兴趣开放实验室

OpenLab

2025 项目复试题

By OpenAdmin

2025 年 10 月 20 日

项目考核说明

注意　答题前请认真阅读此说明，并按照说明规定的方式答题和提交

1. 除第一题规定提交格式外，请在以下 4 道大题中（Git 题除外）**任选一道**完成。我们也鼓励有能力的同学完成其中的多道。如果完成多道题目，计分时按照每道题目的得分从高到低排序，总分为「**最高分题目得分** +50%× **其它题目得分**」。
2. 项目考核时间自 2025 年 10 月 21 日起，至 2025 年 11 月 7 日止。最终的展示形式为现场演示和简易答辩，根据完成度与展示效果综合评分。即使某道题目未完成，也欢迎你跟我们聊聊解决问题过程中的心路历程，我们也会将你的认真程度计入评分。
3. 问题排序与难度无关，每道题侧重的是不同方面，你可以任选自己感兴趣的方面回答。如果感觉某道题目太难，可以尝试在 b 站等网站上搜索题目相关技术，或是查看题目附带的教程或提示。
4. 如果关于题目内容有任何问题，欢迎在「考核群」里提问（公平起见，请尽量不要单独联系出题人）。

祝你考核顺利，玩的开心!

1 Git 与 GitHub

出题人：Criheacy、Qie、Qer

与笔试题中的第一题类似，如果你能按本题的要求提交其余题目的代码，就可以获得额外的加分。

1.1 题目要求

Git 是一个非常常用的版本管理工具 (VCS, Version Control System)，它可以帮助你备份防止代码丢失，可以让你“撤销”到这份程序的上一个版本，还可以在多人协同开发中协调进度；提前学习这项工具的使用可以极大提升开发效率。

而 GitHub 是一个代码的在线仓库。它有点像是百度网盘之类的工具，不过它是专门为存储代码设计的。结合 Git 工具的使用，你可以在网上备份代码文件，也可以随时浏览项目的历史版本。

本题中，你需要了解并掌握它们的基本用法。详细计分规则如下：

+10%	在本地建立若干个 Git 仓库，用来储存每道题目的代码文件。
+15%	建立一个 GitHub 账号，创建若干个**公开仓库** (Public Repository) 用来储存每道题目的代码文件。
+20%	在上述基础上，添加 `README.md` 文件用于简介、`.gitignore` 文件用于忽略一些不应该被提交的文件（比如相关环境、大量的数据集等，一般来说仓库大小应该在 `10MB` 以下）；有良好的文件目录结构（比如代码放在 `/src` 文件夹下、资源放在 `/res` 文件夹下等）。
+30%	在上述基础上，代码文件是使用 `Git` 工具提交的，合理管理 `Git` 的提交节点和分支结构，能看到代码的历史版本。

1.2 参考资料

1. https://www.ruanyifeng.com/blog/2018/10/git-internals.html
2. https://docs.github.com/cn/get-started/quickstart/create-a-repo
3. https://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html

1.3 注意事项

1. Github 因为特殊原因，有可能无法访问，在这种情况下，你可以选择 Gitee；但 Gitee 设置公开仓库需要人工审核，所以我们依然建议使用魔法绕开限制使用 Github。

2. 关于 Git 的教程在网上非常多，不必局限于我们所给的参考资料。

3. 可视化管理 Git 的工具非常多，合理的利用这些工具可以帮助你提高使用 Git 的效率。

2 让我康康!

出题人：crest

今年的 10 月如期而至，新学期，新生活，闪动大学的一切都在刺激着小 C。在 closelab 的纳新宣讲中，小 C 更是领会到了 web 技术的魅力，于是它决定深入学习 web 前端，感受创造的快乐。

2.1 认识新朋友

进入新的班级，小 C 想交到志同道合的新朋友，但是小 C 有些社恐，对完全陌生的人总是有很强的戒备心。小 C 想大家要是有个高效认识彼此的途径就好了，此时大 C 给小 C 提建议说个人网站是个不错的展示自己的方法。于是小 C 决定搭建他自己的个人网站，你也一起来吧!

2.1.1 本题要求

基本功能：(10 分)

1. 在恰当的位置展示你的昵称、邮箱、爱好等，也可以当作个人博客来搭建（无硬性要求），总之内容想加什么加什么
2. 不要过于简陋（谁家个人网站只有两行字啊喂）

2.1.2 加分项

利用 github pages、vercel 等静态网页部署工具，让所有人能访问你的个人网站（也可以部署在服务器上）(15 分)

2.2 电动车去哪了 ⁉

小 C 是早八苦手，但看到闪动大学有共享电动车，终于松了一口气。可实际上每到早八哪里也找不到车，这令小 C 十分苦恼。直到小 C 注意到了 “地图找车” 的功能，他才发现那么多广告中竟然藏着如此有用的功能。于是小 C 立志要让大家看看无广告的 ui 有多么简洁，请你帮帮他。

2.2.1 本题要求

基本功能：(30 分)

1. 分别展示停车点的信息（至少包括名称、详细信息、中心点）和车辆的信息（包括编号、位置、电量）
2. 计算车辆到停车点的距离，距离若小于 10m 则算作在该停车点中。最后在停车点的信息后面追加“车辆数”字段
3. UI 风格统一，页面干净美观，没有广告）

2.2.2 加分项

1. 搭建简单后端接口服务，而不是将数据写死在前端（10 分）
2. 应用响应式设计，让手机浏览器也能轻松使用网页（15 分）
3. 能筛选出距离你的设备较近的停车点（部分浏览器由于权限限制无法获取设备信息，有代码即可酌情给分）（10 分）

2.3 一看就在这儿！（基于第 2 题实现的管理页面，扩展出地图页面）

在开发中，我们有时会遇到与“第三方服务 API”（Third-party Service API）相关的任务。这类任务的核心是“遵循第三方服务的规范，将外部功能快速整合到自身网页中”，常见形式包括：

1. 通过 <script src=” 第三方 SDK 地址”></script> 引入 SDK（如嵌入百度地图、高德地图，示例代码 googlemap-embeded.html 在附件中）
2. 通过 <iframe> 嵌入第三方页面（如嵌入 B 站视频，示例代码 bili-embeded.html 在附件中）

2.3.1 本题要求：（30 分）

本题需你学习“第一种形式（SDK 引入）”，选择地图服务 SDK（如百度地图 SDK、Leaflet），完成地图嵌入与数据可视化功能。基本功能：

1. 在该页面成功嵌入地图底图（各种底图都可以，比如通过 Leaflet 或 OpenLayers 加载 OpenStreetMap（osm）的地图、使用百度地图 SDK）
2. 能在地图上用多边形（polygon）和点（point）分别绘制所有停车点的所在区域和车辆位置

2.3.2 加分项

1. 能与停车点进行交互：点击某停车点或车辆后，能在恰当的位置显示其名称等信息（提示：可以使用地图 SDK 的 InfoWindow 组件）（10 分）

2.4 验收要求

1. 随机提问考察代码原理、实现逻辑等方面问题
2. 验收时需要提交所写代码，若使用 GitHub 会酌情进行考察
3. 超出要求范围的实现会酌情加分。

2.5 Tips

2.5.1 学习资源：

1. 什么是第三方 API(https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Extensions/Client-side_APIs/Third_party_APIs)
2. 百度地图开放平台（https://lbsyun.baidu.com/）
3. 百度地图使用方法（https://juejin.cn/post/6991479115643617310）

2.5.2 注意事项：

1. 本题题目可以不使用服务器，可以直接使用 live server（vscode 插件）实现对页面的挂载。
2. 本题所有数据自拟即可，格式可以参考附件中给出的数据。
3. 请善用搜索引擎和 AI 问答，b 站也有不少教程可以参考，但不要 vibe coding。

3 小 E 的游戏天原

出题人：CWling，wb

“前面的区域现在快来探索吧!”小 E 在古神的呓语中醒来，只见自己被五个传送法阵围了起来。就在好奇心驱使小 E 向前踏出第一步之前，一个传送法阵坍缩了，周围的空间也随之扭曲起来，小 E 急忙向着某一个传送法阵奔去……

从 3.2 - 3.5 的四个题目中任意挑选一个，依照任务要求完成游戏项目的相应功能。

3.1 勇士传说 - Unity 启航

3.1.1 步骤

1. 安装 Unity 引擎和代码编辑器：
   https://www.bilibili.com/video/BV1mL411o77x/
2. 参考资源：小 E 背包里的一些有意思的小玩意儿：
   https://www.bilibili.com/video/BV1PL4y1e7hy/
   https://www.bilibili.com/video/BV1ZvbDzaE6X/

3.2 时空守护者之废土探索 - Unity 3D

小 E 通过传送法阵，惊讶地发现自己进入了现实的夹缝。这里存在着无数个承载着重要历史片段的“时空碎片”。小 E 窥探着这些碎片，或为一座饱经沧桑的古堡，或为一座诡谲的迷宫，或为一座埋葬有秘密的疗养院，或为一个平凡却充满故事的街角……

然而，一股未知的干扰波正在侵蚀着这里，小 E 毅然担起了时空守护者的重任。他决定从重建一处时空碎片中的小小世界开始……

3.2.1 探索须知

“一片无人踏足的新世界，此处的旧日支配者已然沉睡，而小 E 你，则是这个世界的新神。”小 C 这样说道。

“不必顾虑太多，带上你的 Unity 3D，自由发挥就行。不过，你可以看看这些，或许能带来一些灵感。”

- https://www.bilibili.com/video/BV18T411P7Ec/
- https://www.bilibili.com/video/BV1Zx4y147k5/
- https://www.bilibili.com/video/BV1GJ411x7h7/
- https://assetstore-fallback.unity.com/3d

3.2.2 重建提示

sparkles 为小 E 点亮一方心仪的世界。

基础 (35 分)

- 完善游戏人物；
- 实现人物控制（选做：若有多层建筑，需实现平滑的上下楼交互）；
- 进行简单的场景交互；
- 点亮世界——放置灯光、调整光影、给小 E 一个手电筒耍耍。

进阶 (55 分)

- 为交互操作添加音效、并为世界添加风格合适的背景音乐；
- 实现交互的动画呈现；
- 实现动态敌人（选做：实现一场酣畅淋漓的战斗）。

特性 (10 分) 显示迷你地图，标识小 E 位置。

3.3 视觉小说 - Ciallo

“这是一个因你而在的故事，”热心同学小 C 表示，“小 E 将在旅途中读到何种故事呢?”

3.3.1 写在前面

本题可以参考一个基于 Unity 的教程：
https://www.bilibili.com/video/BV1CKZmY4Eau

3.3.2 功能要求

基础 (30 分)

1. 对话框和背景图（5 分）
2. 文案的储存和读取（5 分）
3. 更多效果（包括文字打字机效果、头像效果、音频植入、音乐效果）（5 分）
4. 立绘随对话出现与消失和背景图的切换（10 分）
5. 立绘随剧情进行简单移动（5 分）

进阶 (70 分)

1. 选择与分支功能 + 返回主干的文件跳转功能（5 分 + 5 分）
2. 自动播放功能和跳过功能（5 分）
3. 基础的存档功能 + 存档的删除与覆盖（15 分 + 5 分）
4. 初始界面（5 分）
5. 完整的历史记录功能（10 分）
6. CG 画廊功能（5 分）
7. 输入内容（主角姓名）（5 分）
8. 画面设置（10 分）

附加题 (20 分) 自由发挥进行功能拓展，可考虑以下方向：

- 多语言切换
- 本地化
- 开场动画
- 好感度系统
- 场景切换

3.4 平面 2D 游戏 - 新的开始

小 E 身处无垠草原，白云点缀蓝天——一切都是像素绘制。小 E 低头看自己的双手，它们变成了像素方块。小 E 用变成一个像素点的大拇指摁了摁手腕上戴着的像素手表——boom 地一声，小 E 为自己换了一套背带裤。

小 E 看着空中浮动的像素云朵，耳边回响复古电子乐，一群像素蚂蚁在地上爬行，它们摞出了几个字—— “欢迎来到小 E 的游戏天原”。

3.4.1 写在前面

本题可以参考一个基于 Unity 的仿《星露谷物语》农场游戏教程：
https://www.bilibili.com/video/BV1TC4y1B7VZ?p=14

也可以根据自己的兴趣选择钻研战斗地牢类型或解密剧情类型。

3.4.2 功能要求

基础 (30 分)

- 完成控制角色移动（5 分）
- 完成基本的瓦片地图绘制（5 分）
- 完成瓦片地图的分层（10 分）
- 完成场景和角色的碰撞（10 分）

进阶 (60 分)

- 拥有基本的 UI 设置（15 分）
- 拾取物品并存入背包（15 分）
- 整理背包功能（20 分）
- 丢弃物品功能（10 分）

特色性内容 (10 分) 锄地功能及动画实现（10 分）。若选其他课程类型，也可设计特色功能以获得相同加分。

附加 (20 分) 在完成上述功能基础上，自行添加课程未涉及的功能，例如：

- 场景转换（如农场切换到屋内、洞穴、城镇）
- 可修改场景的角色行为（如砍树后取消碰撞）
- 战斗系统和多个 NPC 角色

3.5 深渊回响 - 卡牌游戏

小 E 在传送的过程中被一座机械式古文明遗迹吸引，失控能量构成的“镜像体”困住了小 E，令小 E 不得不停下脚步，在这个虚拟空间中模拟探索、战斗并最终破解遗迹核心，继续踏上旅途。

小 E 又一次来到了“深渊回响”之下，这一次，他会……

3.5.1 核心目标

制作一款可运行的单机回合制卡牌游戏。

3.5.2 功能要求

基础 (50 分)

- 创建可被玩家使用的卡牌 UI 系统，实现优雅的卡牌布局；
- 实现模块化的卡牌效果；
- 创建敌人系统及基础行为逻辑；
- 创建简单的玩家系统；
- 实现目标系统。

进阶 (35 分)

- 实现每回合自动回复的能量系统；
- 能量反馈与动画；
- 古代遗物——实现永久强化系统。

特性 (15 分) sparkles “这一次，让我们来尝试一些新东西。”若觉得这次探险之旅还不够有意思，可尝试添加更多功能展示能力，如：

- 多层遗迹探索；
- 存档与读档；
- 卡牌强化系统。

3.5.3 后话

可参考的 Unity 教程：
https://space.bilibili.com/173991951/lists/5471717

4 HarmonyOS 开发初探

出题人：Handwer

写在前面

- 第一题和第二题是**准备工作**，用于创建新工程并熟悉预览器与模拟器的使用。本次开发无需真机，没有华为设备亦可完成。
- 从第三题开始正式进入鸿蒙开发学习。若你已有其他领域开发经验，且认为视频课程效率较低，可选择自学，但须确保覆盖所有知识点。

  推荐替代学习资源： B 站黑马程序员。
- 若认为第四题的 Codelabs 步骤繁琐，可自行实现相同页面效果，但验收时需清晰阐述页面布局逻辑。
- 欢迎使用 AI 工具辅助学习与开发。验收时如能说明使用场景并展示 Prompt，可作为**加分项**。

出题团队已验证的 AI 工具：

- **Claude Sonnet 4/4.5**（最佳选择）
- ChatGPT 5（可用）
- Qwen 3（可用）
- ChatGLM-4.6（可用）
- DevEco Studio 自带 Code Genie（不推荐）
- 文心一言 4.5 Turbo（不推荐）
- DeepSeek V3/R1（不可用）

题外话：出题团队现已全面在日常项目中引入 *Claude Sonnet 4* 作为 *AI* 辅助工具。

4.1 开发入门 · 欢迎启程

4.1.1 步骤

1. 在华为开发者联盟官网注册并实名认证开发者账号。
2. 下载并安装 DevEco Studio 6.0，创建一个空白工程。

3. 启动 Previewer（预览器），看到“Hello World”即可。

参考文档：应用开发导读 · 快速入门
推荐视频：华为开发者学堂课程

4.2 第一行代码 · Hello SDU

4.2.1 步骤

1. 打开创建的工程，将“Hello World”改为“Hello SDU”。
2. 在 `Text("Hello SDU")` 后添加 `.fontColor(Color.Red)`：

```
Text("Hello SDU")
.fontColor(Color.Red)
```

注意：鸿蒙中单双引号都可用于字符串，且没有“字符”概念。所有代码必须用英文输入法和符号。

3. 打开预览器，确认“Hello SDU”显示为红色。
4. 参考课程【使用模拟器运行应用】，将应用运行到模拟器中。

视频：使用模拟器运行应用

4.3 构建第一个待办应用 · 从任务出发

4.3.1 必做任务

观看【ArkUI 框架介绍】课程，学习并掌握以下内容：

- 声明式开发范式
- 常用组件及功能
- 布局能力与交互统一

观看【声明式 UI 语法】，学习并掌握以下内容：

- 组件格式、生命周期、属性
- 容器组件与常见布局
- 组件状态与交互

- 列表数据 ForEach

阅读【UI 范式】文档，学习并掌握以下内容：

- 基础语法描述
- 声明式 UI 描述
- 创建自定义组件（了解格式即可，无需深入研究）
- `if/else` 控制渲染

完成【Codelabs：待办列表】作为验收成果。

4.3.2 额外加分项

- 阅读【UI 范式】更多内容；
- 尝试独立完成【案例：待办列表】，或给该项目添加你喜欢的更多功能作为验收成果；
- 完成【闯关习题】。

视频课程：待办应用开发课程。

4.4 布局实践 · 构建登录页面

4.4.1 步骤

1. 复习【声明式 UI 语法】；
2. 观看【布局你的页面】，学习并掌握以下内容：
   - 线性布局
   - 层叠布局（Stack）
   - `Column` / `Row` 及其属性
3. 阅读【布局概述】，学习并掌握以下内容：
   - 组件内容区
   - margin、border、padding
4. 观看【组件简单页面】课程，并阅读四个相关文档，掌握常见组件用法。

完成【案例：页面与数据】作为验收成果。
视频课程：布局与页面构建。

4.5 额外加分项 · 构建可交互的待办应用

4.5.1 进阶任务

阅读状态管理概述，了解状态管理机制。
阅读【状态管理（V1)】，理解以下功能的用法（不要求掌握原理)：

- `@State`、`@Prop`、`@Link`
- `@Provide` / `@Consume`
- LocalStorage 与 AppStorage
- （选学）`@Observed` / `@ObjectLink`

阅读 MVVM 模式，了解架构设计。
学习组件 `TextInput` 与 `Button` 的用法。
回到【Codelabs：待办列表】代码，实现以下功能：

- 添加输入框，输入待办事项标题；
- 添加按钮，点击后动态添加待办事项；
- （可选）实现数据持久化。

这将是你的验收成果。

5 Eternal Band

出题人：wh，Darkstars

组乐队从来不是一件容易的事情，更何况是组一辈子乐队。
在某条世界线上，演出非常顺利，Ob 一串字母大人决定做点什么纪念武道馆表演的成功。
于是她找到了你，一位写程序的好手，来做这一点小小的工作。

5.1 题目描述（满分 100 分）

5.1.1 搜索提问之章

就像乐队中有着性格各异的成员，每个人或许都有着或多或少的缺点和不足一样，我们编写的程序中也躲不开各种各样的问题和报错，所以我们得向千早爱音女士学习，把大家（代码们）团结起来，逐渐克服各种困难和不足——在这个过程中，学会搜索和提问是十分必要的。

一般来说，解决问题的流程如下：

(1) 检查操作流程是否出错；

(2) 掌握并复制报错内容，并在搜索引擎进行搜索；

- 推荐的搜索引擎：Bing，Google (如有条件)；

(3) 如果你没有找到有用的信息，可以考虑带着更详细的问题描述去咨询 Deepseek，ChatGPT 等大语言模型，根据它们的分析，**仔细甄别**后上手修改；

(4) 如果还是没有解决，你也可以去一些更加专业的平台进行检索，这种平台通常会对你的英语水平有一定要求（当然我们可以借助大模型或翻译平台）；

- 推荐：stackoverflow，博客园，内容质量相对较高且基本无广告；
- 谨慎选择：知乎，CSDN，内容良莠不齐且付费内容较多，注意分辨；

(5) 如果依旧没有解决，可以在技术社团等群聊中提问并给出详细的日志/截图/问题说明，希望路过的大佬予以解答；

(6) 如果问题仍然没有解决，请考虑换一种方案，或者 睡一觉以求问题自己消失 （不是）。

5.1.2 配置环境之章

题目要求（额外加分 15 分）

在历经无数的磨炼之后，一支好的乐队才能问鼎武道馆，除了要燃烧着青春与热血外，一个舒适的，供大家表现的舞台也至关重要。所以我们可以尝试为自己搭建一个这样的”舞台”，供自己在里面”大显身手”。

工程上大部分后端服务都是跑在 Linux 环境下，因此，掌握在该环境下编程的技能十分必要。

考虑到大部分新生对于 Linux 系统的了解有限，所以本题**可选**在 Linux 环境下完成，但如果你成功配置好相关环境，将会获得一个**额外的加分**，同时加深对操作系统的了解。在大学的未来几年以至工作后，这个环境都将非常有用。

配置一个 Linux 虚拟机环境并不难，我们更关注的是性能损失是否理想。

最小化性能损失的方法无疑是双系统，但安装过程及其不友好，这里建议有一定基础的同学尝试；但同时，它也是最优秀的解决方案——这意味着你同时拥有了图形化界面 + 低延迟 + 高性能，等大三图形学实验的时候你会再次感受到它的伟大。

如果不强制要求图形化界面的话，我们有一个更加简单的方案：基于 Windows 的 Linux 子系统：Wsl。通常我们会使用 Wsl+Unbuntu 22.04+Vscode 的 Remote 插件来获得更好的编程体验。

当然，你也可以选择购买一个 Linux 云服务器并使用 Vscode 进行 SSH 远程连接，这是最简单方便的方法，实际使用体验和 WSL 差不多，但这种方案需要你付出三十多元的高昂代价——更直观的说，你会失去一只疯狂星期四的秘汁全鸡加两瓶冰镇的可口可乐。

额外加分项（15 分）

(1) 配置出一个能够正常运行的 linux 系统并学会使用简单的终端指令。(10 分)

(2) 配置 Linux 下的 Docker 环境，并学会使用 Docker 对环境进行隔离。(5 分)

附录

配置 Wsl

Vscode+SSH 进行服务器远程连接

Windows11 + Linux 双系统安装教程

Ubuntu 上安装 Docker

5.1.3 功能实现之章

题目描述

终于到了这个时刻！你需要借助 Python、HTTP 和数据库的一点儿基础知识来帮助 *Oblivionis* 完成她的小任务，接下来的内容将会十分轻松愉快。

架构说明

为了方便大一新同学们入门，我们提供了一个简易的辅助框架供大家使用，后端部分使用**FastAPI**，这一基于 Python，快捷而又简单直观的框架进行开发，它会提供新手友好的交互式文档，辅助你的调试工作。

当然，本题**不限制语言/框架**。给出的辅助框架中已经包含前端 html 文件，使用 vscode 的 Live Server 插件可以很方便地渲染。如果使用其他框架，前端只需最基础的可视化即可，不作额外加减分。

关于 python 环境的配置：强烈建议使用虚拟环境，方便进行依赖管理。

Listing 1: 项目结构

```
项目根目录/
    frontend/         # 前端主文件夹
        index.html    # 主页（乐队查询）
        band.html     # 乐队信息展示
        music.html    # 音乐信息展示
    backend/          # 后端主文件夹
        main          # 主函数
        models/       # 实体类层
            bangdream_models.py
        services/     # 业务逻辑层
            file_manager.py
            db_manager.py
        routers/      # 接口层
            band_with_file.py
            band_with_db.py

    test_async/ # 测试同步异步
        sync.py # 同步代码
        async.py # 异步代码
    data/           # 数据文件
        band_info.json # 乐队信息数据
        song_data.json # 歌曲信息数据
    前端接口需求文档.md # 描述接口的文档
    Readme.md # 说明文档
```

1. 数据模型层 (Models)：定义数据结构和验证规则。例如，一个用户模型：

Listing 2: 用户模型示例

```
from pydantic import BaseModel
class UserCreate(BaseModel):
```

```
    username: str
    email: str
    password: str
```

什么是类？类是创建对象的蓝图，它定义了对象拥有的属性（数据）和方法（行为）。上面的 `UserCreate` 类规定了一个用户注册时必须提供 `username`, `email`, `password` 这三个字符串类型的属性。

2. 业务逻辑层 (Services): 封装核心业务逻辑，如数据的增删改查（CRUD）。这一层与数据源（文件或数据库）交互。

   为什么分层？将业务逻辑与接口分离，使得代码更易维护、测试和复用。例如，`file_manager` 和 `db_manager` 提供了相同的功能接口，但底层实现不同，上层路由可以灵活切换。

3. 路由层 (Routers): 接收 HTTP 请求，解析参数，调用 Service 层处理，并格式化返回响应。

题目要求（90 分）

注意：对于所有需要用到的 API 接口，都会在前端接口需求文档 `.md` 中说明需求。同时不必局限于框架给出的方法，你可以自由增加需要的的函数。

（1）“首先，乐队要有自己的主页。”——*Oblivionis* 如是说。（10 分）

具体而言，你需要补全文档中的”乐队相关接口”所对应的部分在 `file_manager.py` 与 `band_with_file.py` 中的实现，使得我们可以通过乐队名称（如 `name=MyGO!!!!!`）获对应的乐队的数据。

我们提供了一个 `band_info.json` 文件作为数据源，形如：

Listing 3: 乐队信息数据示例

```
[
  {
    "id": 1,
    "name": "MyGO!!!!!",
    "description":
        "迷失自我，但却向前。她们以充满情感的摇滚乐，表达年轻人的迷茫与坚定。",
    "created_at": "2025-10-21T00:05:15.094343"
  },
  {
    "id": 2,
    "name": "Ave Mujica",
```

```
    "description":
        "虚伪的假面，真实的自我。这是一个神秘且充满戏剧性的乐团，每个成员都带着面具。",
    "created_at": "2025-10-21T00:05:15.094357"
  },
  {
    "id": 3,
    "name": "Poppin'Party",
    "description": "充满活力的流行摇滚乐队，用音乐传递快乐和正能量。",
    "created_at": "2025-10-21T00:05:15.094358"
  }
]
```

补全实现之后,我们应该可以使用 `GET http://0.0.0.0:8000/api/bands?name=MyGO!!!!!` 来获取 mygo 的乐队信息。当不带参数查询的时候，默认获取所有乐队信息。

(2) 歌いましょう鳴らしましょう (10 分)

Doloris 希望网站能展示对应乐队的歌曲信息，但歌曲更新太频繁，如果跟乐队信息一样用一个文件手动维护的话，未免太麻烦了。

因此 Doloris 希望你能补全文档中的”歌曲相关接口”所对应的部分在 `file_manager.py` 与 `band_with_file.py` 中的实现，使得我们可以通过操作 API 对乐曲信息实现增删查改。

乐曲信息包括曲名、作者、歌词，

- 不限制存储文档格式及类型,实现功能即可。可以参考给出的 `song_data.json` 中歌曲的格式。
- 当然，由于可能出现想好了名字但没有歌词，或者大家共同创作的情况，所以除了歌曲名、所属乐队外，作者、歌词字段可以为空。

(3) ようこそ AveMujica の世界へ (30 分)

将数据存放在一个简单的 txt 里其实是很危险的，不仅数据查询和操作效率低，没有回滚止损手段，并发较大时还可能出现写入冲突。因此，你需要学会使用数据库，并完成以下任务：

1. 完善 `db_manager.py` 中的相关函数，以及 `band_with_db.py` 中的对应接口。框架中提供的函数基于 SQLite 实现，可以使用其他数据库重写该部分，保证功能实现正常即可。(20 分)
2. 学习在终端中与数据库进行基础交互操作，例如：

- 创建一张表
- 查询表中的记录
- 插入单条数据
- 更新单条数据

掌握常用基础操作即可。(5 分)

3. 尝试编写一个自动化脚本或使用其他方案，批量插入随机生成的数据，以便进行功能测试。(5 分)

代码中已经实现了表的初始化，并插入了简单的测试数据。

(4) 进阶一：异常处理（10 分）

你需要完善程序的异常处理逻辑，使得程序可以根据不同类型的异常，返回具有明确含义的 HTTP 状态码。例如：

- 当客户端请求一个不存在的资源（如某 ID 对应的歌曲）时，接口应返回 `404 (Not Found)`。

其他常见状态码如 `400 (Bad Request)`、`500 (Internal Server Error)` 等也应根据实际情况合理使用。

(5) 进阶二：分页处理（10 分）

考虑到歌曲数量可能非常多，一次性获取全部数据会降低性能与体验。因此，需要为相关接口增加分页查询功能。

具体而言，你需要参考前端接口需求文档`.md` 中关于”支持分页”的要求，修改接口设计，使其能够接收 `page_index`（指定要查询的页码）和 `page_size`（指定每页包含的数据条数）作为参数。接口的响应应仅包含符合分页条件的数据。

(6) 进阶三：初识异步（20 分）

Oblivionis 想为程序增加注册登录的功能，她决定使用手机号 + 验证码的方式实现注册，方法很简单，接受到请求后，后端给指定的邮箱发送一个验证码，但她马上遇到了一个问题：测试的时候明明没问题，怎么用户多了之后速度会变得非常慢?

原因很简单，因为发送邮件要调用第三方的 api。在并发很少的情况下我们对延迟的感知并不明显，同步请求是一个阻塞的过程，假设同时发来了 1000 条请求，每个线程处理一个请求，但服务器线程数有限（比如默认 200），剩下的 800 个请求会排队，每个线程阻塞 1 秒的话，用户等待时间可能长达 5 秒以上；

这种情况下，我们就需要用到异步，仅需几个线程，在等待 API 时处理其他请求，1 秒内就能完成所有 1000 个请求的处理，用户等待时间基本等于 API 延迟。

现在，给出一个模拟同步请求发送验证码的代码 `sync.py`，请尝试将发送验证码修改为异步操作，并比较一下：与同步相比，异步性能提高了多少？（5 分）

Listing 4: 同步发送验证码示例

```python
import time
import random

def send_verification_code(phone_number):
    """同步发送验证码（模拟网络延迟）"""
    code = random.randint(100000, 999999)
    print(f"Sending verification code {code} to {phone_number}")
    time.sleep(1) # 模拟1秒网络延迟
    return code

def main(phone_numbers):
    results = []
    for phone in phone_numbers:
        code = send_verification_code(phone)
        results.append(code)
    return results

if __name__ == "__main__":
    phone_numbers = [f"138000000{i}" for i in range(10)] # 10个测试手机号
    start_time = time.time()

    codes = main(phone_numbers)

    end_time = time.time()
    print("\nVerification codes:", codes)
    print(f"Total time taken: {end_time - start_time:.2f} seconds")
```

- 理解了异步的优点后，请尝试在 `db_manager.py` 与 `band_with_db.py` 中增加异步获取歌曲信息的的方法，需要支持分页（10 分）
- 编写一个压力测试脚本或使用其他工具，分别调用同步版本和异步版本的接口（例如，模拟 100 个客户端并发请求不同页码的数据），记录并分析两者的响应时间、吞吐量等关键指标，验证异步化在高并发场景下的显著性能提升。（5

分）

附录

(1) openlab 纳新课 _ 后端部分，以及配套的 ppt。

(2) FastAPI 相关教程

(3) FastAPI 使用 SQL 数据库

5.1.4 运行调试之章

好的乐队需要不断地磨合，来共同进步，写代码当然也是离不开调试的。你是否觉得，每次运行项目都使用前端进行测试有些过于麻烦了？

事实上这确实有些麻烦，毕竟不是所有框架都会给你特地准备一个可以热更新的交互式文档。对于前后端分离的项目来说更是如此：总不能专门写一个前端用来调试。

幸运的是，我们可以使用 API 测试工具大大减少工作量。

题目要求（10 分）

学会使用 Postman 等工具对后端接口进行测试。

附录

Postman 使用教程

5.2 验收要求

- 验收前需要多做测试，确保功能展示无误；
- 需要在验收前准备好说明文档；
- 随机提问考察代码原理、实现逻辑等方面问题；
- 验收时需要提交所写代码，或者将代码上传的 Github 仓库；
- 各项独立评分，总分最多 100 分，但还是希望有余力的同学在题目基础上多多优化拓展，多做多得。

5.3 注意事项

- 不限制编程语言；
- 不要**直接**用大模型进行代码生成！它只是学习的工具，请时刻谨记；
- 本题可在完成各小题基本要求上任意扩展，视扩展完成度与实用性给予额外加分；
- 附录提供的都是超链接，点击即可跳转。