第一步：

配置方式

菜单模块配置：

{"staticUrl":"{host}/static/gcjg_quality_field_h5/index.html#/NewsManagement?tenantId={tenantId}&id={orgId}&type={orgType}&userToken={token}&lang={lang}"}

gcjg_quality_field_h5:所属部门对应的项目路径
NewsManagement：对应项目里面暴漏出来的路由 path

第二步：

代码基础改动，兼容之前的模式

userInfo.js 全量覆盖

```
export default {
  data() {
    return {
      userInfo: {},
      tenantConfig: {},
    };
  },
  created() {
    this.getCurrentUserInfo();
  },
  methods: {
    // 获取当前登录用户信息
    async getCurrentUserInfo() {
      this.userInfo = window.entryUserInfo || {};
      window['userInfo'] = this.userInfo || {};
      sessionStorage.setItem('userInfo',
JSON.stringify(this.userInfo));
      return Promise.resolve(this.userInfo);
    },
  },
};
```

Main.js 部分改动

```
const initVue = async () => {
  router.beforeEach((to, from, next) => {
    const token = getUrlVar('token');
    const tenantId = getUrlVar('tenantId');
    // 清除旧的userInfo并设置新的token和tenantId
    sessionStorage.removeItem('userInfo');
    sessionStorage.setItem('token', token);
    sessionStorage.setItem('tenantId', tenantId);
    const userToken = getUrlVar('userToken');
    const theme = getUrlVar('theme');
    store.commit('THEME', theme);
    sessionStorage.setItem('userToken', userToken);
    // h5单页面
    next();
  });
  // 判断是否是直接加载
  if (!isDirectlyLoad) {
    const userInfo = await api.getCurrentUserInfo();
    window['userInfo'] = userInfo || {};
    sessionStorage.setItem('userInfo', JSON.stringify(userInfo));
    window['entryUserInfo'] = userInfo || {};        You, 3分钟前 • userinfo找不
  }
  new Vue({
    el: '#app',
    router,
    components: {
      App,
    },
    store,
    template: '<App/>',
  });
};
// 基础信息来源于原生
```

Main.ts 全量代码如有差别自行替换 ， 改动不大

```
// The Vue build version to load with the `import` command
// (runtime-only or standalone) has been set in webpack.base.conf with an
alias.
import Vue from 'vue';
import App from './App.vue';
import router from './router';
import store from './store';
import api from 'api/bimoneApi';
import { fetchNativeApi } from 'api/offLineConfig';
import 'utils/flexible.js';
import { getUrlVar } from 'utils/utils.js';
import { startupParameterConfig } from 'utils/startupParameterConfig.js';
import Component from 'vue-class-component';
import echarts from 'echarts';
import Echats from 'vue-echarts';
import moment from 'moment';
import { defaultCustomConfig } from
'src/tenantConfig/standard/defaultPageConfig';
import {
  gotoPage,
  replacePage,
  getRouteParams,
```

```
    resetNavInfo,
    routeBack,
} from '@/app/pageRoute';
Vue.prototype.$gotoPage = gotoPage;
Vue.prototype.$replacePage = replacePage;
Vue.prototype.$getRouteParams = getRouteParams;
Vue.prototype.$resetNavInfo = resetNavInfo;
Vue.prototype.$routeBack = routeBack;
import { setTheme } from 'src/assets/theme/theme';
// TODO 暂时全局引入，因为组件库不知道用到了哪个组件，怕有遗漏
// 组件库引入
import '../node_modules/vant/lib/index.css';
import mobileComponents from 'gcjg-mobile-components';
Vue.use(mobileComponents);
import Vant from 'vant';
Vue.use(Vant);
import './styles/index.scss';
import { themes } from 'src/assets/theme/model.js';
window['themes'] = themes;
Vue.config.productionTip = false;
Component.registerHooks([
    'beforeRouteEnter', // 进入路由之前
    'beforeRouteLeave', // 离开路由之前
    'beforeRouteUpdate',
]);
Vue.prototype.$chart = echarts;
Vue.component('chart', Echats);
Vue.prototype.$moment = moment;
// 初始化一些配置
startupParameterConfig(getUrlVar, api);
const isDirectlyLoad = getUrlVar('isDirectlyLoad');
// 对于打过来的路由进行解析
const initVue = async () => {
    router.beforeEach((to, from, next) => {
        const token = getUrlVar('token');
        const tenantId = getUrlVar('tenantId');
        // 清除旧的 userInfo 并设置新的 token 和 tenantId
        sessionStorage.removeItem('userInfo');
        sessionStorage.setItem('token', token);
        sessionStorage.setItem('tenantId', tenantId);
        const userToken = getUrlVar('userToken');
        const theme = getUrlVar('theme');
        store.commit('THEME', theme);
        sessionStorage.setItem('userToken', userToken);
```

```
        // h5 单页面
        next();
    });
    // 判断是否是直接加载
    if (!isDirectlyLoad) {
        const userInfo = await api.getCurrentUserInfo();
        window['userInfo'] = userInfo || {};
        sessionStorage.setItem('userInfo', JSON.stringify(userInfo));
        window['entryUserInfo'] = userInfo || {};
    }
    new Vue({
        el: '#app',
        router,
        components: {
            App,
        },
        store,
        template: '',
    });
};
// 基础信息来源于原生
const initEvent = () => {
    const handleGetTenantConfig = (key, datas, result) => {
        if (!result) {
            console.log('获取基础配置失败');
            setTheme('standard');
            return;
        }
        datas = datas || {};
        const config = JSON.parse(datas.value || '{}') || {};
        const mergedConfig = Object.assign({}, defaultCustomConfig(), config);
        window['tenantConfig'] = mergedConfig;
        const { commonConfig: { form: { segments = {} } = {} } = {} } =
            mergedConfig;
        const showSegments = segments.show !== false;
        sessionStorage.setItem('showSegments', String(showSegments));
        setTheme(mergedConfig?.Theme?.theme);
        getModuleConfig();
    };
    // 封装获取模块配置的回调函数和调用
    const getModuleConfig = () => {
        fetchNativeApi(
            (key, datas, result) => {
                if (key === 'getModuleConfig' && result) {
```

```
                window['moduleConfig'] = datas || {};
            }
            initVue();
        },
        'getModuleConfig',
        'GET',
        `/construction-basic/tenantConfig/moduleConfig`,
    );
};
fetchNativeApi(
    handleGetTenantConfig,
    'getTenantConfig',
    'GET',
    `/construction-basic/tenantConfig?name=Estate_Mobile_Custom_Config`,
);
};
const loadEvent = () => {
    // 判断是否是直接加载
    if (isDirectlyLoad) {
        initVue();
    } else {
        window.Glodon3rdNativeEventOnload = () => {
            initEvent();
        };
    }
};
window.Glodon3rdNativeEvent ? initEvent() : loadEvent();
```

utils/startupParameterConfig.js 全量

```
import { customPathMatch } from 'src/router/moduleConfig';
export const startupParameterConfig = (getUrlVar, api) => {
    // 组件内部使用
    if (window['bimOneApi'] && Object.keys(window['bimOneApi']).length) {
        window['bimOneApi'] = Object.assign(window['bimOneApi'], api);
    } else {
        window['bimOneApi'] = api || {};
    }
    window['env'] = process.env.NODE_ENV;
    window['getUrlVar'] = getUrlVar;
    // 全局跳转参数
    window['getModuleUrl'] = (moduleCode) => {
        const fullPath = fileNameMatchFullPath(moduleCode);
        return `{host}${
```

```
                process.env.VUE_APP_REAL_ENV !== 'dev'
                    ? '/static/gcjg_quality_field_h5'
                    : ''

    }/index.html#/${fullPath}?tenantId={tenantId}&id={projectId}&userToken={token}
&token={token}&lang={lang}`;
    };
};

/**
 * 文件名匹配路由全路径
 */
const fileNameMatchFullPath = (moduleCode) => {
    const pathObj = customPathMatch();
    let fullPath = moduleCode;
    for (const [key, fileObj] of Object.entries(pathObj)) {
        if (fileObj.fileNames?.some((fileName) => fileName === moduleCode)) {
            fullPath = `${key}_${moduleCode}`;
            break;
        }
    }
    return fullPath;
};
```

shims-tsx.d.ts 全量覆盖

```
import Vue, { VNode } from 'vue';

declare global {
    namespace JSX {
        // tslint:disable no-empty-interface
        interface Element extends VNode {}
        // tslint:disable no-empty-interface
        interface ElementClass extends Vue {}
        interface IntrinsicElements {
            [elem: string]: any;
        }
    }
    interface Window {
        Glodon3rdNativeEventOnload: any;
        Glodon3rdNativeEvent: any;
        gldne: any;
    }
}
```

路由配置 router

router/moduleConfig.js 全量覆盖

```
// h5 单页面
import PhoneSinglePage from 'page/singlePage/PhoneSinglePage.vue';
import PhoneSinglePageTest from 'page/singlePage/PhoneSinglePageTest.vue';
import ComponentsTest from 'page/singlePage/ComponentsTest.vue';
import MaterialDictMain from 'components/materialSealingSampleZh/common/MaterialDictMain.vue';
import AddMaterialDict from 'components/materialSealingSampleZh/common/AddMaterialDict.vue';

/**
 * 默认基础路由
 * @returns basicRoutes
 */
export const basicRouters = () => {
    // 1、默认路由
    const basicRoutes = [
        {
            // h5 单页面
            path: '/phoneSinglePage',
            name: 'phoneSinglePage',
            component: PhoneSinglePage,
        },
        {
            path: '/MaterialDictMain',
            name: 'MaterialDictMain',
            component: MaterialDictMain,
        },
        {
            path: '/AddMaterialDict',
            name: 'AddMaterialDict',
            component: AddMaterialDict,
        },
    ];
    return basicRoutes;
};
/**
 *  2、自定义路由，全路径路由模式
 */
export const customPathMatch = () => {
    /**
```

```
        * 1. key（materialAcceptance） === 文件夹名字/模块的名称
        * 2. path === 'components 下的全路径'
        * 3. fileNames === 路径下所有单页面的名称
        * 例： window['getModuleUrl']('materialAcceptance_MaterialAcceptanceBrandDict')此路由业务跳转时为 key_fileNames 例： materialAcceptance_MaterialAcceptanceBrandDict
        */
    return {
        materialAcceptance: {
            path: 'materialAcceptance/park',
            fileNames: [
                // 材设进场
                'MaterialAcceptanceBrandDict',
                'MaterialAcceptanceConfirm',
                'MaterialAcceptanceDetail',
                'MaterialAcceptanceList',
                'MaterialAcceptanceProcess',
                'MaterialApproachApply',
                'MaterialApproachDataAdd',
                'MaterialEquipmentAcceptance',
                'MaterialInspectionReport',
                'MaterialSamplingInspection',
                'ProgressList',
                'SealedSamplesToGuide',
            ],
        },
    };
};
/**
 * 3、自动路由,会匹配文件加下所有的 co
 * 前提是规范写法 components/模块名/层级（企业、项目）/文件名称（保证文件名全局唯一）
 * 例： window['getModuleUrl']('SpaceEffectTemplateList')
 */
export const automaticPath = () => {
    return [];
};
```

Router.js 全量覆盖

```
/*
 * @Descripttion:
 * @version:
 * @Author: licz
 * @Date: 2021-01-12 10:59:16
```

```
 * @LastEditors: Please set LastEditors
 * @LastEditTime: 2024-05-24 17:46:53
 */
import Vue from 'vue';
import VueRouter, { RouteConfig } from 'vue-router';

// 前端监控
import ClientMonitor from 'skywalking-client-js';
import { basicRouters, customPathMatch, automaticPath } from './moduleConfig';
Vue.use(VueRouter);
/**
 * 1、默认写死路由
 */
const getBasicRoutes = () => {
    return basicRouters();
};
/**
 * 2、自定义路由，全路径拼写
 */
const getCustomRouter = () => {
    const routes: Array<RouteConfig> = [];
    const pathObj = customPathMatch();
    for (const key in pathObj) {
        const fileObj = pathObj[key];
        fileObj?.fileNames?.length &&
            fileObj?.fileNames.forEach((fileName) => {
                const route: RouteConfig = {
                    path: `/${key}_${fileName}`,
                    name: `${key}_${fileName}`,
                    component: () => import(`components/${fileObj.path}/${fileName}`),
                };
                routes.push(route);
            });
    }
    return routes;
};
/**
 * 3、自动路由,根据文件夹名找到下面 company/park 目录的所有组件加载
 */
const getAutomaticRouter = () => {
    const pageContext = require.context(
        `components/`,
        true,
        /(?:park|company)\/(.*?\.vue)/,
```

```
    );
    const folderNameArr = automaticPath();
    let pageRoutes: RouteConfig[] = [];
    folderNameArr?.length &&
        folderNameArr?.forEach((folderName) => {
            pageContext?.keys()?.map((id) => {
                if (id.includes(folderName)) {
                    const componentPath = id.replace(/\.\//g, '').replace(/\.vue$/, '');
                    const componentName = id
                        .split('/')
                        .pop()
                        .replace(/\.vue$/, '');
                    pageRoutes.push({
                        path: `/${componentName}`,
                        name: componentName,
                        component: (resolve) =>
                            require([`components/${componentPath}`], resolve),
                    });
                }
            });
        });
    return pageRoutes;
};

// 路由初始化
const initRouter = () => {
    // 1、默认路由  2、自定义路由  3、自动路由
    let routesArr: Array<RouteConfig> = [];
    const basicRoutes = getBasicRoutes();
    routesArr = routesArr.concat(basicRoutes);
    const customRouter = getCustomRouter();
    routesArr = routesArr.concat(customRouter);
    const AutoRouter = getAutomaticRouter();
    routesArr = routesArr.concat(AutoRouter);
    console.log(routesArr);

    const router = new VueRouter({
        base: process.env.BASE_URL,
        routes: routesArr,
    });
    // routerEvent(router);
    return router;
};
const routerEvent = (router) => {
```

```
    router.afterEach(() => {
        process.env.NODE_ENV === 'development' &&
            ClientMonitor.setPerformance({
                pagePath: location.href,
                useFmp: true,
                vue: Vue,
            });
    });
    // 监听注册
    process.env.NODE_ENV === 'development' &&
        ClientMonitor.register({
            service: 'gcjg_quality_field_h5', // 应用名称
            serviceVersion: 'v1', // 应用版本号
            traceSDKInternal: true, // 追踪 sdk
            PagePath: location.href, // 当前路由
            useFmp: true,
            vue: Vue,
        });
};
// 初始化路由
const router = initRouter();
export default router;
```

第三步：

开发模式

```
window.Glodon3rdNativeEvent &&
          window.Glodon3rdNativeEvent.openPage({
            url:
window['getModuleUrl']('materialAcceptance_MaterialApproachDat
aAdd'),
            params: {
              item: {
                value: {
                  add: false,
                  disabledIds: this.allFacilityIds || [],
                },
              },
            },
          });
```