誠樸雄偉 勵學敦行

南京大學
NANJING UNIVERSITY

《BiliHarmony鸿蒙应用开发技术文档》

@BiliHormony

题　　目：	BiliHarmony鸿蒙应用开发技术文档
上课时间：	每周三 晚 2:00
授课教师：	刘钦
姓　　名：	胡杨,范家磊,王宇翔,黄海锋,万鹏举,黄扬昊
学　　号：	221900495等
组　　别：	六人组开发
日　　期：	20225.1.18

BiliHarmony鸿蒙应用开发技术文档

简介：

项目名称：BiliHarmony
项目简介：

BiliHarmony 是一款基于鸿蒙操作系统（HarmonyOS）并使用 ArkTS 开发的第三方 Bilibili 客户端。该项目旨在为鸿蒙用户提供流畅、高效的 B 站视频观看体验，同时充分利用鸿蒙系统的分布式能力和 ArkTS 的高性能特性。

主要功能：

- **视频播放**：支持高清视频播放等功能，提供沉浸式观看体验。
- **内容推荐**：根据用户兴趣推荐热门视频等内容。
- **用户互动**：支持点赞、评论、投币等互动功能，与 B 站社区无缝连接。

技术亮点：

- 基于 ArkTS 开发，充分发挥鸿蒙系统的性能优势，实现高效渲染和流畅交互。
- 采用模块化设计，便于功能扩展和维护。

适用人群：

- 鸿蒙系统用户
- Bilibili 爱好者
- 对跨设备协同体验有需求的用户

未来规划：

- 支持更多 B 站功能，如直播、动态、消息通知等。
- 优化多设备协同体验，探索更多鸿蒙系统特性。
- 提供更丰富的个性化设置和主题支持。

目录：

1 简单部署

1.1 前端部署

解压ZIP文件:

BiliHormony_frontend.zip

直接在DevecoStudio中使用模拟器运行./videoPlayer3/videoPlayer/entry模块

1.2 后端部署

在主文件夹中./ 运行:

```
uvicorn main:app --reload
```

在提示中会告诉您一些事项命令:

```
请您打开设置的数据库,运行下面指令:
`alter table users convert to character set  utf8mb4`
和`alter table video convert to character set  utf8mb4`,
在运行完成以后,请输入y进行测试数据导入(y/n)
```

请按照上面的说明在MySQL的客户端服务器中运行命令:

```
alter table users convert to character set  utf8mb4
alter table video convert to character set  utf8mb4
```

然后再在后端的命令行中输入“y”,启动后端服务器

2 前端技术说明

2.1 技术栈

使用HormonyOS中的ArkTs语言编写

2.2 文件结构

```
├─componet
│      WebComponent.ets
│
├─entryability
│      EntryAbility.ts
│
├─fonts
│      iconfont.json
│      iconfont.ttf
│
├─model
│      back.jpg
│      back2.jpg
│      bar.jpg
│      HttpModel.ets
│      taskbar.png
│      VideoController.ets
│
├─pages
```

```
        blockpage.ets
        CommonConstants.ets
        HomePage.ets
        Login.ets
        musicPage.ets
        search.ets
        signup.ets
        SlashPage.ets
        user.ets
        videoPage.ets
    
├─view
        Comments.ets
        IndexModule.ets
        IndexSwiper.ets
        Model.ets
        MusicPlayer.ets
        MusicPlaySlider.ets
        Predata.ets
        test.ets
        VideoPlayer.ets
        VideoPlaySlider.ets
    
└─viewmodel
         ParamItem.ets
         SwiperVideoItem.ets
         VerticalVideoItem.ets
         VerticaMusicItem.ets
         VideoData.ets
```

2.3 重要页面/类说明

以下是代码中重要的模块、类和方法总结：

2.3.1 1. 重要模块

2.3.1.1 (1) 登录模块 (`Login`)

- **功能**：实现用户登录功能，包括账号密码输入、记住密码、登录请求、跳转主页等。
- **关键点**：
  - 使用 `http` 模块发送登录请求。
  - 通过 `data_preferences` 存储用户信息。
  - 支持记住密码功能。
  - 登录成功后跳转到主页。

2.3.1.2 (2) 注册模块 (`signup`)

- **功能**：实现用户注册功能，包括用户名、密码输入、密码校验、注册请求、跳转登录页面等。
- **关键点**：
  - 校验两次输入的密码是否一致。
  - 发送注册请求到后端，获取生成的账号。

- 注册成功后提示用户记住账号，并跳转到登录页面。

2.3.1.3 (3) 视频模块（`VideoModule`，`CateVideoModule`，`SearchVideoModule`，`HotVideoModule`）

- **功能**：展示视频列表，支持分类、搜索、热门视频等功能。
- **关键点**：
  - 使用 `http` 模块请求视频数据。
  - 支持下拉刷新功能。
  - 视频列表展示使用 `Grid` 布局。
  - 支持点击视频跳转到播放页面。

2.3.1.4 (4) 视频播放模块（`VideoPlayer`）

- **功能**：播放视频，支持点赞、投币、进度控制等功能。
- **关键点**：
  - 使用 `Video` 组件播放视频。
  - 支持点赞、投币功能，通过 `http` 模块与后端交互。
  - 使用 `VideoSlider` 组件控制视频进度。

2.3.1.5 (5) 分类模块（`KindModule`）

- **功能**：展示视频分类，支持点击分类跳转到对应分类页面。
- **关键点**：
  - 使用 `List` 组件展示分类。
  - 点击分类后跳转到对应分类的视频列表页面。

2.3.1.6 (6) 广告模块（`Ads`）

- **功能**：展示轮播广告。
- **关键点**：
  - 使用 `AutoChangeImageWithAnimation` 组件实现图片轮播。
  - 支持淡入淡出动画效果。

2.3.2 2. 重要类

2.3.2.1 (1) `PreferenceModel`

- **功能**：管理用户偏好设置，如记住密码功能。
- **关键方法**：
  - `writeData(data: PreData)`：保存用户账号密码。
  - `getPreference()`：获取存储的用户账号密码。

2.3.2.2 (2) `VerticalItemV`

- **功能**：展示单个视频的封面、标题、类型等信息。
- **关键点**：
  - 使用 `Image` 和 `Text` 组件展示视频信息。
  - 支持点击跳转到视频播放页面。

2.3.2.3 (3) `VideoController`

- **功能**：控制视频播放，包括播放、暂停、跳转等功能。
- **关键点**：
  - 与 `Video` 组件绑定，控制视频播放状态。

2.3.2.4 (4) `VideoSlider`

- **功能**：实现视频进度条，支持拖动调整播放进度。
- **关键点**：
  - 与 `VideoController` 配合使用，控制视频播放进度。

2.3.3 3. 重要方法

2.3.3.1 (1) 登录相关方法

- `login()`：
  - 发送登录请求，校验账号密码。
  - 登录成功后跳转到主页。
- `saveAccountsPreferences(user: User)`：
  - 保存用户账号密码到偏好设置。

2.3.3.2 (2) 注册相关方法

- `signup()`：
  - 发送注册请求，校验用户名和密码。
  - 注册成功后提示用户记住账号，并跳转到登录页面。

2.3.3.3 (3) 视频相关方法

- `getList()`：
  - 发送请求获取视频列表数据。
  - 支持分类、搜索、热门视频等不同场景。
- `prepared(duration: number)`：
  - 视频准备完成时调用，初始化视频时长和进度条。
- `finish()`：
  - 视频播放完成时调用，重置播放状态。

2.3.3.4 (4) 点赞和投币相关方法

- `setdianzan()`：
  - 发送点赞请求，更新点赞状态和数量。
- `settoubi()`：
  - 发送投币请求，更新投币状态和数量。

2.3.3.5 (5) 并发处理方法

- `ConcurrentFunc(num1: number, num2: number)`：
  - 使用 `taskpool` 实现并发计算，用于点赞和投币的数量更新。

2.3.4 4. 重要接口

2.3.4.1 (1) `LoginResponse`

- **功能**：定义登录请求的响应结构。
- **字段**：
  - `success: boolean`：登录是否成功。
  - `message?: string`：返回的消息。

2.3.4.2 (2) `User`

- **功能**：定义用户信息结构。
- **字段**：
  - `username: string | null`：用户名。
  - `account: string`：账号。
  - `password: string`：密码。

2.3.4.3 (3) `VideoHomepage_response`

- **功能**：定义视频信息的响应结构。
- **字段**：
  - `url: string`：视频地址。
  - `name: string`：视频名称。
  - `bv: number`：视频标识。
  - `cover_url: string`：视频封面地址。
  - `type: number`：视频类型。

2.3.5 5. 重要组件

2.3.5.1 (1) `Refresh`

- **功能**：实现下拉刷新功能。
- **关键点**：
  - 支持自定义刷新区域内容。
  - 支持刷新状态回调。

2.3.5.2 (2) `Video`

- **功能**：播放视频。
- **关键点**：
  - 支持自动播放、循环播放、进度控制等功能。
  - 提供 `onPrepared`、`onFinish` 等回调方法。

2.3.5.3 (3) `Badge`

- **功能**：展示带有数字角标的图标。
- **关键点**：
  - 用于点赞和投币的计数显示。

2.4 重要技术手段

1. 登陆注册功能：

注册界面使用用户和密码进行注册，检查格式正确后，提交注册申请到后端，后端生成一个自增的账号uid，以账号为主键，将用户名和密码添加到数据库中，并给定初始币数999，后端处理完成后，将生成的uid发送到前端，前端接收uid利用promptAction.showDialog显示给用户，用户记住后通过uid和密码进行登录，（因为uid是自增的主键，所以不可能存在冲突，所以用户的用户名和密码可以相同）。

登录界面通过uid和密码进行登录，用户输入uid和密码后，前端先检查是否为空，若为空提示用户，不为空就发送数据给后端，后端首先找有无匹配的uid，若无返回账号不存在，有匹配的则将用户输入的密码与后端uid对应的密码比对，不一致则密码错误，一致则登录成功，后端将对应的用户名和币数发送到前端，供其他页面使用。

2. UI设计优化：

利用状态变量监听手机方向，根据状态变量的不同值渲染不同的组件，实现页面随手机方向翻转，对于页面的响应式布局，我们通过使用百分比的height和width，以及对每个页面利用scroll或者list，grid实现滚动，来保证页面横向时，横方向仍铺满，纵向超过屏幕范围的部分通过滚动来使其得以加载与显示，确保界面元素不会出现变形或遮挡现象。当然，对于视频显示组件，为了功能性考虑，我们的横屏会通过监测窗口方向实现对组件的差异化渲染，达到全屏的效果，不再显示评论区。

我们利用router.pushUrl自带的平移动画效果实现页面切换的动画。

3. 多线程优化：

利用taskpool线程池实现，使用多线程来优化点赞和投币的处理，实现多线程任务的高效调度与资源管理。用**catch** (error)实现多线程环境下的异常处理。

4. 视频展示：

前端接收后端传来的url，视频名称，与视频分类，前端将其添加到Grid中，展示在主界面上，展示图片与视频标题，一次性加载一定量的视频，如果下拉界面，则重新加载，如果上拉界面，则继续向后端请求，增加视频数量，点击后进行router跳转，跳转到videopage搭建视频界面，然后调用videoplayer，构建视频播放器，调用视频url播放视频，同时实现点赞，投币与暂停功能。点赞，投币通过多线程进行，将更新后的数据与后端同步，同时通过if-else逻辑实现点赞或投币完成后点赞符号变红色。通过VideoController组件实现视频的暂停与播放。

5. 视频推荐

用户的每次点赞或投币，都会增加自己对该类别视频喜好的权值，每次下拉刷新或退出应用时，把更新后的权重返回后端，后端进行归一化处理，以后在对这个账号发送视频，则以权值为概率，选择不同类别的视频，发送给前端。

6. 视频搜索

用户搜索视频时，将搜索的名称发送给后端，后端搜索数据库中有没有这个视频，若有则在搜索页面显示这个视频，否则显示没有该视频。

7. 视频分区

点击视频对应的分区时，后端就从对应的类别随机返回视频信息给前端，供前端展示。

8. 使用首选项Preference

在登陆之后勾选记住密码的操作,将密码和账户保存在用户手机里面的首选项Preference文件中,并且在下次进入应用的时候会读取里面的数据,直接填写密码账户.

3 后端技术说明

3.1 技术栈

使用python==3.12的fastapi模块,univorn容器作为运行环境

3.2 主要文件结构

```
C:.
|  .env // 环境变量
|  main.py // 主要逻辑
|  myjson
|  README.md
|  test.json
|  test_main.py
|
├─.idea
|
├─app
|  |  crud.py // 增删查改
|  |  database.py // 数据库加载
|  |  models.py // 数据库表
|  |  schemas.py // 返回与接受的结构定义
|  |  security.py //安全相关函数
|  └  __init__.py
|
└─__pycache__


```

3.3 重要模块

以下是针对 `main.py`、`crud.py` 和 `schemas.py` 三个重要模块的详细功能介绍和解析：

3.3.1 1. `main.py`：实现与前端的通信接口，以及数据库的初始数据插入

功能概述

- **通信接口**：提供 RESTful API，供前端调用，实现用户管理、视频管理、登录认证等功能。
- **数据库初始化**：在应用启动时，插入初始数据（如默认配置、热门视频等）。
- **核心功能**：
  - 用户注册、登录、获取用户信息。
  - 视频的增删改查、点赞、投币等操作。
  - 根据用户喜好推荐视频。
  - 日志记录和配置管理。

关键模块

1. **用户管理**：
   - 用户注册：通过 `/users/signup` 接口实现用户注册。
   - 用户登录：通过 `/users/login` 接口实现用户登录，并返回 JWT 令牌。
   - 获取用户信息：通过 `/user/detail` 接口获取用户的喜好和硬币数。
2. **视频管理**：
   - 获取热门视频：通过 `/video/hot` 接口获取点赞数最多的视频。

- 获取首页推荐视频：通过 `/video/homepage` 接口根据用户喜好推荐视频。
- 获取分类视频：通过 `/video/cate` 接口获取指定类型的视频。
- 视频点赞和投币：通过 `/video/getlike` 和 `/video/getcoin` 接口实现视频点赞和投币功能。

3. **初始化数据**：
   - 在应用启动时，加载 `test.json` 文件中的视频数据并插入数据库。
   - 初始化配置表，确保数据加载状态被记录。
4. **日志记录**：
   - 使用 `logging` 模块记录 API 请求日志，便于调试和监控。

3.3.2 2. `crud.py`：后端的主要实现逻辑

功能概述

- 实现与数据库的交互逻辑，包括用户管理、视频管理、数据查询和更新等操作。
- 封装业务逻辑，供 `main.py` 中的接口调用。

关键模块

1. **用户管理**：
   - 查询用户：通过 `get_user_by_account` 和 `get_user` 函数查询用户信息。
   - 创建用户：通过 `create_user` 函数实现用户注册。
   - 获取用户详情：通过 `get_user_details` 函数获取用户的喜好和硬币数。
2. **视频管理**：
   - 创建视频：通过 `create_video` 函数插入视频数据。
   - 查询视频：通过 `get_video_by_bv`、`get_hot_videos`、`get_videos_by_type` 等函数查询视频信息。
   - 更新视频：通过 `update_coin_and_like` 和 `increment_video_like` 函数实现视频点赞和投币功能。
3. **推荐算法**：
   - 根据用户喜好推荐视频：通过 `get_homepage_videos_by_like` 函数实现基于用户喜好的视频推荐。
4. **随机视频**：
   - 通过 `get_random_videos_by_num` 函数随机获取指定数量的视频。

3.3.3 3. `schemas.py`：与前端通信的类的定义

功能概述

- 定义与前端通信的数据模型（Pydantic 模型），用于请求和响应的数据验证和序列化。
- 确保数据格式的一致性，并提供 ORM 支持。

关键模块

1. **用户模型**：
   - `UserBase`：用户基础模型，包含账号和密码。
   - `UserCreate`：用户注册模型，继承自 `UserBase`。
   - `User`：用户响应模型，包含用户 ID、喜好和硬币数。
   - `UserDetailRequest` 和 `UserDetailResponse`：用于获取用户详情的请求和响应模型。
2. **视频模型**：

- `VideoBase`：视频基础模型，包含视频 URL、名称和封面 URL。
- `VideoCreate`：视频创建模型，继承自 `VideoBase`，并包含点赞数、投币数和类型。
- `VideoResponse`：视频响应模型，包含视频 ID、类型、名称、点赞数、投币数、URL 和封面 URL。
- `VideoBv`、`VideoType`、`VideoLike`、`VideoCoin`、`VideoName`：用于视频相关操作的请求模型。

3. **其他模型**：
   - `UpdateCoinRequest` 和 `UpdateCoinResponse`：用于更新用户硬币数和视频硬币数的请求和响应模型。

3.3.4 模块之间的关系

1. `main.py` 调用 `crud.py` 中的函数，实现业务逻辑。
2. `crud.py` 使用 `schemas.py` 中的模型进行数据验证和序列化。
3. `schemas.py` 定义了与前端通信的数据格式，确保数据的一致性和安全性。

3.4 数据库说明

3.4.0.1 两张表

1. users

u_id	account	password	like	coin
Integer	String	String	String	Integer
主键	唯一			
NOT NULL	NOT NULL	NOT NULL		NOT NULL

2. videos

bv	name	type	like	coin	url
Integer	String	Integer	Interger	Interger	String
主键					
NOT NULL	NOT	NOT NULL	NOT NULL	NOT NULL	NOT NULL

3.5 重要技术手段

3.5.1 1. 使用 SQLAlchemy 结合连接池高效管理数据库连接

SQLAlchemy 的连接池机制通过预先创建并维护一定数量的数据库连接，避免了频繁创建和销毁连接的开销，从而显著提升了数据库操作的性能。连接池可以根据实际需求动态调整连接数量，支持高并发场景，同时通过设置连接超时和回收策略，有效防止连接泄漏和资源浪费。这种方式特别适合需要频繁访问数据库的应用场景，能够显著降低数据库的响应时间并提高系统的整体吞吐量。

3.5.2 2. 结合 MySQL 使用异步数据库库（如 `asyncmy` 或 `databases`）

通过使用异步数据库库（如 `asyncmy` 或 `databases`），可以将数据库操作转换为非阻塞模式，避免传统同步操作导致的线程阻塞问题。异步操作利用 `async`/`await` 语法，使服务器能够在等待数据库响应的同时处理其他请求，从而大幅提升并发处理能力。这种方式特别适合高并发的 Web 应用，能够有效提高服务器的资源利用率和响应速度。

3.5.3 3. 利用 FastAPI 的依赖注入机制自动获取数据库连接或会话

FastAPI 的依赖注入机制可以自动管理资源（如数据库连接）的生命周期，减少代码重复并提高可维护性。通过定义依赖项（如数据库会话），可以在路由处理函数中自动注入所需的资源，确保资源的正确初始化和释放。这种方式不仅简化了代码结构，还增强了代码的可测试性，因为依赖项可以轻松替换为模拟对象进行单元测试。

3.5.4 4. 使用 Depends 创建数据库连接的依赖注入方法

通过 FastAPI 的 `Depends` 方法，可以创建数据库连接的依赖注入逻辑。这种方式将数据库会话的管理与业务逻辑分离，使代码更加模块化和可复用。依赖注入方法会在每个请求开始时自动创建数据库会话，并在请求结束时自动释放，确保资源的有效管理和避免连接泄漏。这种方法特别适合需要频繁访问数据库的应用场景，能够显著提高代码的可读性和可维护性。

3.5.5 5. 使用多重测试集成方式对后端进行测试

通过多重测试集成方式（包括单元测试、集成测试和性能测试），可以全面覆盖系统的功能、性能和安全性。单元测试用于验证单个函数或模块的正确性，集成测试用于确保多个模块的协同工作，而性能测试则用于评估系统在高并发场景下的表现。这种多层次的测试策略能够有效提高开发效率和软件质量，帮助开发者快速发现并修复潜在问题，确保系统的稳定性和可靠性。

3.5.6 6. 记录用户点赞内容的类型，并依此更新用户偏好

通过记录用户点赞内容的类型，可以动态分析用户的兴趣偏好，从而实现个性化的内容推荐。每次用户点赞时，系统会更新用户的偏好数据，并根据这些数据调整推荐算法，为用户提供更符合其兴趣的内容。这种个性化推荐算法不仅能够提升用户的满意度和参与度，还能增加用户粘性，为平台带来更高的活跃度和留存率。通过不断优化推荐算法，可以进一步提升推荐效果，为用户提供更加精准和个性化的服务。

---

4 前后端连接说明

4.1 user类接口说明

请求方式均为post

1. 注册：/user/signup
   前端发送的json格式为

```
{
"account" :  str
"password": str
}
```

接受示例:

```
{
  "account": "string",
  "password": "string",
  "u_id": 0,
  "like": "",
  "coin": 100
}
```

2. 登录：/user/login
   前端发送的json格式为

```
{
"account" :  str
"password": str
}
```

3. 我的界面中的数据：/user/detail
   前端发送的json格式为

```
{
"account" :  str
}
```

接受示例:

```
{
  "like": "",
  "coin": 0
}
```

4.2 video类接口说明

请求方式均为post

1. 根据bv号返回视频/video/bv

   请求示例:

```
{
  "bv": 0
}
```

   接受示例:

```
{
  "bv": 0,
  "type": 0,
  "name": "string",
  "like": 0,
  "coin": 0,
  "url": "string",
  "cover_url": "string"
}
```

2. 返回热门视频,/video/hot

   返回video的List,示例:

```
[
  {
    "bv": 0,
    "type": 0,
    "name": "string",
    "like": 0,
    "coin": 0,
    "url": "string",
    "cover_url": "string"
  }
]
```

3. 返回主页视频,/video/homepage

   返回示例同2

4. 返回一个类型的视频,/video/cate

   接受示例:

```
{
  "type": "string"
}
```

   返回示例同2

5. 根据bv号查询视频赞数,/video/like

   请求示例:

```
{
  "bv": 0
}
```

   返回示例:

```
{
  "like": 0
}
```

6. 根据bv号查询视频币数,/video/coin

   请求示例:

```
{
  "bv": 0
}
```

   返回示例:

```
{
  "coin": 0
}
```

7. 根据name来搜索相关的视频,/video/name

   请求示例:

```
{
  "name": "string"
}
```

返回示例同2

8. 根据账号和视频处理投币操作,video/getcoin

请求示例:

```
{
  "account": "string",
  "bv": 0
}
```

返回示例:

```
{
  "message": "string",
  "user": {
    "account": "string",
    "password": "string",
    "u_id": 0,
    "like": "",
    "coin": 100
  },
  "video": {
    "bv": 0,
    "type": 0,
    "name": "string",
    "like": 0,
    "coin": 0,
    "url": "string",
    "cover_url": "string"
  }
}
```

9. 根据bv号处理点赞操作,/video/getlike

请求示例:

```
{
  "bv": 0
}
```

返回示例:

```
{
  "bv": 0,
  "type": 0,
  "name": "string",
  "like": 0,
  "coin": 0,
  "url": "string",
  "cover_url": "string"
}
```

5 其他技术说明

5.1 使用docker部署

5.1.1 修改.env文件

修改.env文件中的数据库相关的IP地址,端口,用户,密码:

```
DB_URL = "mysql+pymysql://root:114514@localhost:3306/backend2?charset=utf8mb4"
```

5.1.2 构建docker

运行命令:

```
docker build -t myfastapi .
# 语法说明
#docker build -t 镜像名称 Dockerfile文件所在路径
```

5.1.3 运行docker

运行命令:

```
docker run -d --name myfastapi -p 8000:8000 myfastapi
```

5.1.4 镜像导出/导入

```
docker save -o myfastapi.tar myfastapi
docker load < fastapi.tar
```

5.2 使用Conda部署

5.2.1 建立conda环境

```
conda create -n backend python==3.12
```

5.2.2 激活conda环境/安装python依赖

```
conda activate backend
pip install -r requirements.txt
```

5.2.3 运行python文件

```
uvicorn main:app --reload
```

5.3 数据库部署技术

使用MySQL数据库

在后端程序第一次运行完以后会在数据库中建立两张users和video表,此时还需要对数据库进行一定的部署:在对应的数据库运行命令:

```
alter table users convert to character set utf8mb4;
alter table video convert to character set utf8mb4;
```

上面的命令用于将两张表的字符集进行转换为utf8mb4.