSize);
+    let dataSource = users.map(item=>item.toJSON());
+    const result = {
+        data: dataSource,
+        total,
+        success: true,
+        pageSize,
+        current,
+    };
+    return res.json(result);
+});

router.post('/register', async (req, res) => {
    let user = req.body;
    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
    user = await UserModel.create(user);
    res.send({ status: 'ok',access: user.toJSON().access});
});

router.post('/login/account', async (req, res) => {
    let query = {username:req.body.username,password:req.body.password};
    let dbUser = await UserModel.findOne(query);
    if (dbUser) {
        let user = dbUser.toJSON();
        let token = jwt.sign(user, config.secret,{expiresIn:'5d'});
        return res.send({ status: 'ok', token, type: 'account', access: user.access });
    } else {
        return res.send({
            status: 'error',
            type: 'account',
            access: 'guest'
        });
    }
});

router.get('/currentUser', async (req, res) => {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            let user = jwt.verify(authorization.split(' ')[1], config.secret);
            res.json(user);
        } catch (err) {
            res.status(401).send({});
        }
    } else {
        res.status(401).send({});
    }
});
router.get('/login/outLogin', async (req, res) => {
    res.send({ data: {}, success: true });
});
module.exports = router;
```

4.权限中间件

4.1 checkLogin.js

checkLogin.js

```
let config = require('./config');
let jwt = require('jsonwebtoken');
let { UNAUTHORIZED } = require( "http-status-codes");
const checkLogin = async (req,res,next) => {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            console.log(authorization);
            let user = jwt.verify(authorization.split(' ')[1], config.secret);
            console.log(user);
            req.user = user;
            return next();
        } catch (err) {
            return res.status(UNAUTHORIZED).send({});
        }
    } else {
        return res.status(UNAUTHORIZED).send({});
    }
};
module.exports = checkLogin;
```

4.2 checkPermission.js

checkPermission.js

```
let config = require('./config');
let jwt = require('jsonwebtoken');
let { FORBIDDEN } = require( "http-status-codes");
const checkPermission = (...allowed)=>{
  return async (req,res,next) => {
    console.log(req.user,allowed);
    if(req.user && allowed.indexOf(req.user.access)!=-1){
        next();
    }else{
        return res.status(FORBIDDEN).send({});
    }
  };
}

module.exports = checkPermission;
```

4.3 routes\user.js

```
let express=require('express');
let {UserModel} = require('../model');
let jwt = require('jsonwebtoken');
let config = require('../config');
+let checkLogin = require('../checkLogin');
+let checkPermission = require('../checkPermission');
const router = express.Router();
//增加
+router.post('/user', checkLogin,checkPermission('admin'),async (req, res) => {
    let user = req.body;
    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
    user = await UserModel.create(user);
    res.send({ status: 'ok',access: user.toJSON().access});
});
//删除
+router.delete('/user', checkLogin,checkPermission('admin'),async (req, res) => {
    console.log('req.body',req.body);
    await UserModel.remove({_id:req.body.key});
    return res.json({success:true});
});
//修改
+router.put('/user', checkLogin,checkPermission('admin'), async (req, res) => {
    let user = req.body;
    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
    user = await UserModel.findByIdAndUpdate(user.id,user,{useFindAndModify:false});
    res.send({ status: 'ok',access: user.toJSON().access});
});
//查询
+router.get('/user', checkLogin,checkPermission('admin'),async (req, res) => {
    let { current = 1, pageSize = 10,sorter,filter={},...query} = req.query;
    if(sorter){
        sorter = sorter?JSON.parse(sorter):{};
        for(let key in sorter){
            sorter[key]= sorter[key]
        }
    }
    if(filter){
        filter = filter?JSON.parse(filter):{};
        for(let key in filter){
            if(filter[key])
             query[key]=filter[key];
        }
    }
    let total = await UserModel.countDocuments(query);
    let users = await UserModel.find(query)
    .sort(sorter).skip((current-1)*pageSize).limit(pageSize);
    let dataSource = users.map(item=>item.toJSON());
    const result = {
        data: dataSource,
        total,
        success: true,
        pageSize,
        current,
    };
    return res.json(result);
});

router.post('/register', async (req, res) => {
    let user = req.body;
    let hash = require('crypto').createHash('md5').update(user.email).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${hash}?s=48`;
    user = await UserModel.create(user);
    res.send({ status: 'ok',access: user.toJSON().access});
});

router.post('/login/account', async (req, res) => {
    let query = {username:req.body.username,password:req.body.password};
    let dbUser = await UserModel.findOne(query);
    if (dbUser) {
        let user = dbUser.toJSON();
        let token = jwt.sign(user, config.secret,{expiresIn:'5d'});
        return res.send({ status: 'ok', token, type: 'account', access: user.access });
    } else {
        return res.send({
            status: 'error',
            type: 'account',
            access: 'guest'
        });
    }
});

router.get('/currentUser', async (req, res) => {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            let user = jwt.verify(authorization.split(' ')[1], config.secret);
            res.json(user);
        } catch (err) {
            res.status(401).send({});
        }
    } else {
        res.status(401).send({});
    }
});
router.get('/login/outLogin', async (req, res) => {
    res.send({ data: {}, success: true });
});
module.exports = router;
```