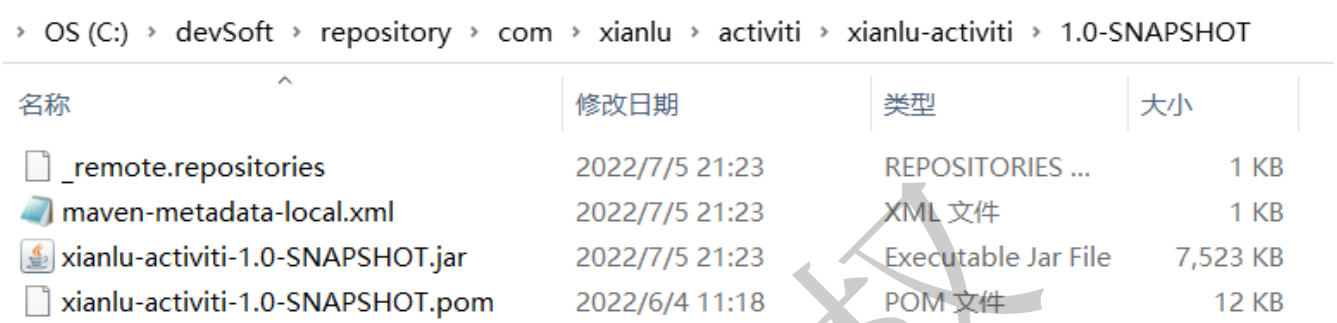
若依前后端分离版整合闲鹿工作流

若依版本	闲鹿工作流版本	作者	联系方式
任意	V1.x	闲鹿软件开发工作室	微信(手机) 13123392160、QQ 1055202292

一、本地安装工作流依赖

执行以下命令即可安装 `xianlu-activiti-1.0-SNAPSHOT.jar` 到本地 maven 仓库。

```
mvn install:install-file -Dfile=xianlu-activiti-1.0-SNAPSHOT.jar -
DgroupId=com.xianlu.activiti -DartifactId=xianlu-activiti -Dversion=1.0-SNAPSHOT -
Dpackaging=jar
```



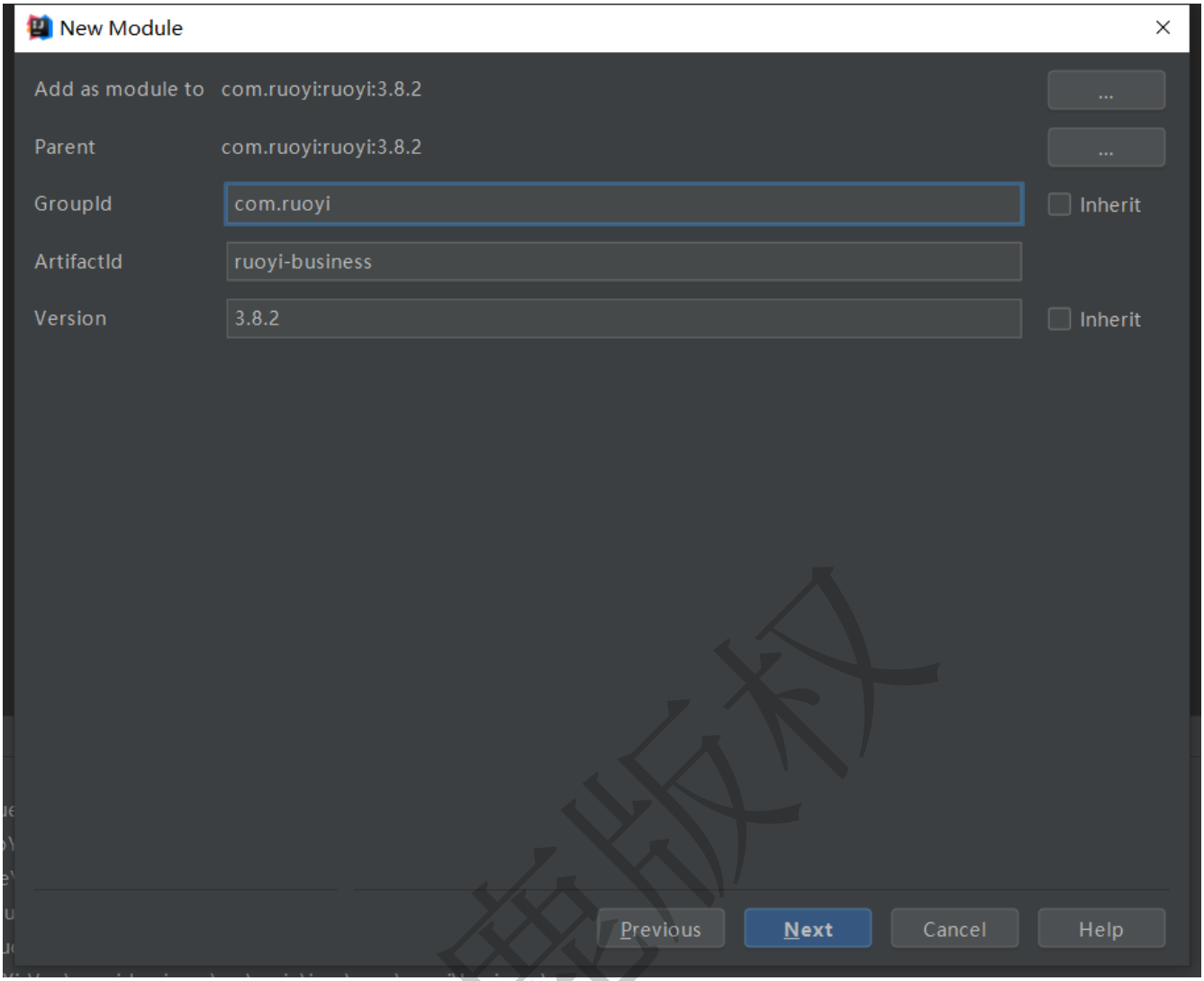
二、获取若依最新版

码云下载

三、整合工作流步骤

1、新建业务模块 module

新建业务模块 ruoyi-business，模块名称随意，或者直接用 ruoyi-admin 也行，视具体项目情况而定。



2、修改依赖配置 pom.xml

3.2.1 ruoyi-business 依赖 xianlu-activiti

```
<!-- 核心模块-->
<dependency>
  <groupId>com.ruoyi</groupId>
  <artifactId>ruoyi-framework</artifactId>
</dependency>
<!--闲鹿工作流依赖 start-->
<dependency>
  <groupId>com.xianlu.activiti</groupId>
  <artifactId>xianlu-activiti</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
```

```
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
</dependency>
<!--Shiro核心框架 -->
<dependency>
  <groupId>org.apache.shiro</groupId>
  <artifactId>shiro-core</artifactId>
  <version>1.8.0</version>
</dependency>
<!--客户端 jwt-->
<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>3.4.0</version>
</dependency>
<!-- spring security 安全认证 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- websocket -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
<dependency>
  <groupId>org.activiti</groupId>
  <artifactId>activiti-spring-boot-starter-basic</artifactId>
  <version>6.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!--activiti modeler 5.22 start-->
<dependency>
  <groupId>org.activiti</groupId>
  <artifactId>activiti-json-converter</artifactId>
  <version>6.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.activiti</groupId>
      <artifactId>activiti-bpmn-model</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!-- xml解析依赖-->
```

```
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-codec</artifactId>
  <version>1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-css</artifactId>
  <version> 1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-svg-dom</artifactId>
  <version>1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-svggen</artifactId>
  <version>1.7</version>
</dependency>
<!-- xml解析依赖-->
<!-- Swagger3依赖 -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>io.swagger</groupId>
      <artifactId>swagger-models</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!-- oss -->
<dependency>
  <groupId>com.qiniu</groupId>
  <artifactId>qiniu-java-sdk</artifactId>
  <version>[7.2.0, 7.2.99]</version>
</dependency>
<dependency>
  <groupId>com.aliyun.oss</groupId>
  <artifactId>aliyun-sdk-oss</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
  <groupId>com.qcloud</groupId>
  <artifactId>cos_api</artifactId>
  <version>5.5.9</version>
  <exclusions>
    <exclusion>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
    </exclusion>
```

```
  </exclusions>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
<dependency>
  <groupId>de.schlichtherle.truelicense</groupId>
  <artifactId>truelicense-core</artifactId>
  <version>1.33</version>
</dependency>
<!--闲鹿工作流依赖 end-->
```

3.2.2 ruoyi-admin 依赖 xianlu-activiti 和 ruoyi-business

```
<!--闲鹿工作流依赖 start-->
<dependency>
  <groupId>com.xianlu.activiti</groupId>
  <artifactId>xianlu-activiti</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>com.ruoyi.business</groupId>
  <artifactId>ruoyi-business</artifactId>
  <version>4.7.0</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
</dependency>
<!--Shiro核心框架 -->
<dependency>
  <groupId>org.apache.shiro</groupId>
  <artifactId>shiro-core</artifactId>
  <version>1.8.0</version>
</dependency>
<!--客户端 jwt-->
<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>3.4.0</version>
</dependency>
```

```xml
<!-- spring security 安全认证 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- websocket -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
<dependency>
  <groupId>org.activiti</groupId>
  <artifactId>activiti-spring-boot-starter-basic</artifactId>
  <version>6.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!--activiti modeler 5.22 start-->
<dependency>
  <groupId>org.activiti</groupId>
  <artifactId>activiti-json-converter</artifactId>
  <version>6.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.activiti</groupId>
      <artifactId>activiti-bpmn-model</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!-- xml解析依赖-->
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-codec</artifactId>
  <version>1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-css</artifactId>
  <version> 1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-svg-dom</artifactId>
  <version>1.7</version>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-svggen</artifactId>
  <version>1.7</version>
```

```
</dependency>
<!-- xml解析依赖-->
<!-- Swagger3依赖 -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
  <exclusions>
    <exclusion>
      <groupId>io.swagger</groupId>
      <artifactId>swagger-models</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!-- oss -->
<dependency>
  <groupId>com.qiniu</groupId>
  <artifactId>qiniu-java-sdk</artifactId>
  <version>[7.2.0, 7.2.99]</version>
</dependency>
<dependency>
  <groupId>com.aliyun.oss</groupId>
  <artifactId>aliyun-sdk-oss</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
  <groupId>com.qcloud</groupId>
  <artifactId>cos_api</artifactId>
  <version>5.5.9</version>
  <exclusions>
    <exclusion>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
<dependency>
  <groupId>de.schlichtherle.truelicense</groupId>
  <artifactId>truelicense-core</artifactId>
  <version>1.33</version>
</dependency>
<!--闲鹿工作流依赖 end-->
```


3、修改应用配置 application.yml

3.3.1 application-druid.yml

数据库 url 添加参数配置 `&nullCatalogMeansCurrent=true`

例如：

```
url: jdbc:mysql://localhost:3307/ry-vue_20220220?
useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serve
rTimezone=GMT%2B8&nullCatalogMeansCurrent=true
```

3.3.2 application.yml

开启 activiti 日志输出（可选）

```
# 日志配置
logging:
  level:
    # 开启 activiti sql 日志
    org.activiti.engine.impl.persistence.entity: debug
```

activiti 相关配置

```
# Spring配置
spring:
  # activiti 模块
  activiti:
    # 添加这个配置就不会一直调用了
    # 在流程引擎启动就激活AsyncExecutor,异步 true  false 关闭   (切记关闭)
    async-executor-activate: false
    # 解决启动报错: class path resource [processes/] cannot be resolved to URL because it
does not exist
    check-process-definitions: false
    # 检测身份信息表是否存在
    db-identity-used: false
```

其他增强配置

```
xianlu:
  # 附件上传位置
  upload: C:/xianlu-activiti
  # 数据库名
  table-schema: ry_20211019
  # 消息通知域名
  websocket-origin: ws://localhost:81
```

4、跨域支持

修改 SecurityConfig 类的 configure() 回调方法，以便支持静态资源访问和**表单设计器**前端跨域请求。

```
@Override
protected void configure(HttpSecurity httpSecurity) throws Exception {
  httpSecurity
    // 允许跨域 (aisuda 接口)
    .cors().and()
    // amis 字体等资源
    .antMatchers("/**/*.woff2", "/**/*.woff", "/**/*.ttf").anonymous()
    // modeler 资源
    .antMatchers(
    "/modeler/model/**",
    "/modeler/editor-app/**",
    "/modeler/editor/**",
    "/definition/readResource/**",
    "/api/process/readResource"
    ).anonymous()
    .antMatchers("/websocket/**").permitAll()
}
```

5、修改启动类

在 ruoyi-admin 下 RuoYiApplication.java 增加以下注解。

```
@ComponentScan({ "com.ruoyi", "com.xianlu.activiti" })
@SpringBootApplication(exclude = {
        DataSourceAutoConfiguration.class,
        org.activiti.spring.boot.SecurityAutoConfiguration.class,
        // ruoyi-vue 使用 spring security，不用排除
//
org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration.class
})
public class RuoYiApplication {}
```

6、初始化工作流表

工作流引擎表均为 `act_xxx` 开头，在应用首次启动时候会自动创建。

其他表或数据依次执行以下增强 sql 文件即可生成。

```
sql增强.sql
ruoyi_vue_menu增强.sql
```

7、工作流增强JS

在 ruoyi-ui/src/api 下新建目录 activiti，然后新建工作流增强JS文件 activiti4ry-vue.js。

四、业务表单集成流程

1、代码生成表单CRUD

若依常用功能，不做赘述。如以下生成**员工请假**的业务。

2、员工请假列表JS增强

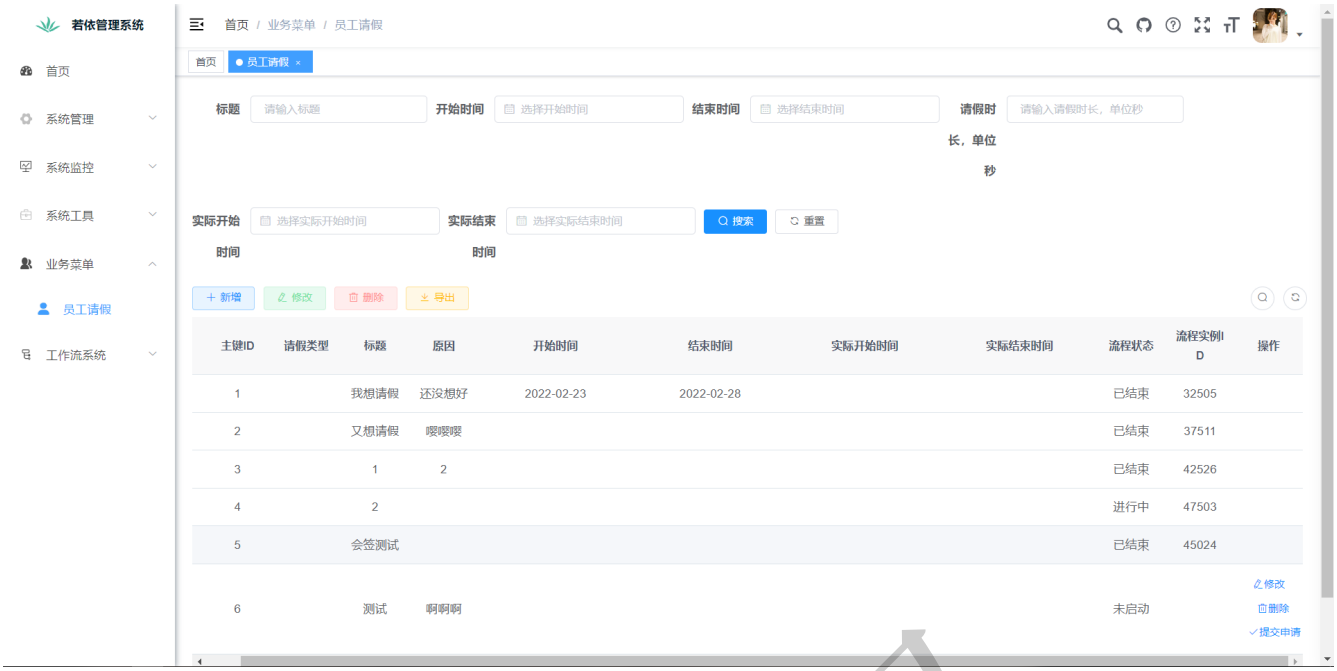
在员工请假列表 leave/index.vue 添加如下代码。

```
<template>
  ...
    <el-table>
    <!--自定义流程列-->
    <el-table-column label="流程状态" align="center" prop="processStatus" />
    <el-table-column label="流程实例ID" align="center" prop="instanceId" />
    <el-table-column label="操作" align="center" class-name="small-padding fixed-width">
      <template slot-scope="scope" v-if="!scope.row.instanceId">
        ...
        <el-button
                  size="mini"
                  type="text"
                  icon="el-icon-check"
                  @click="handleSubmit(scope.row)"
                  v-hasPermi="['business:leave:edit']"
                  >提交申请</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>
<script>
    ...
```

```
    import { listInstanceIdList, submitApply } from '@/api/activiti/activiti4ry-vue';
  ...
  export default {
    name: "Leave",
    data() {
      return {
        ...
        // 流程功能变量
        processVariable: {
          timestamp: new Date().getTime()
        }
      }
    },
    methods: {
      /** 查询员工请假列表 */
      getList() {
        this.loading = true;
        listLeave(this.queryParams).then(response => {
          this.leaveList = response.rows;
          this.total = response.total;
          this.loading = false;

          listInstanceIdList('biz_leave', this, 'leaveList');
        });
      },
      /** 提交申请按钮操作 */
      handleSubmit(row) {
        submitApply(row, this);
      },
    }
  }
</script>
```

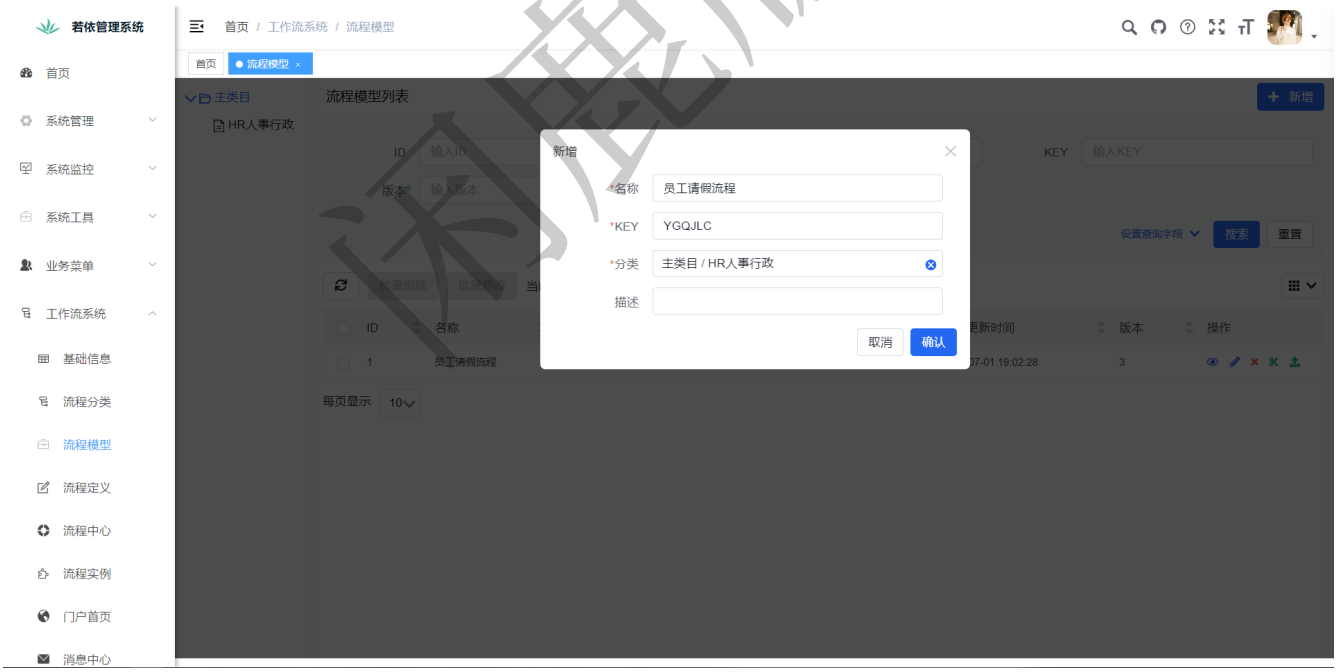
刷新页面，即可查看效果。

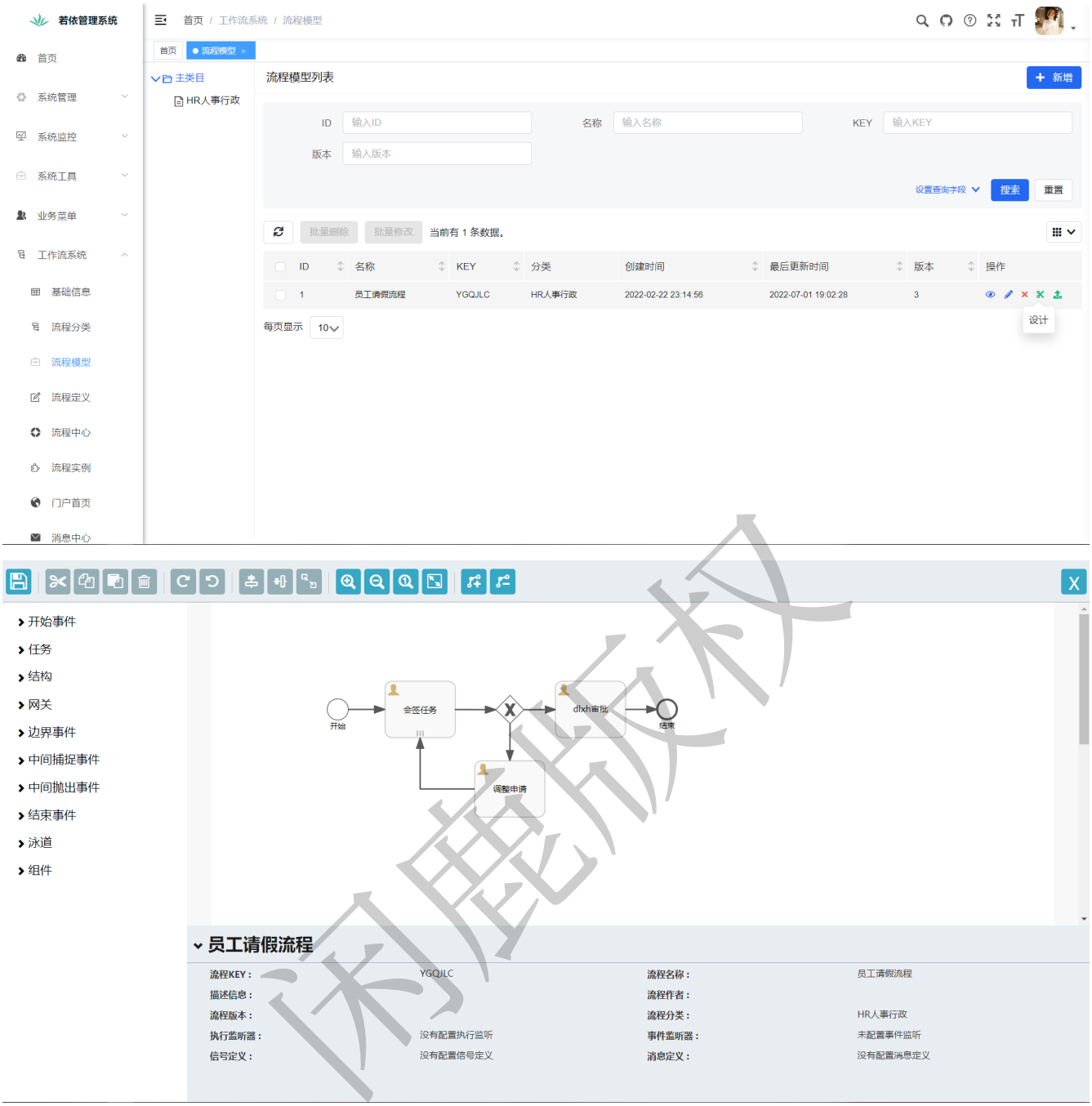


3、流程设计与部署

4.3.1 流程设计

首先需要在**工作流系统 -> 流程模型**新建流程模型，然后点击**设计**按钮，打开流程在线设计窗口。流程设计好后，别忘了点击左上角保存按钮。

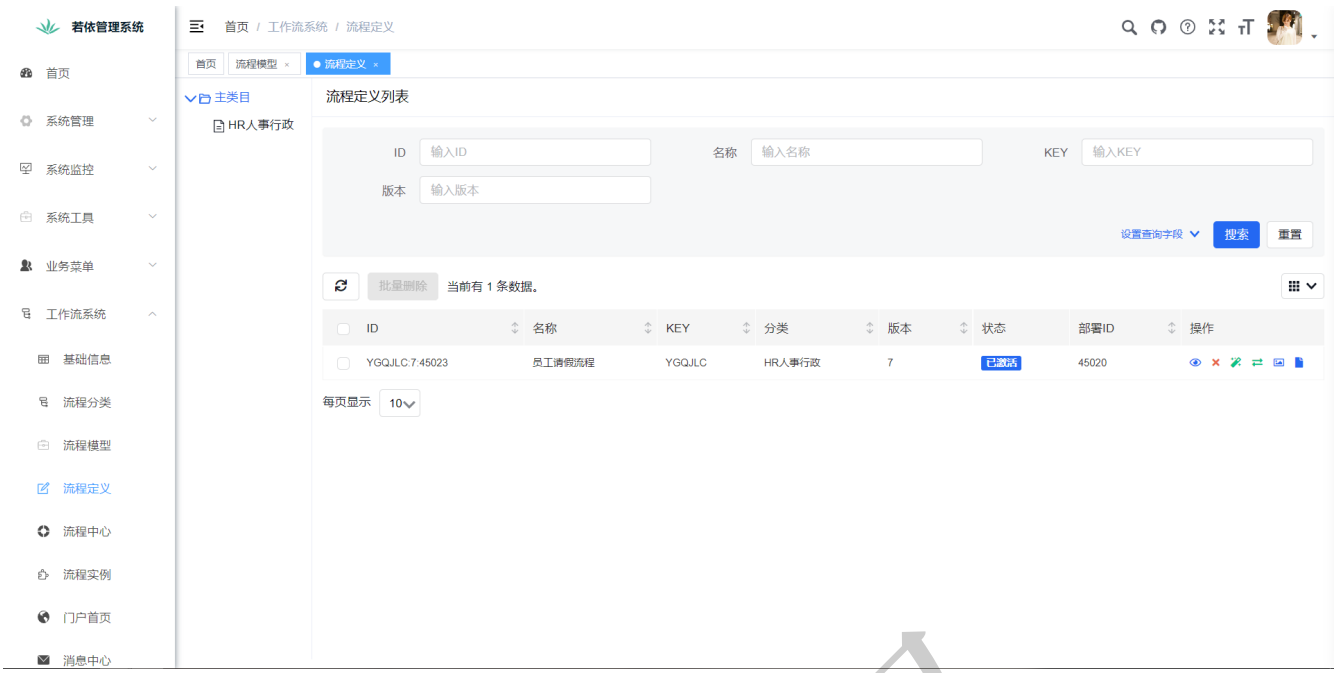




具体的流程设计细节，详见用户使用手册，此处不做过多描述。

4.3.2 流程部署

流程模型设计好后，不能直接使用，需要点击**部署**按钮，发布到流程定义，方能使用。

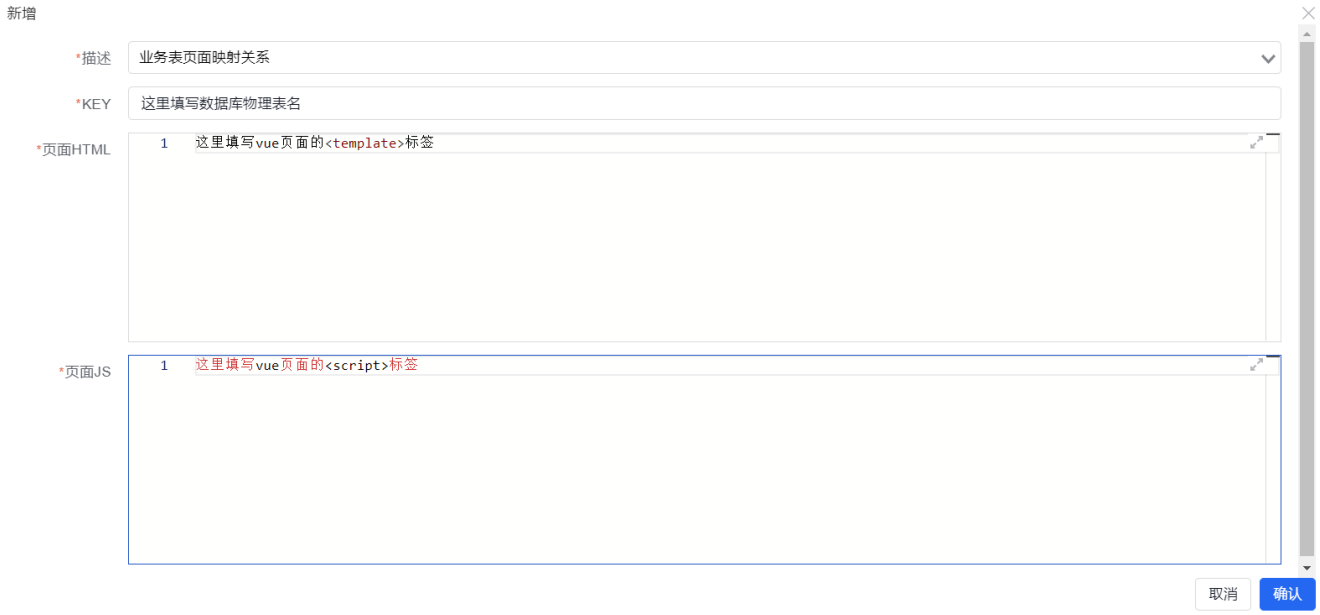


4、表单和流程关联

为实现业务表单和流程关联，需要在**工作流系统 -> 基础信息**添加以下两条记录。

4.4.1 业务表流程定义关系

4.4.2 业务表页面映射关系



页面HTML参考：

```
<el-form ref="form" :model="form" :rules="rules" label-width="80px">
    <el-form-item label="标题" prop="title">
        <el-input v-model="form.title" placeholder="请输入标题" readonly />
    </el-form-item>
    <el-form-item label="原因" prop="reason">
        <el-input v-model="form.reason" type="textarea" placeholder="请输入内容" />
    </el-form-item>
    <el-form-item label="开始时间" prop="startTime">
        <el-date-picker clearable size="small" v-model="form.startTime" type="date" value-
format="yyyy-MM-dd"
            placeholder="选择开始时间">
        </el-date-picker>
    </el-form-item>
    <el-form-item label="结束时间" prop="endTime">
        <el-date-picker clearable size="small" v-model="form.endTime" type="date" value-
format="yyyy-MM-dd"
            placeholder="选择结束时间">
        </el-date-picker>
    </el-form-item>
    <el-form-item label="请假时长，单位秒" prop="totalTime">
        <el-input v-model="form.totalTime" placeholder="请输入请假时长，单位秒" />
    </el-form-item>
    <el-form-item label="实际开始时间" prop="realityStartTime">
        <el-date-picker clearable size="small" v-model="form.realityStartTime" type="date"
value-format="yyyy-MM-dd"
            placeholder="选择实际开始时间">
        </el-date-picker>
    </el-form-item>
    <el-form-item label="实际结束时间" prop="realityEndTime" v-if="dynamicShow">
        <el-date-picker clearable size="small" v-model="form.realityEndTime" type="date"
value-format="yyyy-MM-dd"
```

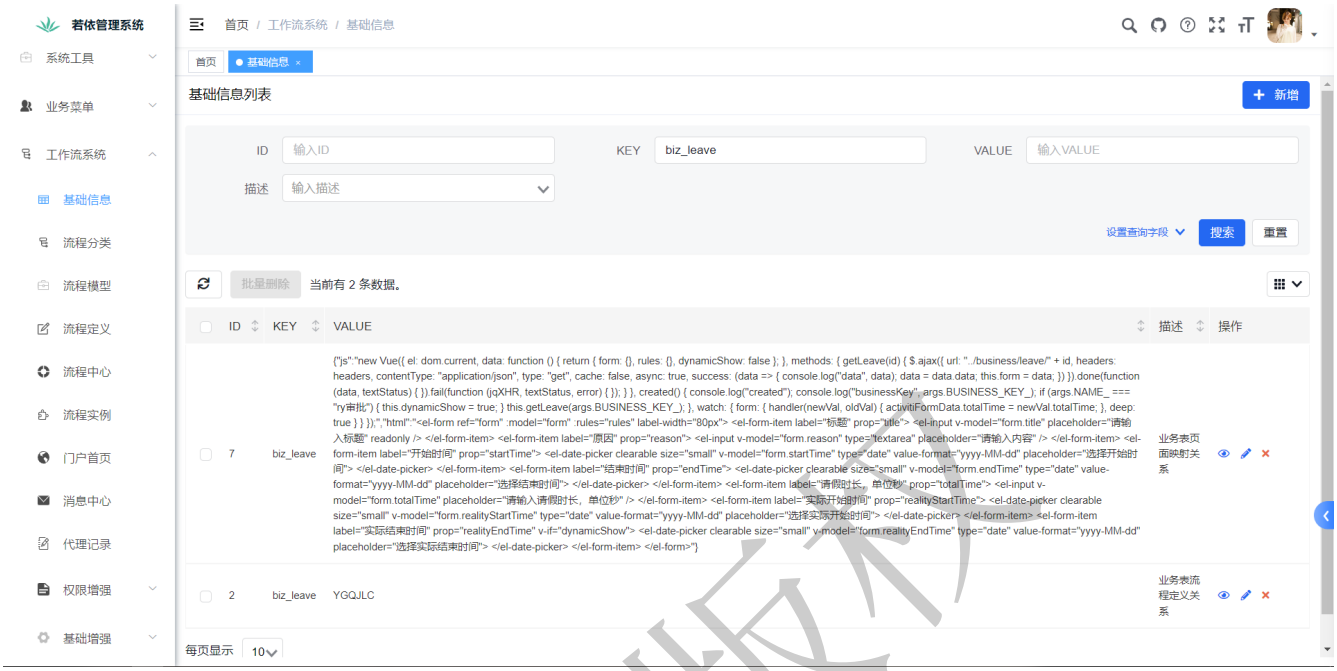
```
            placeholder="选择实际结束时间">
        </el-date-picker>
    </el-form-item>
</el-form>
```

页面JS参考：

```
new Vue({
    el: dom.current,
    data: function () {
      return {
        form: {},
          rules: {},
          dynamicShow: false
      };
    },
    methods: {
        getLeave(id) {
            $.ajax({
                url: "../business/leave/" + id,
                headers: headers,
                contentType: "application/json",
                type: "get",
                cache: false,
                async: true,
                success: (data => {
                    console.log("data", data);
                    data = data.data;
                    this.form = data;
                })
            }).done(function (data, textStatus) {
            }).fail(function (jqXHR, textStatus, error) {
            });
        }
    },
    created() {
        console.log("created");
        console.log("businessKey", args.BUSINESS_KEY_);
        if (args.NAME_ === "ry审批") {
            this.dynamicShow = true;
        }
        this.getLeave(args.BUSINESS_KEY_);
    },
    watch: {
        form: {
            handler(newVal, oldVal) {
                activitiFormData.totalTime = newVal.totalTime;
            },
            deep: true
        }
```

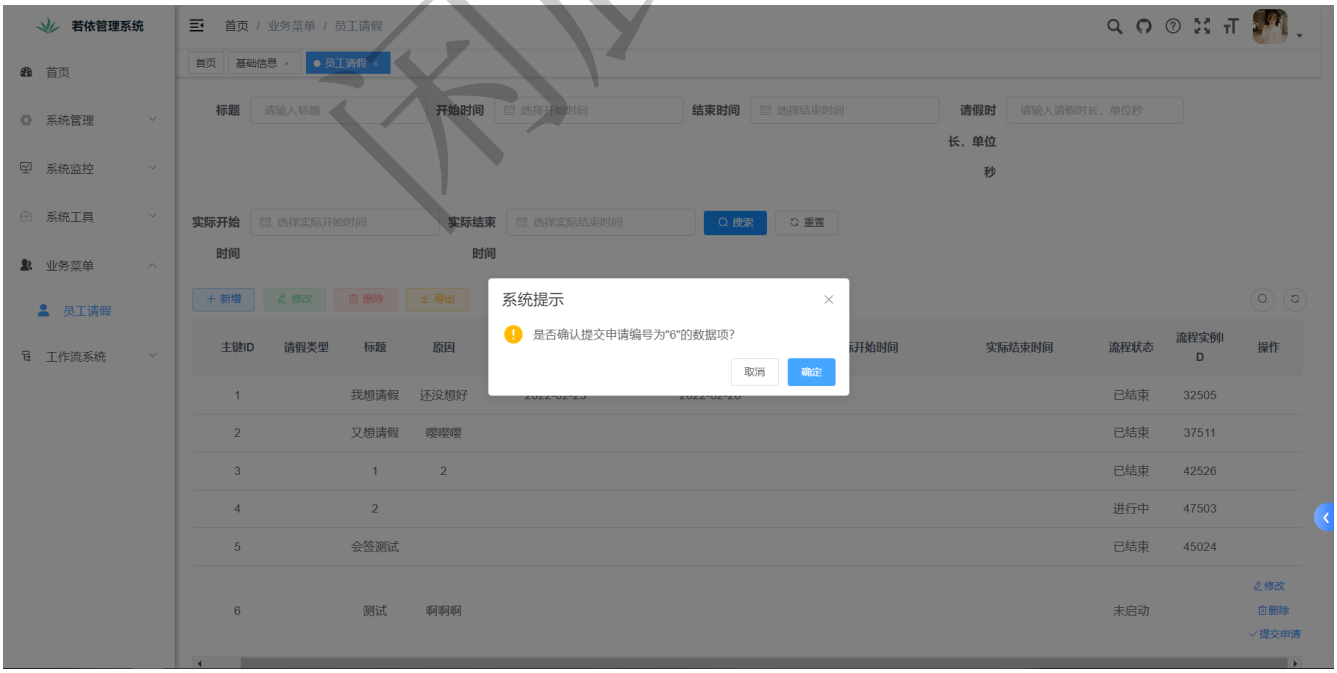
```
    }
});
```

员工请假表单和流程，关联结果参考：



5、发起员工请假流程

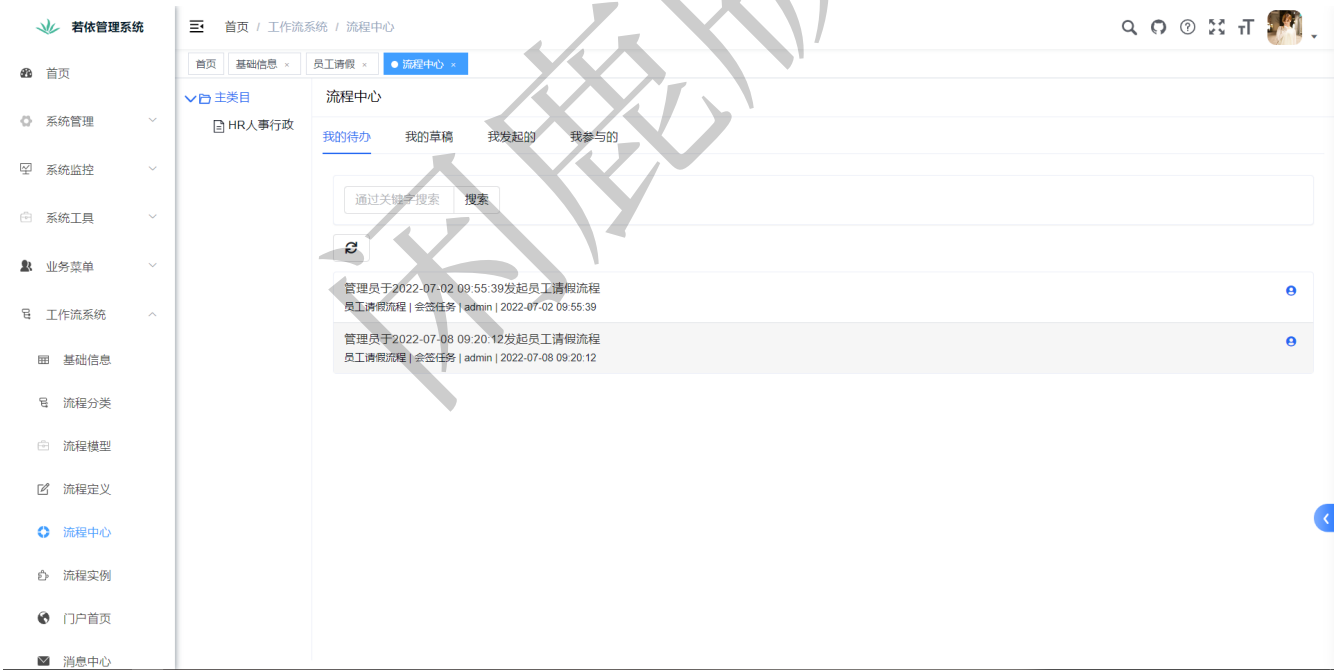
点击员工请假列表中的**提交申请**按钮，即可提交流程。



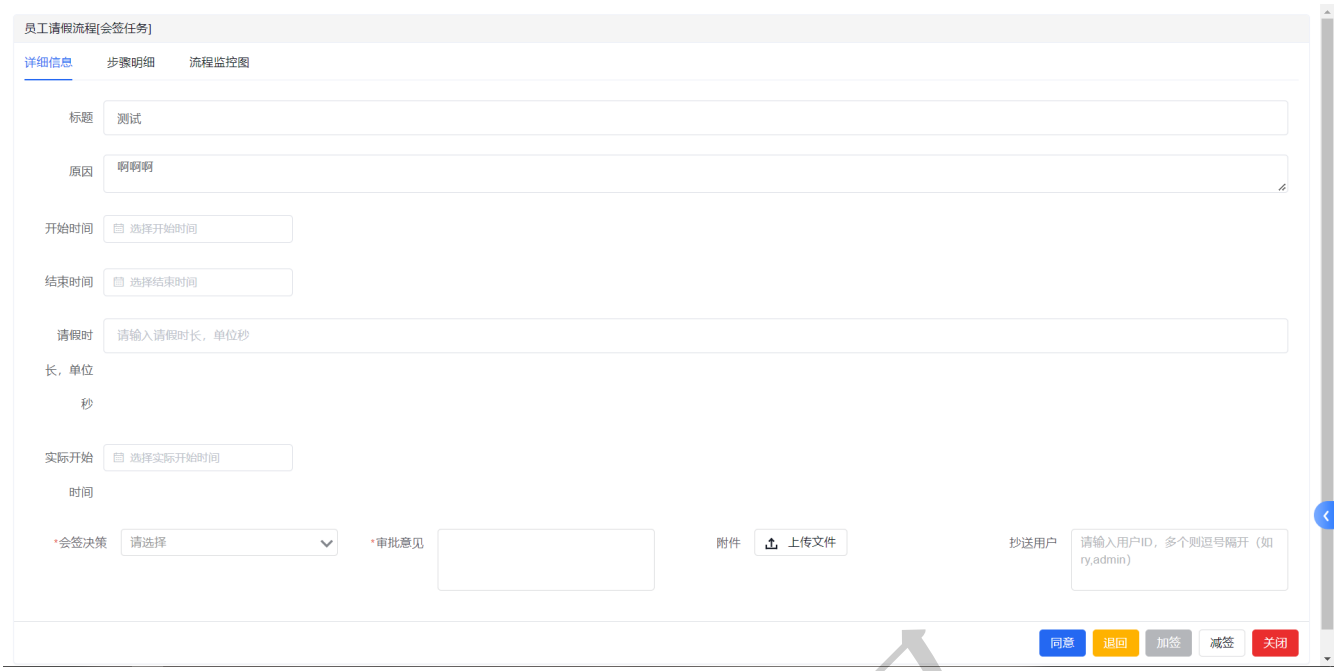
提交成功后，可以看到流程状态为进行中，并且得到流程实例ID。流程详情可到**工作流系统 -> 流程中心**查看。

6、流程中心

流程中心聚合显示了**我的待办、我的草稿(在线表单)、我发起的、我参与的**等流程相关信息。



点击办理，可对该流程进行操作，如：同意、退回等。



五、在线表单集成流程

使用在线表单，需要额外获取表单设计器服务，没有的话联系作者。

1、启动表单设计器

5.1.1 安装 fis3

默认用户已经安装好 node 环境，没有的话自行查找资料安装。

执行以下命令全局安装 fis3 服务

```
npm install -g fis3
```

验证 fis3 已经安装

```
C:\Users\10001392>fis3 -v

 [INFO] Currently running fis3 (D:\devSoft\nodejs\node_global\node_modules\fis3)

  v3.4.46

 __/\\\\\\\\\\\\\\\__/\\\\\\\\\\\_____/\\\\\\\\\\\___
  _\/\\\///////////__\/////\\\///____/\\\/////////\\\_
   _\/\\\_____________\/\\\_______\//\\\______\///__
    _\/\\\\\\\\\\\_____\/\\\________\////\\\_________
     _\/\\\///////______\/\\\___________\////\\\_____
      _\/\\\_____________\/\\\_____________\////\\\___
       _\/\\\_____________\/\\\_____/\\\_____\//\\\__
```

```
        _\/\\\_____________/\\\\\\\\\\\_\///\\\\\\\\\\\\/___
         _\///_____________\///////////____\///////////_____
```

5.1.2 修改请求接口地址

找到表单设计器 public/request 目录下的 helper.js，修改实际的请求地址。（请求地址详见 ruoyi-ui/vue.config.js，默认端口为 80）

```
// 全局路径（开发）
const commonUrl = 'http://localhost/dev-api'
// 全局路径（生产）
// const commonUrl = 'http://localhost/prod-api'
```

5.1.3 指定端口启动表单设计器前端服务

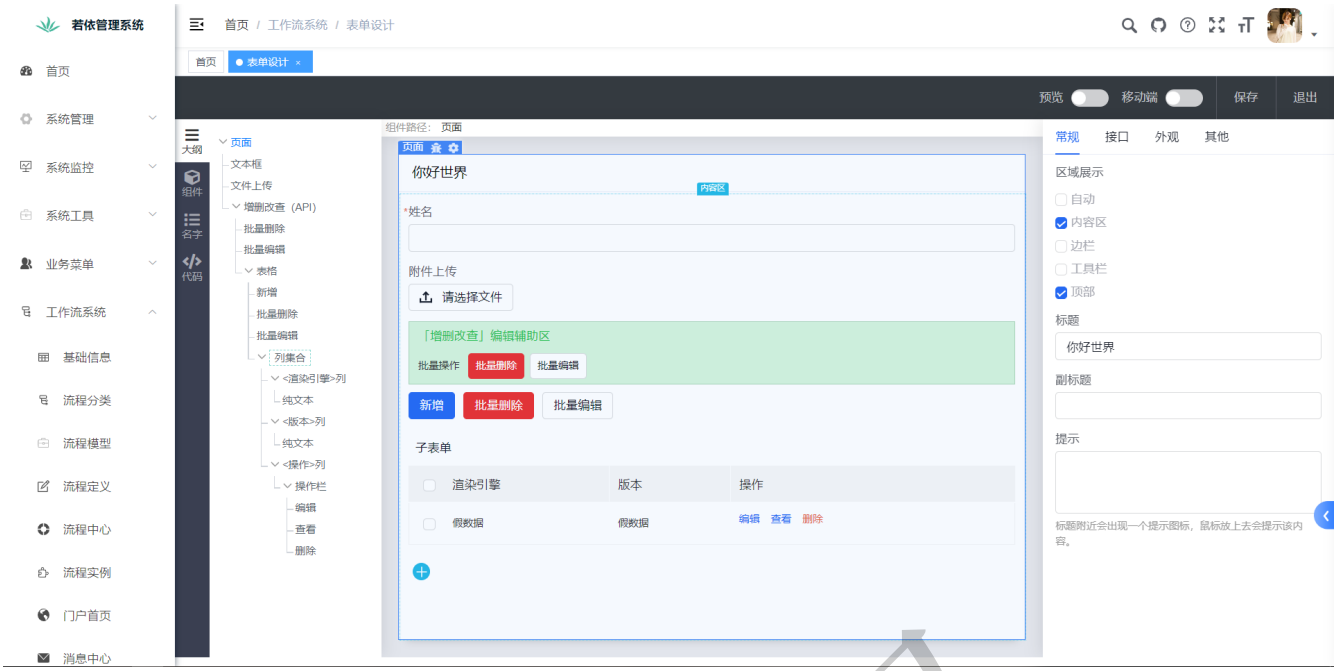
执行以下命令，表单设计器即可运行于 8082 端口。

```
fis3 server start --www public --port 8082 --no-daemon
```


2、表单设计

打开**工作流系统 -> 在线表单 -> 表单设计。**

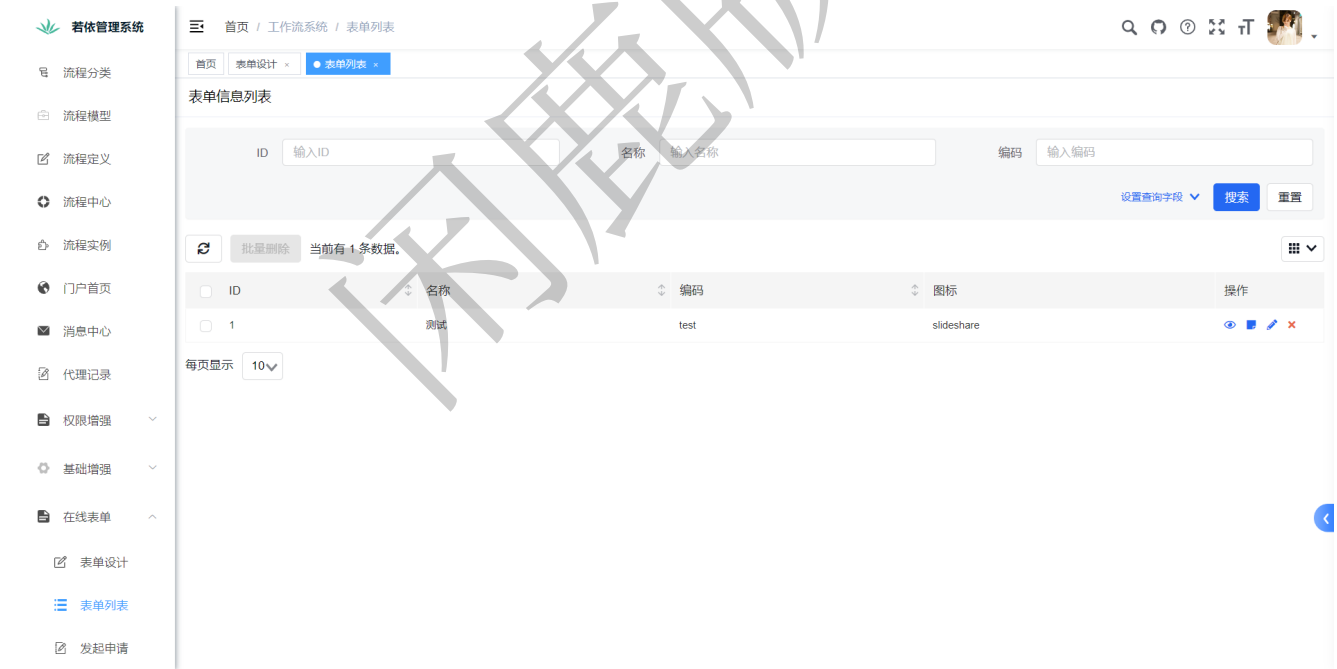
点击左侧小铅笔图标可进入表单编辑界面。表单设计好后，别忘了点击右上角保存按钮。



具体的表单设计细节，详见用户使用手册，此处不做过多描述。

3、表单列表

打开**工作流系统 -> 在线表单 -> 表单列表**，可查看历史表单列表。



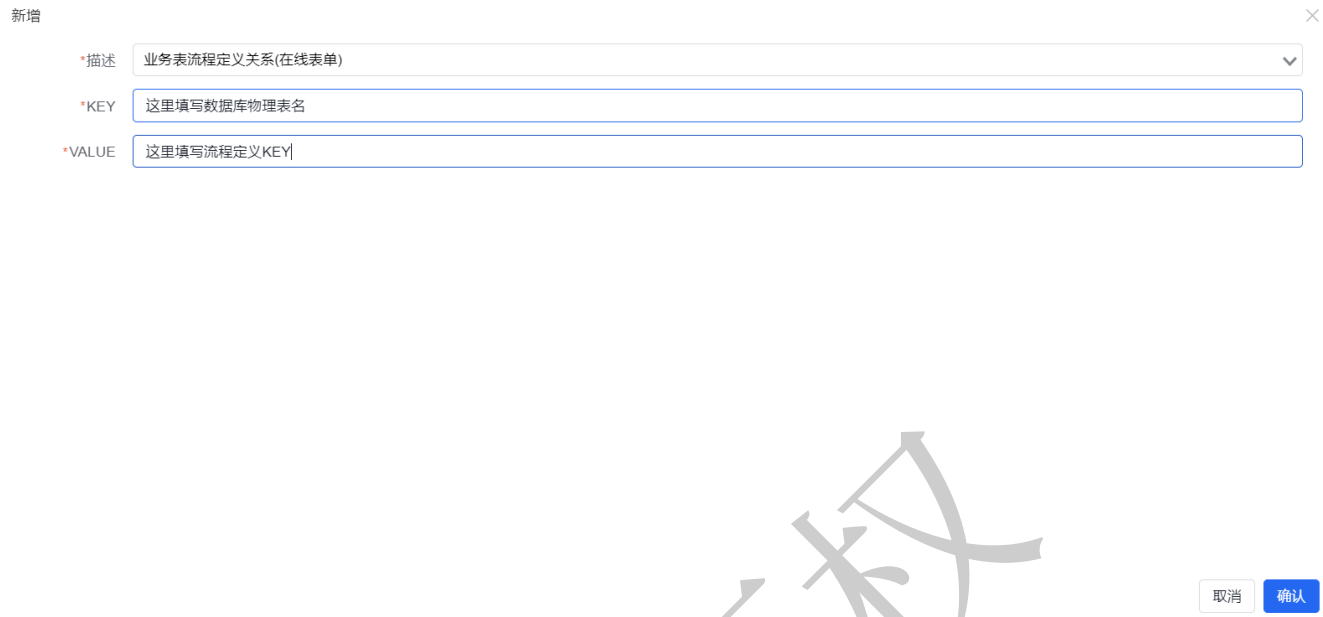
4、流程设计与部署

与 **4.3 业务表单集成流程/流程设计与部署** 同理，不做赘述。

5、表单和流程关联

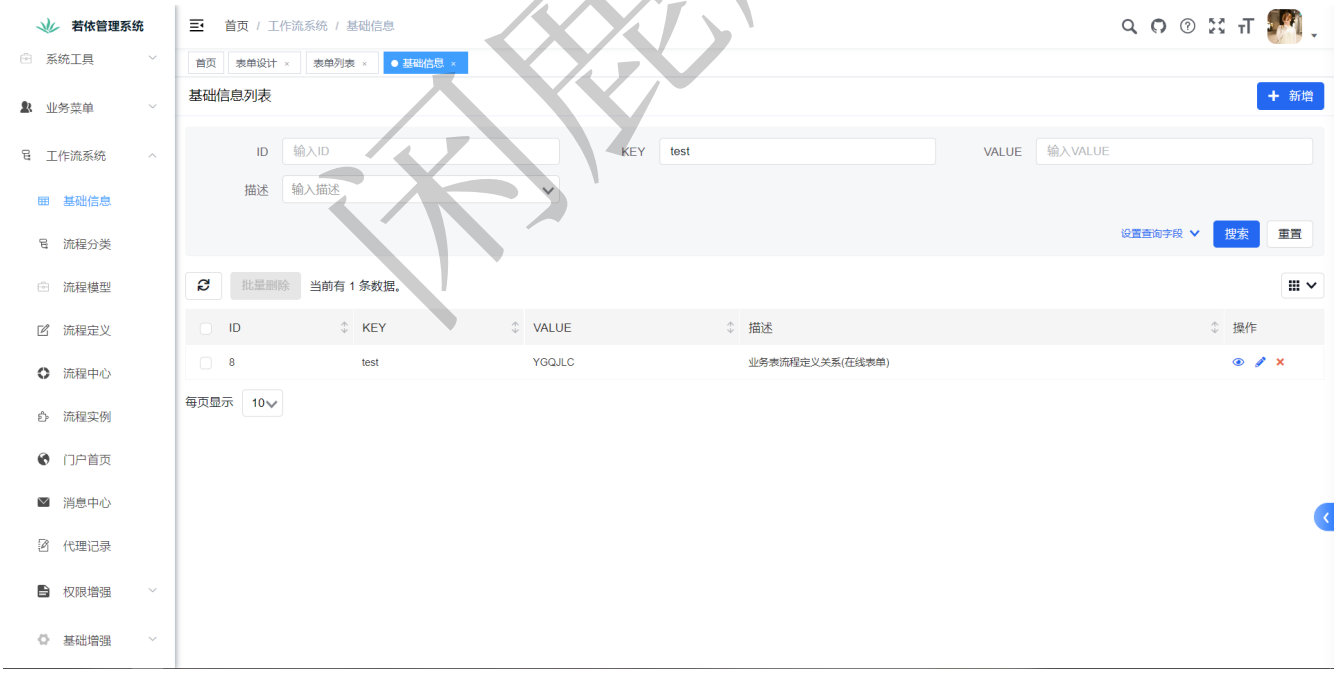
为实现在线表单和流程关联，需要在**工作流系统 -> 基础信息**添加以下一条记录。

5.5.1 业务表流程定义关系(在线表单)



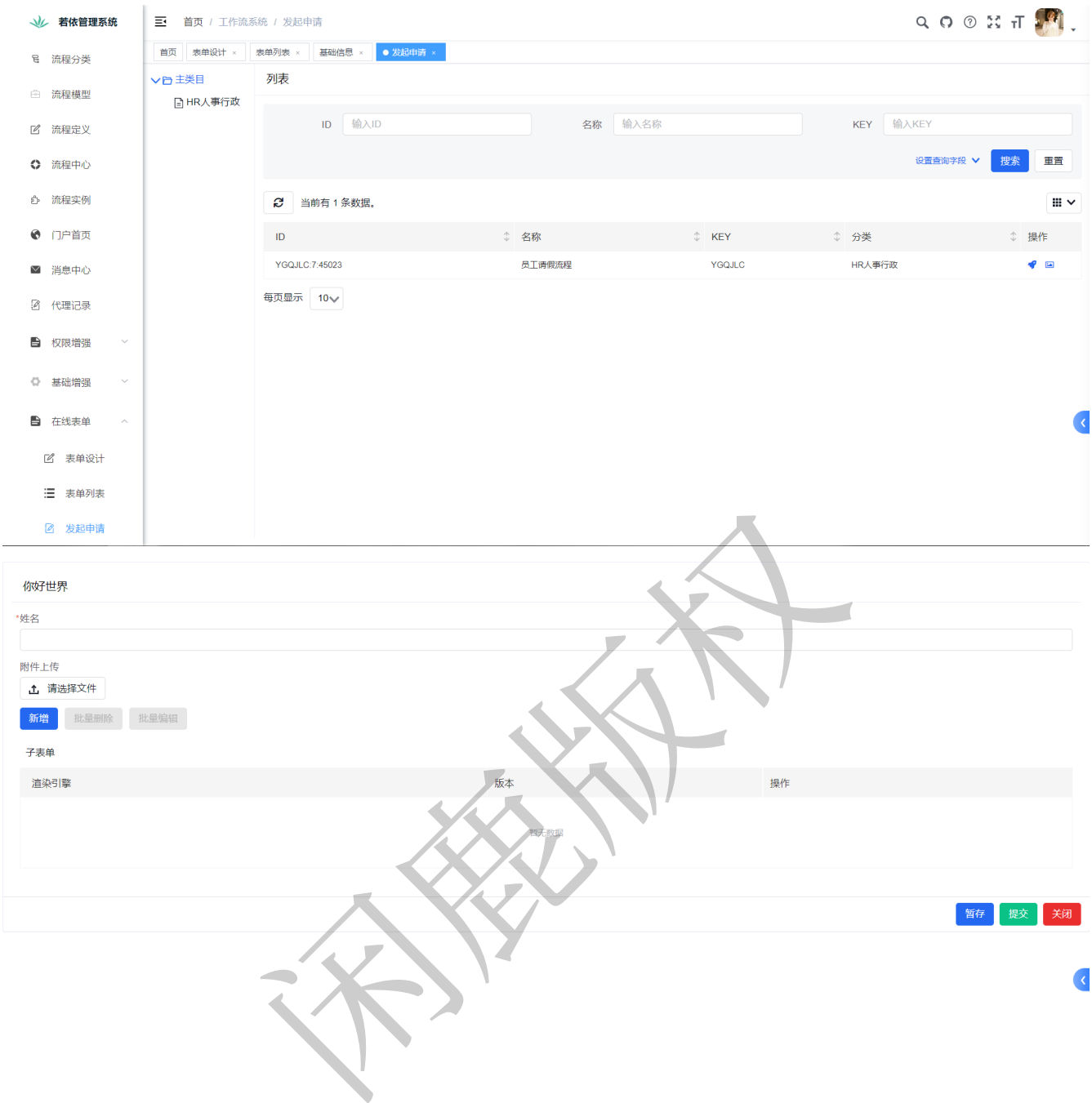
在线表单，不需要配置业务表页面映射关系，系统会自动找到对应的页面。

在线表单和流程，关联结果参考：



6、发起在线表单流程

找到 **工作流系统 -> 在线表单 -> 发起申请** 点击员工请假会签版中的**发起**按钮，即可在新窗口打开流程表单界面。



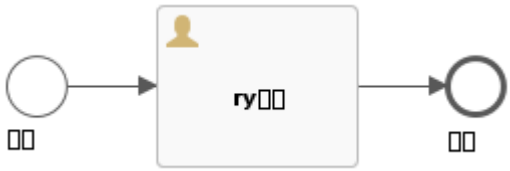
流程表单支持暂存操作，草稿位于 **流程中心 -> 我的草稿**，点击提交按钮即可提交流程。

7、流程中心

与 **4.6 业务表单集成流程/流程中心** 同理，不做赘述。

六、其他

1、Linux 部署项目，流程图文字乱码



乱码原因是因为 Linux 上没有安装中文字体**宋体**，手动安装即可。

找到附件宋体`simsun.ttc`，或者 Windows 系统下，依次打开路径 C:\Windows\Fonts，找到**宋体 常规。**

将字体`simsum.ttc`通过 SFTP 等方式上传到 Linux 服务器上。

找到 jre 字体路径，如：`/usr/local/java/jdk1.8.0_171/jre/lib/fonts`，在 fonts 下新建 **fallback**目录，并放入宋体字体。

执行命令参考：

```
cd /usr/local/java/jdk1.8.0_171/jre/lib/fonts
mkdir fallback
cd fallback
mv ~/simsun.ttc ./

mkfontscale
mkfontdir

# 若找不到命令先进行安装相关工具：
yum install -y fontconfig mkfontscale
```

最后重启项目。

重启完项目，还需要进入流程模型，**重新部署到流程定义**，再预览新的流程图，即正常。

参考链接

activiti流程图在linux上中文乱码问题解决

（文档介绍完毕，如有疑问请联系作者：闲鹿软件开发工作室，微信(手机) 13123392160、QQ 1055202292）