GameDock API v1 文档（逻辑接口说明）

本文档描述 **GameDock 项目使用的逻辑 API 约定**：

- 在当前课程版本中，这些“接口”都可以完全由 **Android 客户端本地实现**（Repository + Retrofit + Room + WorkManager）。
- 将来如果增加真实后端，只要用相同的路径和数据结构实现 REST API，客户端几乎不用大改。

0. 通用约定（Conventions）

0.1 基本信息

- **应用名称**：GameDock — Game Deals & Freebies Aggregator
- **主要目的**：聚合多平台 PC 游戏价格和免费领取信息，提供：
  - 跨商店价格对比
  - 史低价格标记
  - 愿望单 & 打折/限免提醒
- **目标支持商店**：Steam、Epic Games Store、GOG、Humble、Fanatical 等
- **数据格式**： `application/json; charset=utf-8`
- **时间格式**：ISO-8601，例如 `2025-10-19T10:30:00Z`
- **货币格式**：ISO 4217，例如 `"USD"` 、 `"CNY"`

0.2 分页规则

大部分列表接口 **可以** 支持分页参数：

Code (text):

```
page: int       // 页码，从 1 开始，默认 1
pageSize: int   // 每页数量，默认 20，推荐最大 50
```

标准分页响应示例：

Code (json):

```
{
  "items": [ /* list items */ ],
  "page": 1,
  "pageSize": 20,
  "total": 134
}
```

1. 认证（Authentication）

1.1 GameDock 自有账号（可选）

如果后续需要 GameDock 自己的账号系统，可以使用如下 HTTP 头：

Code (http header):

```
Authorization: Bearer <access_token>
```

当前课程版本可以 **完全不实现** 这一层。

1.2 商店账号绑定（Steam / Epic / GOG 等）

商店账号绑定目前在客户端通过 **官方 OAuth / 网页授权流程 + Deep Link 回调** 完成。

- 逻辑端点： `GET /me/accounts`

示例响应：

Code (json):

```
[
  {
    "store": "steam",
    "displayName": "My Steam Account",
    "linkedAt": "2025-10-20T09:00:00Z"
  },
  {
    "store": "epic",
    "displayName": "My Epic Account",
    "linkedAt": "2025-10-21T12:30:00Z"
  }
]
```

> 在课程实现中，这些信息可以完全存在本地（Room / SharedPreferences），不一定是实际的 HTTP 接口。

2. 游戏基础信息（Games）

2.1 搜索游戏

Endpoint： `GET /games/search`
用途：根据关键字搜索游戏，返回基础信息（不含价格）。

查询参数：

Code (text):

```
q      (string, required)  // 搜索关键字，例如 "Cyberpunk 2077"
region (string, optional)  // 区域，默认 "global"
```

响应示例：

Code (json):

```
{
  "items": [
    {
      "id": "cyberpunk-2077",
      "title": "Cyberpunk 2077",
      "coverUrl": "https://cdn.steam.com/cover/1091500.jpg",
      "stores": ["steam", "gog", "epic"],
      "tags": ["RPG", "Open World"],
      "releaseDate": "2020-12-10"
    }
  ],
  "page": 1,
  "pageSize": 20,
  "total": 1
}
```


2.2 获取游戏详情

Endpoint： `GET /games/{gameId}`
用途：获取单个游戏的基础信息（不含价格数据）。

路径参数：

Code (text):

```
gameId (string)  // 统一的游戏 ID，例如 "cyberpunk-2077"
```

响应示例：

Code (json):

```
{
  "id": "cyberpunk-2077",
  "title": "Cyberpunk 2077",
  "coverUrl": "https://cdn.steam.com/cover/1091500.jpg",
  "stores": ["steam", "gog", "epic"],
  "tags": ["RPG", "Open World"],
  "releaseDate": "2020-12-10",
  "shortDescription": "An open-world, action-adventure story set in Night
  "developer": "CD PROJEKT RED",
  "publisher": "CD PROJEKT RED"
}
```


3. 免费游戏（Freebies）

3.1 获取免费游戏列表

Endpoint： `GET /freebies`
用途：返回当前各个支持商店的免费游戏（限时免费 / 永久免费领取）。

查询参数：

Code (text):

```
region (string, optional)  // 区域，默认 "global"
store  (string, optional)  // 指定商店，例如 "steam"、"epic"
page, pageSize (int, optional)  // 分页参数
```

响应示例：

Code (json):

```
{
  "items": [
    {
      "id": "epic-ghostrunner-free-2025-11",
      "gameId": "ghostrunner",
      "title": "Ghostrunner",
      "store": "epic",
      "storeName": "Epic Games Store",
      "isFree": true,
      "originalPrice": 29.99,
      "currency": "USD",
      "startsAt": "2025-10-30T15:00:00Z",
      "endsAt": "2025-11-06T23:59:59Z",
      "url": "https://store.epicgames.com/free-games",
      "thumbnail": "https://cdn.epicgames.com/ghostrunner.jpg",
      "tags": ["Action"]
    }
  ],
  "page": 1,
  "pageSize": 20,
  "total": 12
}
```


> UI 可以根据 `endsAt` 计算倒计时。无网络时，读取 Room 中上一次缓存的列表。

4. 多平台价格比价（Offers）

4.1 获取价格列表

Endpoint： `GET /offers`
用途：返回同一游戏在多个商店的最新价格，并标记是否为历史最低价。

查询参数：

Code (text):

```
q      (string)  // 可选，按名称搜索
gameId (string)  // 可选，统一游戏 ID；q 和 gameId 至少提供一个
region (string, optional)  // 区域，默认 "global"
stores (string, optional)  // 多个商店，逗号分隔，例如 "steam,epic,gog"
```

响应示例：

Code (json):


```
[
  {
    "gameId": "cyberpunk-2077",
    "title": "Cyberpunk 2077",
    "store": "steam",
    "storeName": "Steam",
    "price": 29.99,
    "currency": "USD",
    "originalPrice": 59.99,
    "discountPercent": 50,
    "isHistoricalLow": true,
    "lastCheckedAt": "2025-10-19T10:12:00Z",
    "url": "https://store.steampowered.com/app/1091500",
    "region": "US"
  },
  {
    "gameId": "cyberpunk-2077",
    "title": "Cyberpunk 2077",
    "store": "gog",
    "storeName": "GOG",
    "price": 32.99,
    "currency": "USD",
    "originalPrice": 59.99,
    "discountPercent": 45,
    "isHistoricalLow": false,
    "lastCheckedAt": "2025-10-19T10:13:00Z",
    "url": "https://www.gog.com/en/game/cyberpunk_2077",
    "region": "US"
  }
]
```


> 在比价界面中，通常按 `price` 升序排序，并对 `isHistoricalLow == true` 的条目加“史低”高亮。

5. 价格历史（Price History）

5.1 获取价格历史

Endpoint： `GET /price-history/{gameId}`
用途：为 30/90 天价格曲线提供时间序列数据。

路径参数：

Code (text):

```
gameId (string)  // 统一游戏 ID
```

查询参数：

Code (text):

```
store (string, optional)  // 按商店过滤，例如 "steam"
days  (int, optional)     // 时间窗口，默认 90 天
```

响应示例：

Code (json):

```
[
  {
    "store": "steam",
    "date": "2025-08-01",
    "price": 39.99,
    "currency": "USD",
    "isHistoricalLow": false
  },
  {
    "store": "steam",
    "date": "2025-09-01",
    "price": 29.99,
    "currency": "USD",
    "isHistoricalLow": false
  },
  {
    "store": "steam",
    "date": "2025-10-01",
    "price": 19.99,
    "currency": "USD",
    "isHistoricalLow": true
  }
]
```


> 如果没有历史数据，返回空数组 `[]`，UI 可以显示“暂无历史数据”。

6. 套餐与慈善包（Bundles）

6.1 获取 Bundles 列表

Endpoint： `GET /bundles`

用途：展示当前的套餐 / 慈善包 / 合集折扣（例如 Humble、Fanatical）。

查询参数：

Code (text):

```
source     (string, optional)  // 来源过滤，例如 "humble", "fanatical"
activeOnly (boolean, optional) // 是否只返回未过期活动，默认 true
page, pageSize (int, optional) // 分页参数
```

响应示例：

Code (json):


```
{
  "items": [
    {
      "id": "humble-rpg-legends-2025-10",
      "name": "Humble RPG Legends Bundle",
      "source": "humble",
      "url": "https://www.humblebundle.com/games/rpg-legends",
      "thumbnail": "https://hb.img/rpg_legends.jpg",
      "minPrice": 12.0,
      "currency": "USD",
      "endsAt": "2025-10-31T23:59:59Z",
      "includedGames": [
        "Pathfinder: Kingmaker",
        "Pillars of Eternity",
        "Torment: Tides of Numenera"
      ]
    }
  ],
  "page": 1,
  "pageSize": 10,
  "total": 3
}
```


7. 愿望单（Watchlist）

> 在课程版本中，Watchlist 完全可以由本地 Room 表实现。
> 下面的接口是逻辑上的 CRUD 约定。

7.1 新增愿望单条目

Endpoint： `POST /watchlist`

请求体示例：

Code (json):

```
{
  "gameId": "cyberpunk-2077",
  "title": "Cyberpunk 2077",
  "targetPrice": 20.0,
  "currency": "USD",
  "notifyOnFree": true,
  "notifyOnHistoricalLow": true
}
```

响应 201 示例：

Code (json):

```
{
  "id": "watch-cyberpunk-2077",
  "createdAt": "2025-10-19T10:30:00Z"
}
```

7.2 获取愿望单列表

Endpoint： `GET /watchlist`

响应示例：

Code (json):

```
[
  {
    "id": "watch-cyberpunk-2077",
    "gameId": "cyberpunk-2077",
    "title": "Cyberpunk 2077",
    "targetPrice": 20.0,
    "currency": "USD",
    "notifyOnFree": true,
    "notifyOnHistoricalLow": true,
    "createdAt": "2025-10-19T10:30:00Z"
  }
]
```

7.3 更新愿望单条目（可选）

Endpoint： `PATCH /watchlist/{id}`

请求体示例：

Code (json):

```
{
  "targetPrice": 18.0,
  "notifyOnFree": false
}
```

响应 200 示例：

Code (json):

```
{
  "id": "watch-cyberpunk-2077",
  "updatedAt": "2025-10-20T09:00:00Z"
}
```

7.4 删除愿望单条目

Endpoint： `DELETE /watchlist/{id}`

- 成功： `204 No Content`
- 未找到： `404 Not Found`

8. 通知与后台任务（Notifications & Background Work）

8.1 PriceCheckWorker（价格检查 Worker）

通过 **WorkManager 的 CoroutineWorker** 实现（例如每 6 小时执行一次）。

主要职责：

Code (text):

```
1）从本地存储中读取所有 WatchItem 条目；
2）对每个 gameId 调用 /offers 和 /freebies；
3）根据规则判断：targetPrice、notifyOnFree、notifyOnHistoricalLow；
4）发送本地通知，并记录已通知状态，避免重复提醒。
```

通知示例（概念）：

Code (text):

```
Title: "Cyberpunk 2077 史低价来了！"
Body:  "现在 Steam 仅需 $19.99，创历史最低价，点此查看详情。"
Click: 深链到 /detail/{gameId}（游戏详情界面）
```

9. 状态与健康检查（可选）

9.1 状态接口

Endpoint： `GET /status`
用途：可选健康检查端点（主要为未来后端预留）。

响应示例：

Code (json):

```
{
  "status": "ok",
  "timestamp": "2025-10-19T11:00:00Z",
  "version": "1.0.0",
  "stores": {
    "steam": "ok",
    "epic": "ok",
    "gog": "degraded"
  }
}
```

10. 错误处理（Error Handling）

10.1 标准错误结构

Code (json):

```
{
  "error": {
    "code": "RATE_LIMIT",
    "message": "Too many requests. Please try again later.",
    "details": null
  }
}
```

10.2 常见错误码

Code (text):

```
NETWORK_ERROR  (503): 网络或上游服务不可用
RATE_LIMIT     (429): 请求过于频繁
INVALID_REGION (400): region 参数不合法
NOT_FOUND      (404): gameId 或 watchlist id 未找到
INTERNAL_ERROR (500): 未预期服务器错误
```

11. 数据模型汇总（Data Model Summary）

Code (text):

```
Game:
  id, title, coverUrl, stores, tags, releaseDate

Freebie:
  id, gameId, title, store, storeName, isFree,
  originalPrice, currency, startsAt, endsAt, url, thumbnail, tags

Offer:
  gameId, title, store, storeName, price, currency,
  originalPrice, discountPercent, isHistoricalLow,
  lastCheckedAt, url, region

PricePoint:
  store, date, price, currency, isHistoricalLow

BundleInfo:
  id, name, source, url, thumbnail, minPrice, currency,
  endsAt, includedGames[]

WatchItem:
  id, gameId, title, targetPrice, currency,
  notifyOnFree, notifyOnHistoricalLow, createdAt
```

12. 实现说明（课程版本）

在当前课程版本中，这些“API”都可以 **完全在 Android 客户端内部实现**：

Code (text):

- `DealsRepository` 使用 `Retrofit` 调用公开商店 `API` 或聚合服务；
- `Room` 缓存 `Freebie / Offer / PricePoint` 数据，支持离线浏览；
- `/watchlist` 映射到本地 `WatchItemEntity` 表，做增删改查；
- `PriceCheckWorker` 周期性读取愿望单并调用 `Repository`，
  满足条件时触发本地通知。

如果未来增加后端，只需要在服务器端实现同名 REST 接口，客户端只需将 Repository 的实现从“本地/Mock”切到“网络版”，而不需要大规模修改 UI 和 ViewModel。