1.文件结构：

Hex/

├── Hex/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py # 主路由配置

│ └── wsgi.py

└── Hexgame/

├── migrations/ # 数据库迁移文件

├── static/

├── css/

├── image/

├── js/

├── templates/

├── html/

├── models.py # 数据模型

├── serializers.py # 序列化器

├── utils.py # 工具函数

├── views.py # API视图

└── urls.py # 子路由配置

2.前后端交互：

基础URL：/api/

| 端点 | 方法 | 请求数据 | 响应数据 | 说明 |
| --- | --- | --- | --- | --- |
| games/ | POST | mode | {id, board, player\_turn, human\_color, last\_moves, moves\_history, win\_probability,winner,mode,current\_ai\_player | 创建新游戏 |
| games/<game\_id>/ | GET | null | {id, board, player\_turn, human\_color, last\_moves, moves\_history, win\_probability,winner,mode,current\_ai\_player} | 获取游戏状态 |
| games/<game\_id>/move/ | POST | {x, y} | 同GET响应 | 人类落子 |
| games/<game\_id>/ai\_move/ | POST | null | 同GET响应 | 请求AI落子 |
| games/<game\_id>/undo/ | POST | null | 同GET响应 | 悔棋 |
| games/<game\_id>/restart/ | POST | null | 同GET响应 | 重新开始当前游戏 |
| ai\_games/ | POST | null | {id, board, player\_turn, human\_color, last\_moves,win\_probability, winner} | 创建AI对战 |
| ai\_games/<id>/ | GET | null | 同GET响应 | 获取AI对战状态 |

// 通用响应结构

{

"id": number, // 游戏ID

"mode": "HUMAN\_AI" | "AI\_AI", // 游戏模式

"board": number[][], // 11x11矩阵 (0:空, 1:红方, -1:蓝方)

"player\_turn": "human" | "AI" | "AI\_1" | "AI\_2", // 当前回合玩家

"human\_color": "red" | "blue", // 人类玩家颜色（HUMAN\_AI模式有效）

"last\_moves": number[][], // 11x11矩阵（记录落子顺序）

"winner": null | "human" | "AI" | "AI\_1" | "AI\_2",

"win\_probability": { // 实时胜率

"human": number, // 0~1（HUMAN\_AI模式）

"AI": number, // HUMAN\_AI模式

"AI\_1": number, // AI\_AI模式

"AI\_2": number // AI\_AI模式

},

"moves\_history": [ // 历史动作

{

"player": string, // 玩家标识

"x": number, // 0~10

"y": number,

"color": "red" | "blue" // 实际显示颜色

}

]

}

1. Django框架运行指令：  
   在terminal终端输入 python manage.py runserver