

《Python Web开发》课程设计报告

2022～2023学年第二学期

学 院：

系 专业：

年级班级：

学 号：

姓 名：

指导教师：

成 绩：

教学与研究部 制

**目录**

[1 概述 3](#_Toc139100862)

[1.1设计主题 3](#_Toc139100863)

[1.2目的意义 4](#_Toc139100864)

[1.3工具和技术简介 4](#_Toc139100865)

[2 项目总体设计 5](#_Toc139100866)

[2.1项目需求分析 5](#_Toc139100867)

[2.2概要设计 6](#_Toc139100868)

[2.3数据库设计 7](#_Toc139100869)

[3 详细设计与编码实现 7](#_Toc139100870)

[3.1前后台各模块的实现过程及关键代码 7](#_Toc139100871)

[4 测试 13](#_Toc139100872)

[5 总结 14](#_Toc139100873)

**企业人员信息管理系统**

# 1 概述

## 1.1设计主题

基于Django的企业人员管理系统是一个用于管理和跟踪公司内部企业人员信息的应用程序。它使用MySQL数据库作为后端存储，前端使用HTML、CSS和JavaScript进行设计。

系统的主题是为公司提供一个集中管理企业人员信息的平台，方便HR部门和管理层对企业人员进行管理和监控。系统具有以下功能：

1. 企业人员信息管理：系统允许HR部门添加、编辑和删除企业人员的基本信息，包括姓名、性别、年龄等。
2. 企业人员考勤管理：系统可以记录企业人员的考勤信息，包括上班时间、下班时间和请假情况。管理员可以查看和导出企业人员的考勤记录，以便进行工资结算和绩效评估。
3. 企业人员绩效评估：系统允许管理员对企业人员进行绩效评估，包括工作表现、目标达成情况等。管理员可以为每个企业人员设置评分和评语，并将评估结果保存在系统中。
4. 企业人员培训管理：系统可以记录企业人员参加的培训活动和培训成绩。管理员可以为企业人员安排培训，并查看企业人员的培训记录和成绩。
5. 企业人员福利订单管理：系统可以记录企业人员享受的福利和福利发放情况。管理员可以为企业人员添加福利，如健康保险、年假等，并记录福利的发放日期和金额。
6. 企业人员通知管理：系统可以向企业人员发送通知和公告。管理员可以编写通知内容，并选择发送给指定的企业人员或全体企业人员。

通过使用Django框架和MySQL数据库，系统可以实现数据的高效存储和管理。Django提供了强大的ORM（对象关系映射）功能，使得开发者可以轻松地与数据库进行交互。同时，Django还提供了丰富的后台管理界面，使得管理员可以方便地进行数据的增删改查操作。前端使用HTML、CSS和JavaScript进行设计，可以实现用户友好的界面和交互效果。通过使用CSS和JavaScript，可以实现页面的美化和动态效果，提升用户体验。

## 1.2目的意义

基于Django的企业人员管理系统具有以下重要的目的和意义：

1. 提高工作效率：企业人员管理系统可以集中管理和跟踪企业人员的信息，包括基本信息、考勤记录、绩效评估、培训记录等。通过系统化的管理，HR部门和管理层可以更快速、准确地获取企业人员信息，从而提高工作效率。

2. 加强企业人员沟通和交流：企业人员管理系统可以实现向企业人员发送通知和公告的功能。管理员可以通过系统向企业人员发送重要通知、公司政策变动等信息，提高企业人员的沟通和交流效率。同时，企业人员也可以通过系统提交请假申请、培训需求等，与HR部门进行交流。

3. 提升数据安全性：通过使用MySQL数据库和Django框架，企业人员管理系统可以实现数据的安全存储和访问控制。管理员可以设置权限，限制不同角色的用户对敏感数据的访问。同时，系统还可以进行数据备份和恢复，以防止数据丢失和损坏。

总之，基于Django的企业人员管理系统，可以提高工作效率、优化人力资源管理、加强企业人员沟通和交流、提升数据安全性，并为公司的数据分析和决策提供支持。这将有助于提升公司的管理水平和竞争力。

## 1.3工具和技术简介

企业人员管理系统涉及到以下工具和技术：

1. Django框架：Django是一个高级的Python Web框架，它提供了一系列的工具和库，用于快速开发Web应用程序。Django框架提供了强大的ORM（对象关系映射）功能，使得开发者可以使用Python代码与数据库进行交互，而无需编写复杂的SQL语句。同时，Django还提供了丰富的后台管理界面，方便管理员进行数据的增删改查操作。

2. MySQL数据库：MySQL是一种常用的关系型数据库管理系统，它支持高性能的数据存储和查询。在企业人员管理系统中，MySQL数据库用于存储企业人员的各种信息，如基本信息、考勤记录、绩效评估等。通过使用MySQL数据库，可以实现数据的高效存储和管理。

3. HTML：HTML是一种用于创建网页结构的标记语言。在企业人员管理系统中，HTML用于构建系统的页面结构，包括表单、表格、列表等。通过使用HTML，可以实现页面的结构化和语义化，提高页面的可读性和可维护性。

4. CSS：CSS是一种用于设置网页样式的样式表语言。在企业人员管理系统中，CSS用于设置页面的样式，包括字体、颜色、布局等。通过使用CSS，可以实现页面的美化和统一，提升用户体验。

5. JavaScript：JavaScript是一种用于实现网页交互效果的脚本语言。在企业人员管理系统中，JavaScript用于实现页面的动态效果，如表单验证、数据展示等。通过使用JavaScript，可以增加页面的交互性和用户友好性。

通过使用这些工具和技术，基于Django的企业人员管理系统可以实现数据的高效存储和管理，页面的美化和动态效果，以及用户友好的界面和交互效果。这将提高系统的易用性和用户体验，提升公司的管理效率和竞争力。

# 2 项目总体设计

## 2.1项目需求分析

基于Django的企业人员管理系统使用MySQL数据库和HTML+CSS+JavaScript前端设计，以下是对系统需求的分析：

1. 登录和身份验证：系统应该提供登录功能，企业人员可以使用用户名和密码登录系统。登录成功后，系统应该进行身份验证，根据企业人员的角色和权限，显示不同的功能和界面。

2. 企业人员信息管理：系统应该提供企业人员信息的管理功能，包括企业人员的基本信息、联系方式、部门、职位等。管理员可以添加、编辑和删除企业人员信息。企业人员可以查看和更新自己的信息。

3. 部门管理：系统应该记录企业的部门信息。管理员员可以选择员工的部门所属。

4. 通知和公告：系统应该提供通知和公告功能，管理员可以发布重要通知、公司政策变动等信息。企业人员可以查看通知和公告。

通过以上需求分析，可以设计出一个功能完善、易于使用的企业人员管理系统。系统将提高公司的管理效率、加强企业人员的沟通和交流、提升数据安全性，并为公司的数据分析和决策提供支持。这将有助于提升公司的管理水平和竞争力。

## 2.2概要设计

企业人员管理系统概要设计：

1. 前端页面设计：

* 登录页面：提供用户名和密码输入框，以及登录按钮。使用JavaScript进行表单验证，确保输入的数据合法性。
* 企业人员列表页面：展示所有企业人员的基本信息，包括姓名、部门、职位等。可以使用HTML表格和CSS样式进行布局和美化。
* 企业人员详情页面：展示企业人员的详细信息，包括基本信息、联系方式、考勤记录、绩效评估等。
* 部门详情页面：展示部门的详细信息，包括部门名称等。
* 通知和公告页面：展示公司的通知和公告，包括标题、内容、发布时间等。

2. 前后端交互设计：

* 使用Django的模型（Model）定义数据库表，并通过模型类与数据库进行交互。
* 使用Django的视图（View）处理前端请求，查询数据库并返回数据给前端页面。
* 使用Django的模板（Template）生成动态的HTML页面，将后端数据渲染到前端页面。
* 使用JavaScript处理前端页面的动态效果，如表单验证、数据展示等。

通过以上的概要设计，可以实现一个基于Django的企业人员管理系统，使用MySQL数据库存储数据，并通过HTML+CSS+JavaScript前端设计实现页面的美化和动态效果。系统将提供企业人员信息管理、通知和公告等功能，提升公司的管理效率和企业人员的工作体验。

## 2.3数据库设计

数据库设计：

* 管理人员表（Employee）：包含管理人员的ID、账户密码、姓名、性别等字段。
* 企业人员表（Employee）：包含企业人员的ID、姓名、性别、出生日期、联系方式等字段。
* 部门表（Department）：包含部门的ID、名称、描述等字段。
* 订单表（Training）：包含订单的ID、订单名称、订单内容、订单时间等字段。
* 事务表（Notification）：包含事务的ID、标题、内容、发布时间等字段。

## 2.4 E-R图



# 3 详细设计与编码实现

## 3.1前后台各模块的实现过程及关键代码

基于Django的员工管理系统使用MySQL数据库存储数据，并通过HTML+CSS+JavaScript前端设计的系统，拥有管理人员管理模块、企业人员管理模块、部门管理模块和订单管理模块。下面是各模块的实现过程及关键代码。

1. 管理人员管理模块：

创建一个名为managers的Django应用程序，并在managers/models.py中定Manager模型，包含姓名、职位等字段。在managers/views.py中创建视图函数，如manager\_list用于显示所有管理人员列表。创建一个名为manager\_list.html的模板，使用Django模板语言渲染管理人员列表。

2. 企业人员管理模块：

创建一个名为employees的Django应用程序，并在employees/models.py中定义Employee模型，包含姓名、职位、部门等字段。在employees/views.py中创建视图函数，如employee\_list用于显示所有企业人员列表。创建一个名为employee\_list.html的模板，使用Django模板语言渲染企业人员列表。

3. 部门管理模块：

创建一个名为departments的Django应用程序，并在departments/models.py中定义Department模型，包含部门名称、描述等字段。在departments/views.py中创建视图函数，如department\_list用于显示所有部门列表。创建一个名为department\_list.html的模板，使用Django模板语言渲染部门列表。

4. 订单管理模块：

创建一个名为orders的Django应用程序，并在orders/models.py中定义Order模型，包含订单号、客户、订单金额等字段。在orders/views.py中创建视图函数，如order\_list用于显示所有订单列表。创建一个名为order\_list.html的模板，使用Django模板语言渲染订单列表。

关键代码示例：

models.py文件中的模型定义示例：

python

from django.db import models

class Manager(models.Model):

name = models.CharField(max\_length=100)

position = models.CharField(max\_length=100)

def \_\_str\_\_(self):

return self.name

class Employee(models.Model):

name = models.CharField(max\_length=100)

position = models.CharField(max\_length=100)

department = models.ForeignKey(Department, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.name

class Department(models.Model):

name = models.CharField(max\_length=100)

description = models.TextField()

def \_\_str\_\_(self):

return self.name

class Order(models.Model):

order\_number = models.CharField(max\_length=100)

customer = models.CharField(max\_length=100)

amount = models.DecimalField(max\_digits=10, decimal\_places=2)

def \_\_str\_\_(self):

return self.order\_number

views.py文件中的视图函数示例：

python

from django.shortcuts import render

from .models import Manager

def manager\_list(request):

managers = Manager.objects.all()

return render(request, 'managers/manager\_list.html', {'managers': managers})

def employee\_list(request):

employees = Employee.objects.all()

return render(request, 'employees/employee\_list.html', {'employees': employees})

def department\_list(request):

departments = Department.objects.all()

return render(request, 'departments/department\_list.html', {'departments': departments})

def order\_list(request):

orders = Order.objects.all()

return render(request, 'orders/order\_list.html', {'orders': orders})

模板文件中的HTML代码示例：

manager\_list.html:

html

<!DOCTYPE html>

<html>

<head>

<title>管理人员列表</title>

</head>

<body>

<h1>管理人员列表</h1>

<table class="table table-bordered">

<thead>

<tr>

<th>ID</th>

<th>用户名</th>

<th>密码</th>

<th>重置密码</th>

<th>操作</th>

</tr>

</thead>

<tbody>

{% for obj in queryset %}

<tr>

<td>{{ obj.id }}</td>

<td>{{ obj.username }}</td>

<td>\*\*\*\*\*\*\*\*\*</td>

<td>

<a href="/admin/{{ obj.id}}/reset/">重置密码</a>

</td>

<td>

<a class="btn btn-primary btn-xs" href="/admin/{{ obj.id }}/edit/">编辑</a>

<a class="btn btn-danger btn-xs" href="/admin/{{ obj.id }}/delete/">删除</a>

</td>

</tr>

</table>

</body>

</html>

employee\_list.html:

html

<!DOCTYPE html>

<html>

<head>

<title>企业人员列表</title>

</head>

<body>

<h1>企业人员列表</h1>

<ul>

{% for employee in employees %}

<li>{{ employee.name }} - {{ employee.position }} - {{ employee.department.name }}</li>

{% empty %}

<li>暂无企业人员</li>

{% endfor %}

</ul>

</body>

</html>

department\_list.html:

html

<!DOCTYPE html>

<html>

<head>

<title>部门列表</title>

</head>

<body>

<table class="table table-bordered">  
 <thead>  
  
 <tr>  
 <th>ID</th>  
 <th>名称</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for item in queryset %}  
 <tr>  
 <th>{{ item.id }}</th>  
 <td>{{ item.title }}</td>  
 <td>  
 <a class="btn btn-primary btn-xs" href="/depart/{{ item.id }}/edit/">编辑</a>  
 <a class="btn btn-danger btn-xs" href="/depart/delete/?nid={{ item.id }}">删除</a>  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
</table>

</body>

</html>

order\_list.html:

html

<!DOCTYPE html>

<html>

<head>

<title>订单列表</title>

</head>

<body>

<h1>订单列表</h1>

<ul>

{% for order in orders %}

<li>{{ order.order\_number }} - {{ order.customer }} - {{ order.amount }}</li>

{% empty %}

{% endfor %}

</ul>

</body>

</html>

以上是一个基于Django的员工管理系统的简单示例，可以根据实际需求进行扩展和修改。可以创建更多的模型、视图和模板来实现其他功能，如添加、编辑和删除数据等。

# 4 测试

以下是一个基于Django的企业人员管理系统的测试用例示例:

1. 测试登录功能：

* 输入正确的用户名和密码，点击登录按钮，验证是否成功跳转到企业人员列表页面。
* 输入错误的用户名和密码，点击登录按钮，验证是否显示登录失败的提示信息。

2. 测试企业人员列表页面：

* 验证企业人员列表是否正确显示所有企业人员的基本信息。
* 点击某个企业人员的姓名，验证是否成功跳转到企业人员详情页面。

3. 测试企业人员详情页面：

* 验证企业人员的详细信息是否正确显示。
* 验证企业人员的联系方式是否正确显示。

4. 测试事务页面：

* 验证通知和公告是否正确显示所有公司的标题、内容和发布时间。

5. 测试前后端交互：

* 提交登录表单时，验证是否成功发送POST请求到后端，并返回正确的登录结果。
* 点击企业人员姓名时，验证是否成功发送GET请求到后端，并返回正确的企业人员详情数据。

# 5 总结

基于Django的企业人员管理系统是一个使用MySQL数据库存储数据，并通过HTML+CSS+JavaScript前端设计实现页面的美化和动态效果的系统。该系统主要包括企业人员信息管理、考勤管理、绩效评估、培训管理、通知和公告等功能。

在前端方面，使用HTML+CSS+JavaScript可以实现丰富多样的页面布局和交互效果。HTML用于定义页面的结构，CSS用于美化页面的样式，JavaScript用于实现页面的动态效果和交互行为。

首先，使用HTML可以创建页面的基本结构，包括头部、导航栏、内容区域和底部等。可以使用div、header、nav、section、footer等标签来划分页面的不同部分，使用ul和li标签创建导航栏和列表等。

其次，使用CSS可以对页面进行美化和样式设计。可以设置背景颜色、字体样式、边框样式等，通过选择器和属性来选择和设置页面中的元素。可以使用CSS的盒模型来控制元素的尺寸和布局，使用浮动和定位来实现元素的位置调整。

此外，使用JavaScript可以实现页面的动态效果和交互行为。可以通过DOM操作来修改页面的元素、属性和样式，可以使用事件监听和处理来响应用户的操作，可以使用AJAX技术来实现与后端的数据交互。例如，可以使用JavaScript实现表单验证、数据的动态加载和展示、页面的切换和跳转等功能。

总结起来，基于Django的企业人员管理系统,使用MySQL数据库存储数据，并通过HTML+CSS+JavaScript前端设计实现页面的美化和动态效果。前端的HTML、CSS和JavaScript技术可以灵活地创建页面的结构、样式和交互效果，提升用户体验和系统的可用性。通过合理的布局和设计，可以使页面简洁明了、易于操作，提高企业人员管理的效率和准确性。同时，通过与后端的数据交互，可以实现数据的实时更新和展示，使系统更加实用和可靠。