

**数据库大作业（设计）**

**题目：Steam游戏管理系统**

组名：听取WA声一片

组员：刘诗轩42112204、张傲天42112249、

李鑫煜42112248、沈天祎42112158

学校 陕 西 师 范 大 学

学院 计算机科学学院

指导教师 曹菡

时间 2022年12月14日

目录

[一.概述 3](#_Toc121910010)

[1.1 需求分析 3](#_Toc121910011)

[1.2 设计要求 4](#_Toc121910012)

[1.3 设计内容 4](#_Toc121910013)

[1.4设计特色 5](#_Toc121910014)

[1.5 环境配置 5](#_Toc121910015)

[开发环境： 5](#_Toc121910016)

[运行 5](#_Toc121910017)

[二、概念模型 6](#_Toc121910018)

[1．概念设计 6](#_Toc121910019)

[2．关系图 6](#_Toc121910020)

[3．E-R图 7](#_Toc121910021)

[三、逻辑设计 10](#_Toc121910022)

[四、 需求分析 19](#_Toc121910023)

[4.1系统需求描述 19](#_Toc121910024)

[五、功能展示 20](#_Toc121910025)

[5.1 主页面 20](#_Toc121910026)

[5.2 标签检索页面 20](#_Toc121910027)

[5.3 游戏详情页面 21](#_Toc121910028)

**任务分配表**

|  |  |
| --- | --- |
| 任务 | 被分配人 |
| 关系表的创建 | 刘诗轩 |
| 前端代码 | 张傲天、李鑫煜 |
| 后端代码 | 刘诗轩 |
| 前后台的链接 | 张傲天、刘诗轩 |
| 页面设计 | 张傲天、李鑫煜 |
| ppt 制作 | 沈天祎 |
| ER模型 报告书 | 沈天祎 |
| 成果汇报 | 刘诗轩 |

**STEAM 游戏管理系统**

# 一.概述

## 1.1 需求分析

管理信息系统（MIS）的应用已深入到社会的各行各业，它是信息、软件与科学管理相结合的产物。我们小组发现，现有的MIS往往受限于单一的语种，在特定情况下常常造成不便。我们结合本学期所学，以软件工程的思想为指导，参考Steam等现有的跨国平台，从可行性研究开始，经过系统分析、系统设计、系统实施等阶段的规范开发，针对这一问题开发了多语言的steam游戏管理系统（steam database）。

本题目实现的是网络数据库管理系统，我们选择的是Steam游戏管理平台，通过不同的方式检索自己想要找到的游戏并点击游戏名称查看详细信息。管理员则上传或删改游戏信息，管理自己的游戏数据，并增加和删改不同语种，方便不同国家的人浏览。

## 1.2 设计要求

1、主题明确，软体开发要有创新性，能体现SQL的存储过程、触发器、游标技术等；

2、用高级语言结合MySQL数据库进行开发；

3、写出数据库关系模式及相应的数据库表关系图；

4、绘制steam游戏管理系统相应的系统流程图、E-R图等；

5、完成steam游戏管理系统的前端，包括主界面、标识检索页面、游戏详细信息页面和其各项设计与操作。

6、完成多语言模式的设计和实现

## 1.3 设计内容

本题目实现的是对Steam游戏的管理，我们参考了Steam平台现有的数据和分组方式，针对游戏设计了开发商、发行商以及标签三种检索方式。我能设计了多语言模式，目前填入了英语、简体中文和繁体中文，并预留了其他语言的位置，可以方便后续增加语种。我们设计了前后端系统的完全分离，保障系统的安全性和可用性。

该系统主要包括多方式检索系统、多语言模式系统、游戏信息管理系统、游戏标识管理系统，前后端隔离系统，五个部分，各部分主要功能如下：

多方式检索系统：让用户可以通过开发商、发行商、标签等等方式检索游戏。

多语言模式系统：让用户可以在网页内一键更改语言，让管理员可以在后台添加不同的语言。

游戏信息管理系统：让管理员可以上传和修改游戏信息

游戏标识管理系统：让管理员可以添加和删改游戏的标识，优化用户的检索体验。

前后端隔离系统：用于部署前端和后端，让前端连接后端，调用后端的数据和API。

## 1.4设计特色

设计了可以通过几乎任何方式检索到游戏的检索方式。

设计了多语种模式，可以满足不同国家用户浏览。

## 1.5 环境配置

### 开发环境：

开发语言：前端开发框架为vue，后端开发语言为python,使用的数据库为MySQL

运行环境：python3.7^, MySQL8.0^, Node.js LTS

### 运行

#### 一、后端打开方式

1. 下载python 3.7以上的版本，最好是3.10，并根据安装程序向导完成安装，注意将bin文件夹添加到PATH环境变量。

2. 打开cmd， PowerShell，或者terminal等命令行窗口，然后输入命令pip install virtualenv

3. 然后cd到文件夹，路径看起来是这样的..../SteamDB/Flask

4. 输入virtualenv venv

5.然后cd到文件夹，路径看起来是这样的..../SteamDB/Flask/venv/Scripts

6. 输入 .\activate(Windows系统) 或者 activate(Linux系统等)，如果你看到在命令行最前面多了 (venv) 证明你做对了。

7. 接着cd到misc文件夹，路径看起来是这样的 ..../SteamDB/misc.

8. 输入 pip install -r requirements.txt 然后等待自动安装.

9. cd到Flask文件夹，路径看起来是这样的 ..../SteamDB/Flask.

10. 输入 python -m flask run.

11. 如果你在命令行上看到了 Running on http://127.0.0.1:5000，证明你做对了.

12. 保持这个窗口的打开，你可以把它加入后台，但\*\*\*不要\*\*\*关闭这个进程。

#### 二、前端打开方式

1. SteamDB/Vue目录下打开PowerShell/cmd或是terminal等命令行窗口。

2. 输入命令yarn install

3. 输入命令yarn dev

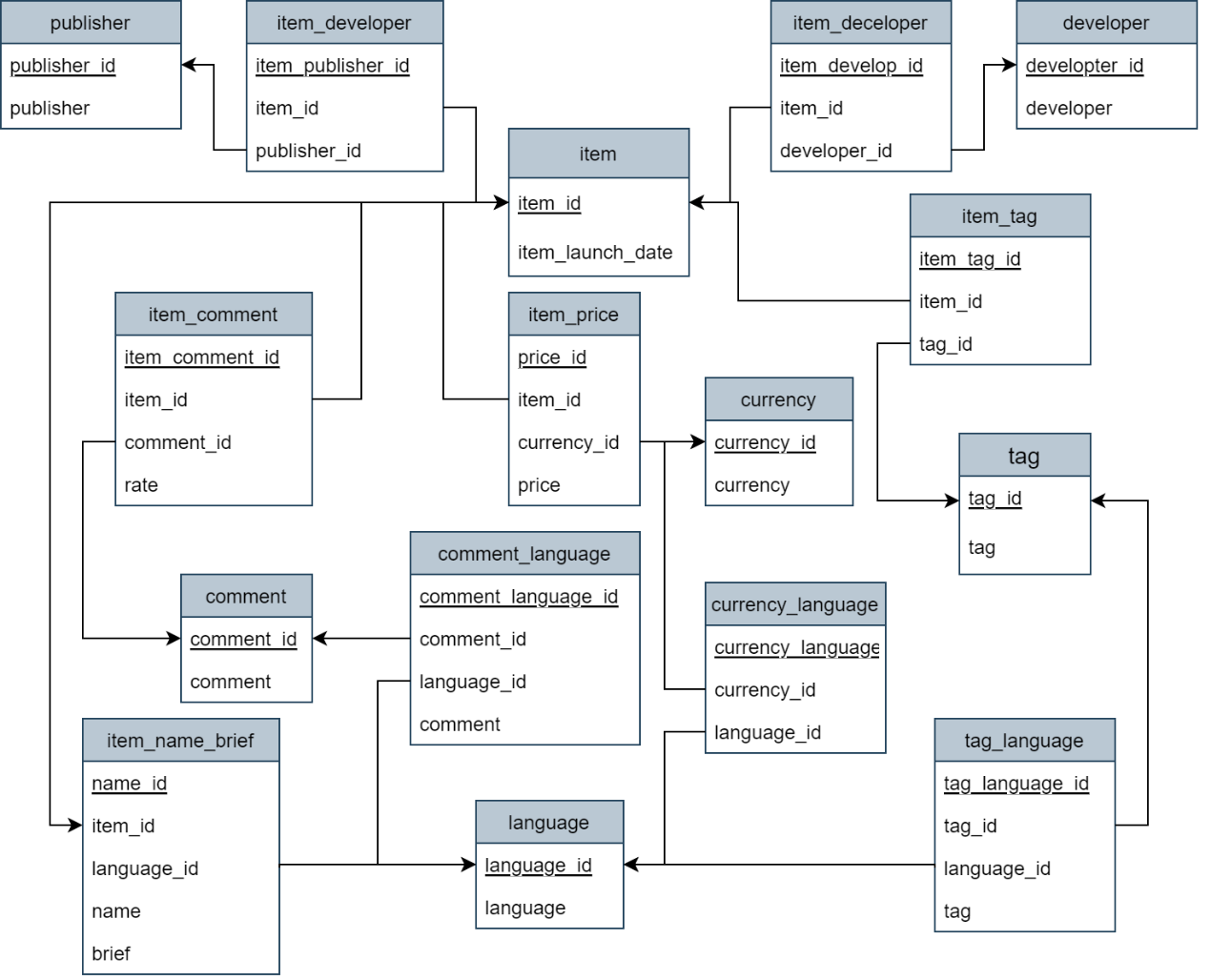
4. 前端页面打开

# 二、概念模型

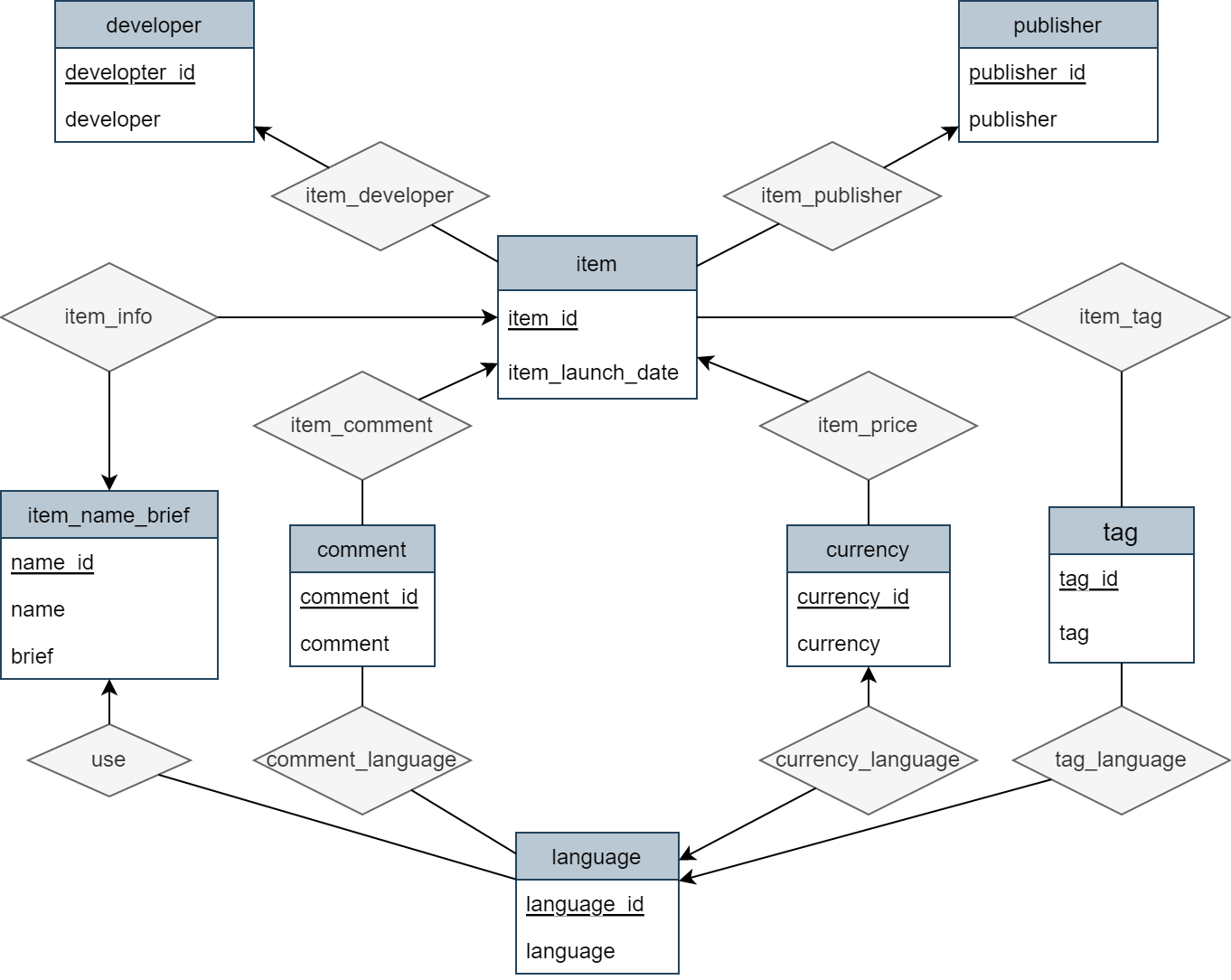
## 1．概念设计

用户分为两类：一般用户和管理员（开发者）。一般用户仅能通过网页使用系统的功能，在多语种的环境下浏览和检索游戏。开发者在能够使用浏览和检索功能的基础上还可以在后端上传和删改游戏信息，管理游戏的检索标识，增删语种，从而对自己的Steam游戏信息进行管理。

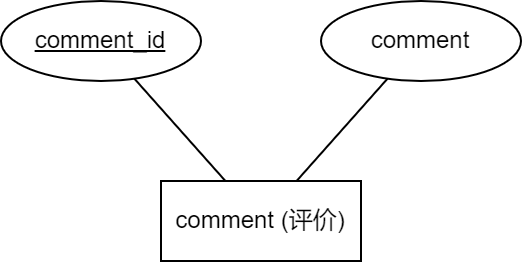
## 2．关系图



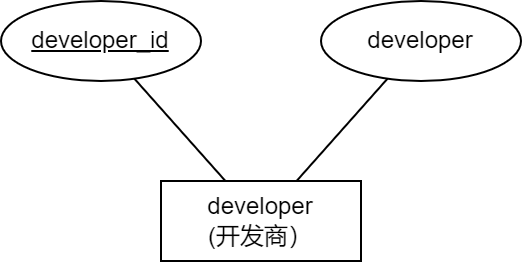
## 3．E-R图



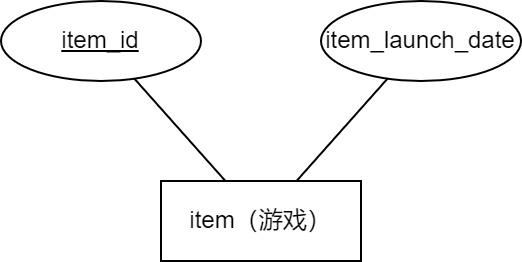
**1.评价表 comment**



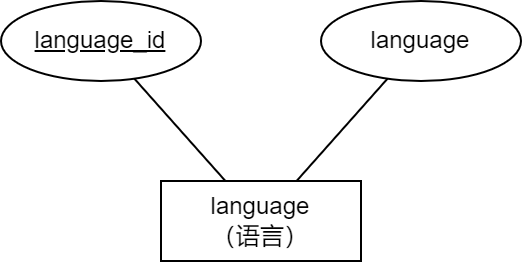
1. **开发商表 developer**



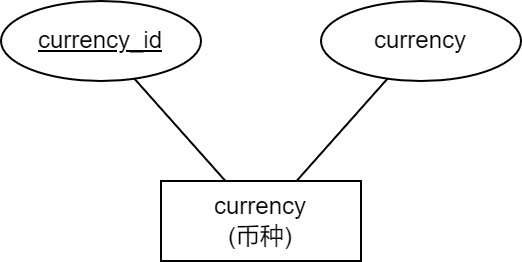
1. **物品表 item**



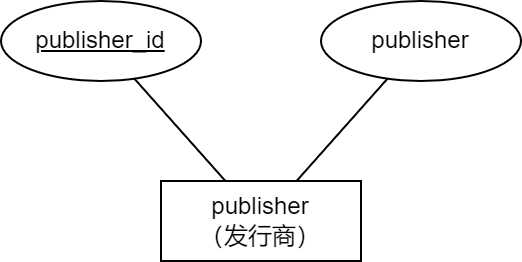
1. **语言表 language**

****

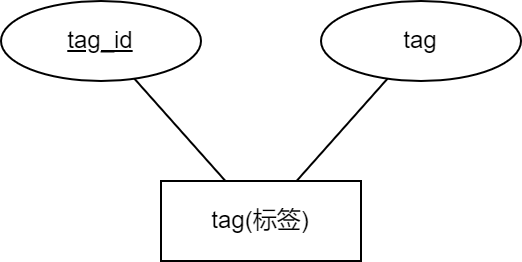
1. **币种表 currency**

****

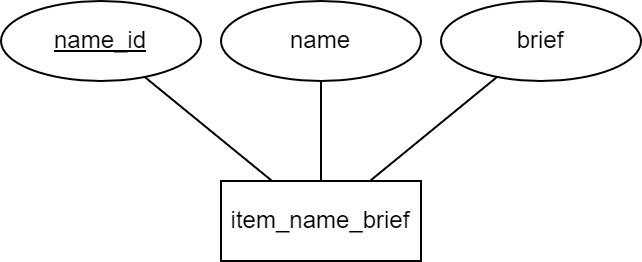
1. **发行商表 publisher**

****

1. **标签表 tag**

****

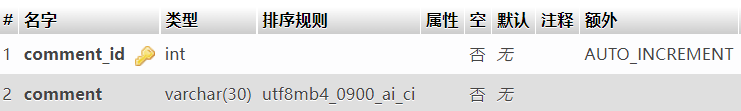
1. **物品信息表 item\_name\_brief**

****

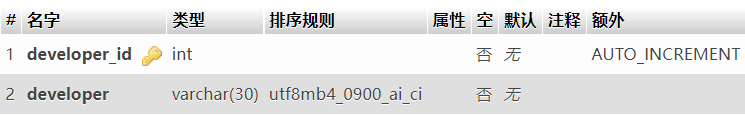
# 三、逻辑设计

共有16个表：

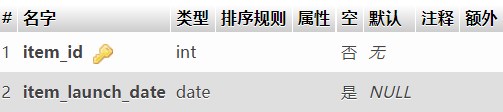
1. **评价表 comment**



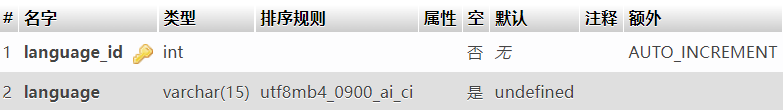
1. **开发商表 developer**



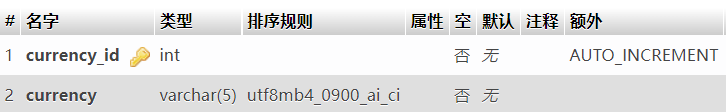
1. **物品表 item**

****

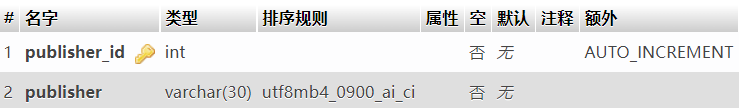
1. **语言表 language**



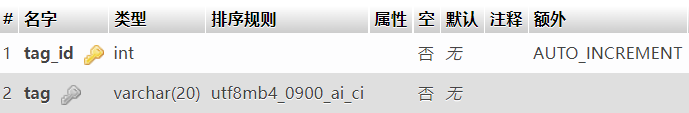
1. **币种表 currency**



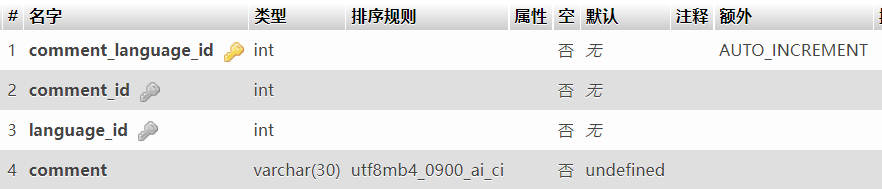
1. **发行商表 publisher**

****

1. **标签表 tag**

****

1. **评论-语言关系表 comment\_language**

****

1. **币种-语言关系表 currency\_language**

****

1. **物品-评论关系表 item\_comment**

****

1. **物品-开发商关系表 item\_developer**

****

1. **物品信息表 item\_name\_brief**

****

1. **物品价格表 item\_price**

****

1. **物品-发行商关系表 item\_publisher**

****

1. **物品-标签关系表 item\_tag**

****

1. **标签-语言关系表 tag\_language**

****

**建表：**

**1.**

create table comment  
(  
 comment\_id int auto\_increment  
 primary key,  
 comment varchar(30) not null  
);

**2**.

create table currency  
(  
 currency\_id int auto\_increment  
 primary key,  
 currency varchar(5) not null  
);

**3.**

create table developer  
(  
 developer\_id int auto\_increment  
 primary key,  
 developer varchar(30) not null  
);

**4.**

create table item  
(  
 item\_id int not null  
 primary key,  
 item\_launch\_date date null  
);

**5.**

create table language  
(  
 language\_id int auto\_increment  
 primary key,  
 language varchar(15) default 'undefined' null  
);

**6.**

create table publisher  
(  
 publisher\_id int auto\_increment  
 primary key,  
 publisher varchar(30) not null  
);

**7.**

create table tag  
(  
 tag\_id int auto\_increment  
 primary key,  
 tag varchar(20) not null,  
 constraint tag\_tag\_uindex  
 unique (tag)  
);

**8.**

create table comment\_language  
(  
 comment\_language\_id int auto\_increment  
 primary key,  
 comment\_id int not null,  
 language\_id int not null,  
 comment varchar(30) default 'undefined' not null,  
 constraint comment\_language\_comment\_null\_fk  
 foreign key (comment\_id) references comment (comment\_id),  
 constraint comment\_language\_language\_null\_fk  
 foreign key (language\_id) references language (language\_id)  
);

**9.**

create table currency\_language  
(  
 currency\_language\_id int auto\_increment  
 primary key,  
 currency\_id int not null,  
 language\_id int not null,  
 constraint currency\_language\_currency\_null\_fk  
 foreign key (currency\_id) references currency (currency\_id),  
 constraint currency\_language\_language\_null\_fk  
 foreign key (language\_id) references language (language\_id)  
);

**10.**

create table item\_comment  
(  
 item\_comment\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 comment\_id int not null,  
 rate float default 0 null,  
 constraint item\_comment\_comment\_null\_fk  
 foreign key (comment\_id) references comment (comment\_id),  
 constraint item\_comment\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id)  
);

**11.**

create table item\_developer  
(  
 item\_develop\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 developer\_id int not null,  
 constraint item\_developer\_id\_developer\_null\_fk  
 foreign key (developer\_id) references developer (developer\_id),  
 constraint item\_developer\_id\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id)  
);

**12.**

create table item\_name\_brief  
(  
 name\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 language\_id int not null,  
 name varchar(50) default 'undefined' null,  
 brief text null,  
 constraint item\_name\_brief\_language\_null\_fk  
 foreign key (language\_id) references language (language\_id),  
 constraint item\_name\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id)  
);

**13.**

create table item\_price  
(  
 price\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 currency\_id int not null,  
 price float default 0 null,  
 constraint item\_price\_currency\_null\_fk  
 foreign key (currency\_id) references currency (currency\_id),  
 constraint item\_price\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id)  
);

**14.**

create table item\_publisher  
(  
 item\_publisher\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 publisher\_id int not null,  
 constraint item\_publisher\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id),  
 constraint item\_publisher\_publisher\_null\_fk  
 foreign key (publisher\_id) references publisher (publisher\_id)  
);

**15.**

create table item\_tag  
(  
 item\_tag\_id int auto\_increment  
 primary key,  
 item\_id int not null,  
 tag\_id int not null,  
 constraint item\_tag\_item\_null\_fk  
 foreign key (item\_id) references item (item\_id),  
 constraint item\_tag\_tag\_null\_fk  
 foreign key (tag\_id) references tag (tag\_id)  
);

**16.**

create table tag\_language  
(  
 tag\_language\_id int auto\_increment  
 primary key,  
 tag\_id int not null,  
 language\_id int not null,  
 tag varchar(20) default 'undefined' null,  
 constraint tag\_language\_language\_null\_fk  
 foreign key (language\_id) references language (language\_id),  
 constraint tag\_language\_tag\_null\_fk  
 foreign key (tag\_id) references tag (tag\_id)  
);

# 四、 需求分析

## 4.1系统需求描述

1. 对于Steam游戏管理系统的总体设计，主要包括如下功能：

**连接远程数据库**：首先建立连接，然后配置数据源。

**整个界面的设计**：主框架，点击按钮分别触发事件。

**前后端分离设计**：前端和后端完全分离，前端通过后端的api调用后端数据。

1. 按照不同标识检索游戏的设计，主要包括如下功能：

**按标签检索**：更具管理员给游戏添加的标签进行检索。

**按开发商检索**：根据不同的开发商对游戏进行检索。

**按发行商检索**：根据不同的发行商对游戏进行检索。

1. 管理员在后端添加和删改游戏信息的设计，主要包括如下功能：

**游戏信息添加**：用于添加游戏条目添加游戏信息。

**游戏信息修改**：管理员可以修改游戏的所有信息。

**游戏信息删除**：用于删除游戏信息。

1. 管理员在后端添加和删改标签的设计，主要包括如下功能：

**游戏信息添加**：用于添加标签。

**游戏信息修改**：管理员可以修改已经设置的所有标签。

**游戏信息删除**：用于删除标签

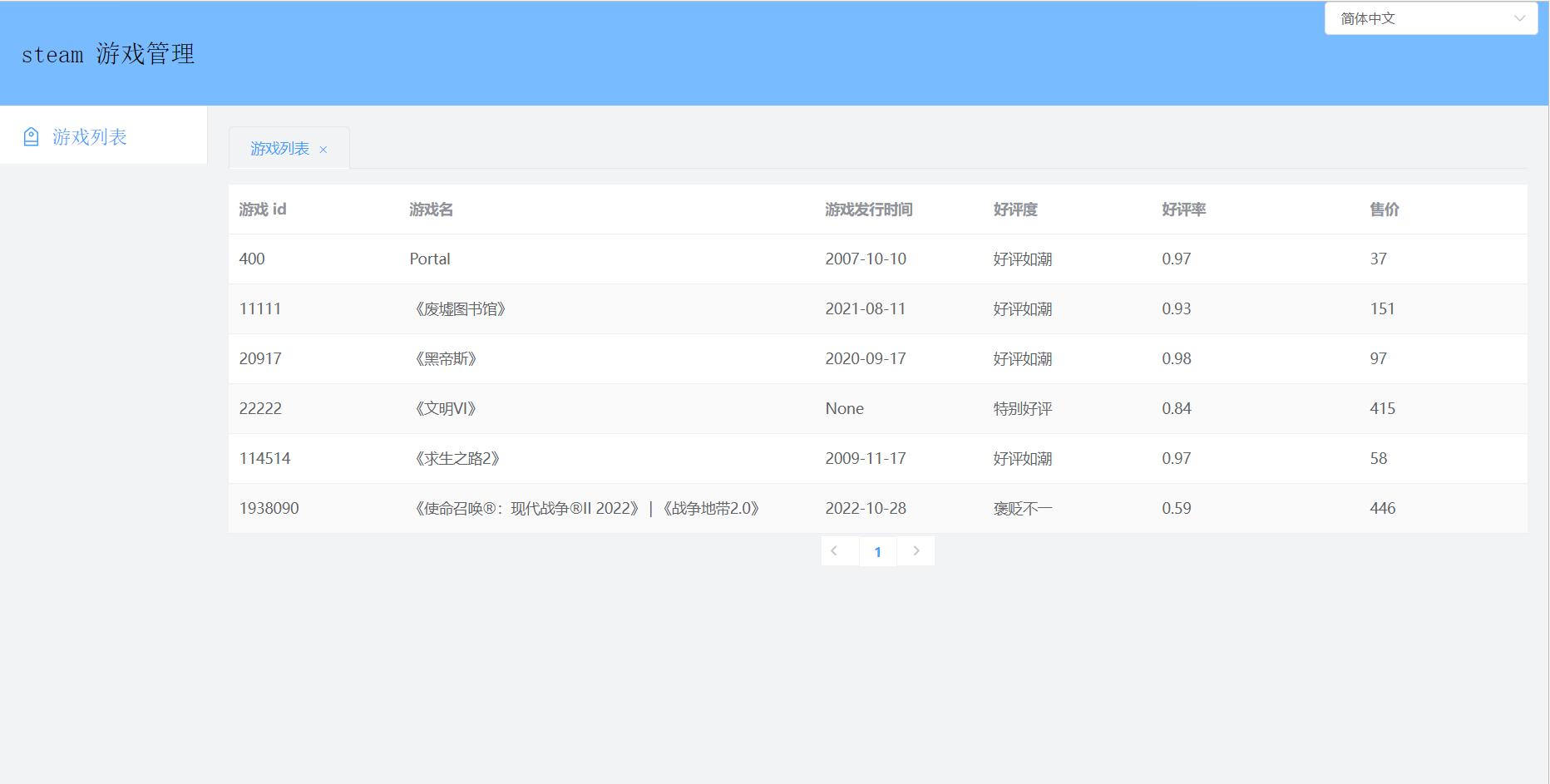
1. 多语言模式的设计，主要包括如下功能：

**上传和删改语种**：管理员可以在数据库中上传和删改不同的语种。

**一键更改语种**：用户可以在网页上一键更改网页所使用的语言。

# 五、功能展示

## 5.1 主页面



5.2 标签检索页面



## 5.3 游戏详情页面

