项目成本管理设计方案说明书

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制人 | 丰硕 | 日期 | 2019.11.28 |
| 审核人 |  | 日期 |  |
| 批准人 |  | 日期 |  |

目录

[项目成本管理设计方案说明书 1](#_Toc26137044)

[1.设计概述 2](#_Toc26137053)

[1.1 任务和目标 2](#_Toc26137054)

[1.1.1 需求概述 2](#_Toc26137055)

[1.1.2 运行环境概述 2](#_Toc26137056)

[2 系统详细需求分析 3](#_Toc26137057)

[2.1 详细需求分析 3](#_Toc26137058)

[2.2 详细系统运行环境及限制条件分析接口需求分析 3](#_Toc26137059)

[3 总体方案确认 4](#_Toc26137060)

[3.1 系统总体结构确认 4](#_Toc26137061)

[4 系统详细设计 5](#_Toc26137062)

[4.1 系统结构设计及子系统划分 5](#_Toc26137063)

[4.2 系统功能模块详细设计 5](#_Toc26137064)

[4.3 系统界面详细设计 6](#_Toc26137065)

[5 信息编码设计 9](#_Toc26137066)

[5.1 代码结构设计 9](#_Toc26137067)

1.设计概述

1.1 任务和目标

项目成本管理软件是承包人为使[项目成本控制](https://baike.baidu.com/item/%E9%A1%B9%E7%9B%AE%E6%88%90%E6%9C%AC%E6%8E%A7%E5%88%B6/377533)在计划目标之内所作的预测、计划、控制、调整、[核算](https://baike.baidu.com/item/%E6%A0%B8%E7%AE%97/445086)、分析和考核等管理工作的平台

1.1.1 需求概述

需要对各类项目进行详细管理，主要实现项目管理（项目登记，收款登记，项目财务核算，项目全成本核算），报销支付，工资支付，现金流水，工时统计，账目汇总统计。

1.1.2 运行环境概述

本系统运行于电脑端

CPU最低要求600兆赫 (MHz) Pentium III 处理器及以上

推荐1 GHz Pentium 4处理器

内存 最低RAM要求256MB

硬盘需要500 MB以上的可用空间

驱动器需要CD-ROM或DVD-ROM驱动器

显示Super VGA (800x600)或更高分辨率的显示器（颜色设置为32位真彩色）

鼠标Microsoft鼠标或兼容的指点设备

2 系统详细需求分析

2.1 详细需求分析

• 详细功能需求分析:软件项目管理贯穿项目全生命周期,实现项目需求、项目规划、成本估计、成本核算、项目任务、质量跟踪的统一管理,支撑内外部项目研发,项目实施、项目维护，实现项目的可视化开发。项目成本、进度、范围是项目管理的核心组成部分，通过系统进进行成本管理可以实现成本估计、项目资金出入账管理、成本核算，对于项目的进度与范围控制，人力资源调度具有重要的意义，保证项目的成功实施，并未后续的项目提供成本估算依据。

• 详细性能需求分析 : 软件项目管理平台考虑到不同操作者的计算机水平不同，在系统的设计时要注重易用性，使大多数的使用者都可以较好地使用系统。由于各种原因会进行误操作，或者输入错误的数据等，系统应能够对这些情况进行处理，系统应保证个别模块出现问题不会对其他模块造成影响。

• 详细系统运行环境及限制条件分析 :前端界面使用HTML5作为主要开发框架,jQuery作为辅助开发框架进行开发。

2.2 详细系统运行环境及限制条件分析接口需求分析

• 系统接口需求分析:无

• 现有硬、软件资源接口需求分析:打印机接口，传真机接口；

• 引进硬、软件资源接口需求分析:无

3 总体方案确认

3.1 系统总体结构确认

1系统组成、逻辑结构及层次确认：系统由项目管理，账目管理、资金管理和合同管理四大部分组成，逻辑结构为网状结构，

2应用系统结构确认：

1）项目管理，实现项目基本信息的管理，包括项目的增删改查，不同的部门人员只能看到自己部门的项目。

2）合同管理，实现合同的登记、审批流程与和合同查询。

3）项目收款管理，进行收款登记、发票登记。开出发票需要进行审批。

4）项目报销流程，针对差旅、采购、外包等各项费用进行填写与报销，审批后能够打印。

5）成本查询，查询每个项目与合同的报销总额度、入账总额度。

3系统工作流程确认：自顶向下：项目登记-收入项目库-项目预算、收款估计、报销支付-核算库。

4 系统详细设计

4.1 系统结构设计及子系统划分

划分为：项目管理（项目登记，收款登记，项目财务核算，项目全成本核算），报销支付，工资支付，现金流水，工时统计，账目汇总统计，合同管理等

4.2 系统功能模块详细设计

1项目管理，实现项目基本信息的管理，包括项目的增删改查，不同的部门人员只能看到自己部门的项目。

2合同管理，实现合同的登记、审批流程与和合同查询。

3项目收款管理，进行收款登记、发票登记。开出发票需要进行审批。

4项目报销流程，针对差旅、采购、外包等各项费用进行填写与报销，审批后能够打印。

5成本查询，查询每个项目与合同的报销总额度、入账总额度。

模块编号： 1

模块名称： 项目管理

输入： 项目名

处理： 排序

算法描述： 快速排序

输出：表格

模块编号： 2

模块名称：合同管理

输入： 合同

处理： 保存

算法描述：

输出： 合同

模块编号： 3

模块名称： 报销支付

输入： 数字

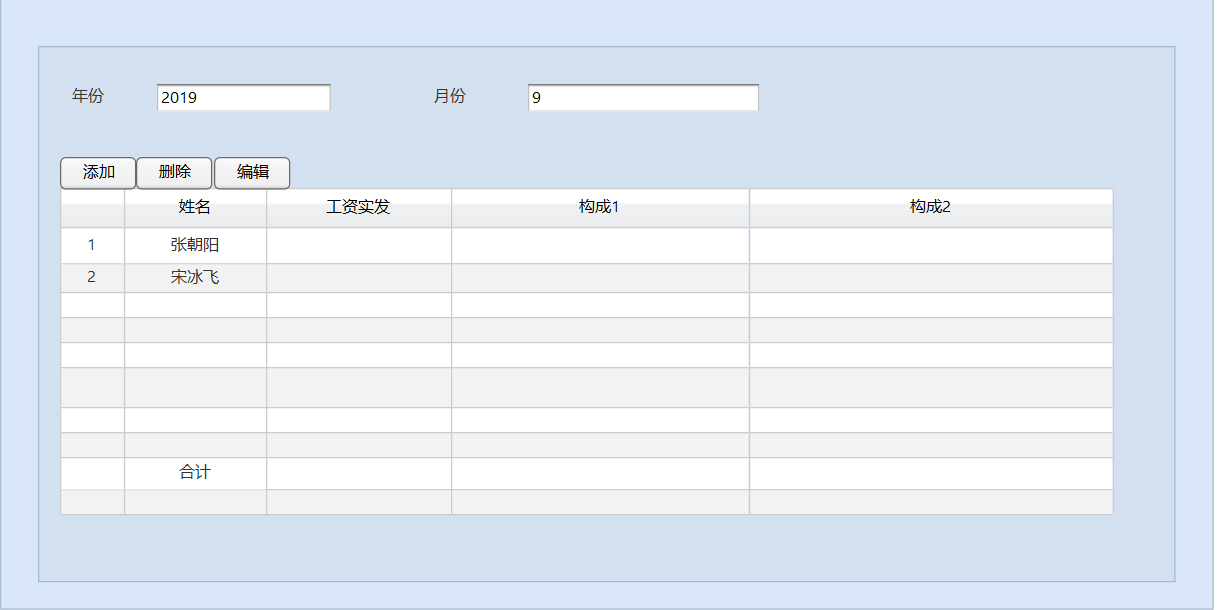
处理：

算法描述：

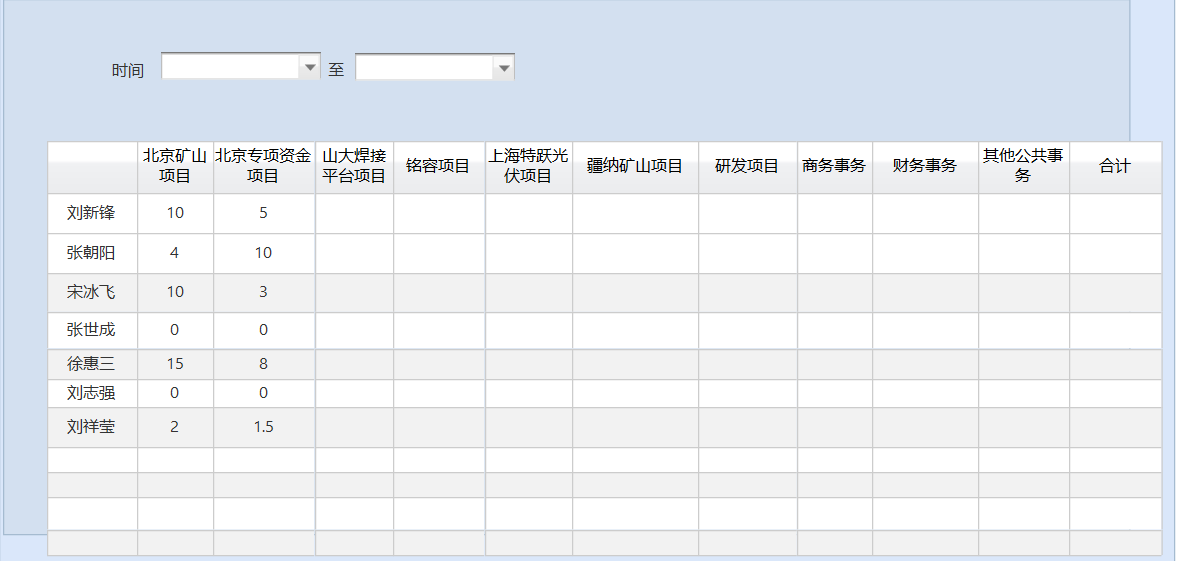
输出： 数字

4.3 系统界面详细设计











5 信息编码设计

5.1 代码结构设计

a、 接口

import java.util.concurrent.Callable;

public interface CallableTask extends Callable<String> {

String readAndSendData2ws(String content);

}

b、抽象类

public abstract class AbstractDataTask implements CallableTask {

protected String filePath;

public AbstractDataTask(String path) {

this.filePath = path;

}

@Override

public String call() throws Exception {

return readAndSendData2ws(filePath);

}

@Override

public String readAndSendData2ws(String filePath) {

try {

File file = new File(filePath);

List<String> lines = new ArrayList<String>(1);

if (file != null && file.exists()) {

lines = Files.readLines(file, Charsets.UTF\_8);

file.delete();

}

for (String line : lines) {

sendData2sw(line);

}

} catch (Exception e) {

e.printStackTrace();

return filePath;

}

return filePath;

}

public abstract void sendData2sw(String content);

}

c、具体实现类

public class UploadDataTask extends AbstractDataTask {

private UploadService uploadService;

public UploadDataTask(String path, UploadService uploadService) {

super(path);

this.uploadService = uploadService;

}

@Override

public void sendData2sw(String line) {

if (StringUtil.isNotBlank(line)) {

final String[] k\_v = line.split("\\s+");

if (k\_v.length < 2) {

return;

}

ThreadpoolProxy.executeTaskByWork(new Runnable() {

@Override

public void run() {

String key = k\_v[0];

String value = k\_v[1];

uploadService.sendData2TargetPlace(key, value);

}

});

}

}

}

a、线程池代理类

public class ThreadpoolProxy {

// 阻塞队列

private static final BlockingQueue<Runnable> WORKER\_BLOCKINGQUEUE = new ArrayBlockingQueue<Runnable>(50);

// 线程工厂

private static final ThreadFactory FACTORY =

new ThreadFactoryBuilder().setNameFormat("thread-%d").setDaemon(true).build();

// 核心线程池

private static final int CORE\_WORKER\_POOL\_SIZE = 8;

// 最大线程池

private static final int MAX\_WORKER\_POOL\_SIZE = 8;

// 线程保持时间

private static final int KEEP\_ALIVE\_TIME = 300;

// 工作线程池负责处理请求

private static final ExecutorService WORKER\_POOL =

new ThreadPoolExecutor(CORE\_WORKER\_POOL\_SIZE, MAX\_WORKER\_POOL\_SIZE, KEEP\_ALIVE\_TIME, TimeUnit.SECONDS,

WORKER\_BLOCKINGQUEUE,

FACTORY, new RejectedExecutionHandlerImpl());

// 阻塞队列

private static final BlockingQueue<Runnable> BOSS\_BLOCKINGQUEUE = new ArrayBlockingQueue<Runnable>(1);

// 核心线程池

private static final int CORE\_BOSS\_POOL\_SIZE = 1;

// 最大线程池

private static final int MAX\_BOSS\_POOL\_SIZE = 1;

// boss线程池

private static final ExecutorService BOSS\_POOL =

new ThreadPoolExecutor(CORE\_BOSS\_POOL\_SIZE, MAX\_BOSS\_POOL\_SIZE, KEEP\_ALIVE\_TIME, TimeUnit.SECONDS,

BOSS\_BLOCKINGQUEUE,

FACTORY, new RejectedExecutionHandlerImpl());

// 监听boss线程池

private static final ListeningExecutorService LISTEN\_BOSS\_POOL =

MoreExecutors.listeningDecorator(BOSS\_POOL);

public static void executeTaskByWork(Runnable task) {

WORKER\_POOL.submit(task);

}

public static ListenableFuture<String> executeTaskByBoss(Callable<String> task) {

return LISTEN\_BOSS\_POOL.submit(task);

}

}

业务常量

public interface DataExecutor {

// 默认休眠时间10s

static final int DEFAULT\_SLEEP\_SECOND = 10;

// 默认最大尝试次数 5次

static final int DEFAULT\_MAX\_TRYTIMES = 5;

// 默认超过最大尝试次数的休眠时间 10min

static final int DEFAULT\_TRYTIME\_OVERSLEEP = 1000 \* 60 \* 10;

// 最大阻塞时间

static final int MAX\_BLOCKING\_TIME = 1000 \* 60 \* 60;

}

