

从AIGC至MVP

使用AIGC工具提升开发效率

呆若牦牛 2024年1月6日

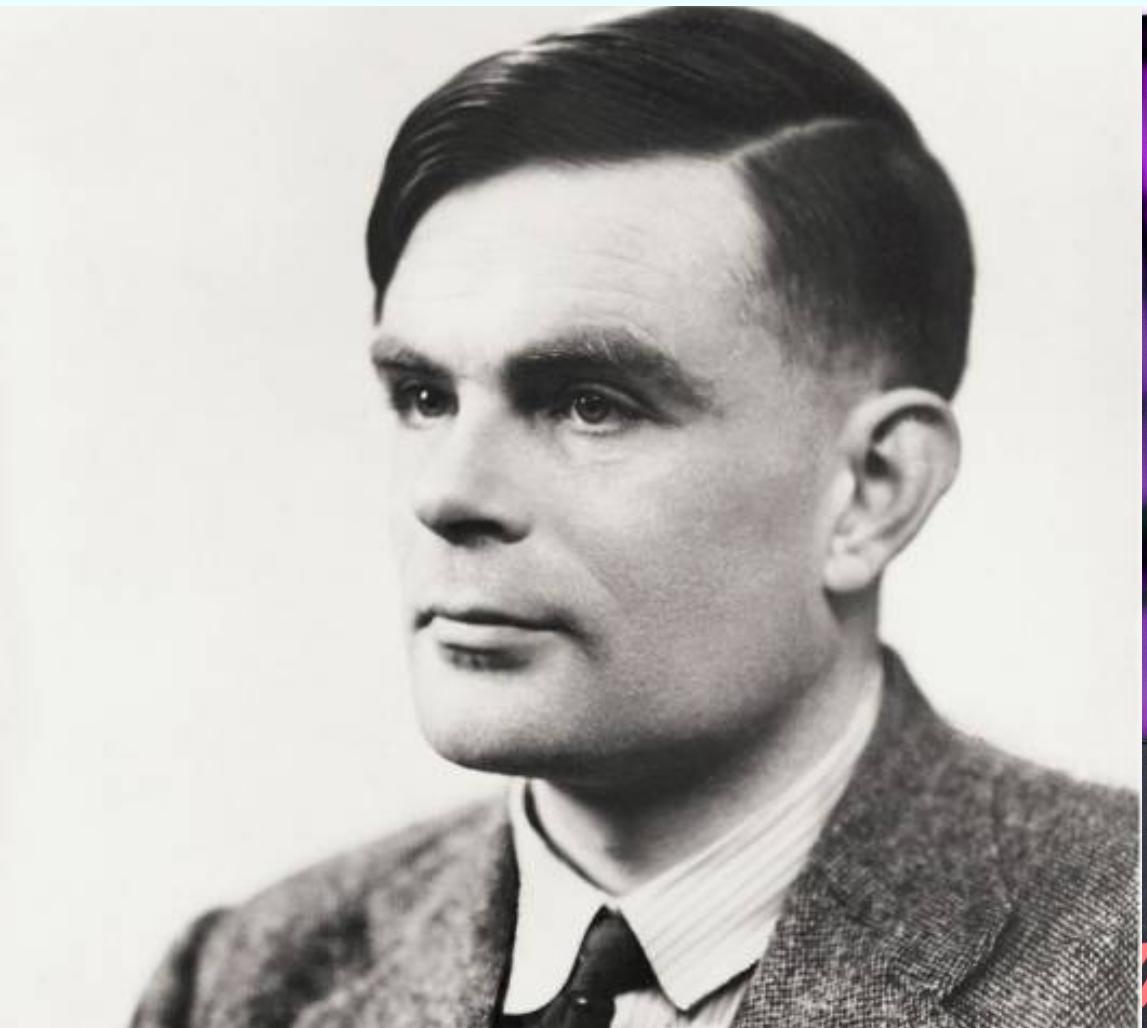
- 01 AI2.0 时代
- 02 Generative AI
- 03 ChatGPT/文心一言/通义千文
- 04 Copilots
- 05 实现MVP
- 06 未来已来

01 - AI 2.0时代

人工智能起源



1956 达特茅斯会议



阿兰 图灵



杰弗里 辛顿

AI 2.0

2022年11月30号ChatGPT3.5 橫空出世

ChatGPT是AI2.0时代的现象级应用



伊尔亚 苏茨克维



萨姆 奥特曼



格雷格 布罗克曼

AI 2.0具备3个鲜明的特点

- 对超级海量的数据，无需进行人工标注，即进行自监督学习
- 基础模型规模非常大，需要几千张GPU来训练
- 训练出来的基础模型具有跨领域知识，可以通过为调用降低成本的方法来训练，以适应不同领域的任务

来源：知乎 李开复

02 - Generative AI

《流浪地球3》AI预告片



微博网友 @数字生命卡兹克 制作

AIGC工具：Midjourney V6 + Magnific + Runway Gen2

主要业务应用场景

- 基于各种不同的数据集构建单一知识库
- 将聊天机器人转变为始终在线的客户服务助理
- 即时查找和总结复杂信息
- 为多渠道生成广告素材内容
- 简化商品的录入，分类，搜索标记和营销方式
- 自动化数据收集和记录流程
- 监控机器生成的事件，以预测即将进行的维护
- 利用以数据为依据的分析洞见加速产品创新
- 打造虚拟造型师来帮助消费者找到所需商品
- 实时获取护理相关信息，提供更好的患者体验
- 控制运营成本并提升内容效果
- 摘要和撰写录音内容并编入索引，以记录讨论中的更多观点
- 利用高级数据分析和通信工具降低碳排放量
- 更快地检测和解决问题，改善客户体验
- 跨设备的一致语音体验
- 提供个性化的内容推荐，包括音乐，视频和博客
- 通过代码协助提高开发者工作效率

来源：<https://cloud.google.com/ai/generative-ai>

写代码？

- 即时查找和总结复杂信息
- 更快地检测和解决问题，改善客户体验
- 通过代码协助提高开发者工作效率

03 - ChatGPT/文心一言/通义千文

对程序员的帮助



ChatGPT4 > 文心一言4.0 > 通义千问2.0

04 - Copilots

主要功能

- 自动完成代码：当你在编写代码时，Copilot 可以根据上下文自动推断可能的代码补全选项，并为你提供建议。
- 提供函数和类的建议：Copilot 可以根据你当前正在解决的问题和上下文，为你提供函数和类的建议。
- 生成注释和文档：Copilot 可以帮助你生成注释和文档，以提高代码的可读性和可维护性。
- 提供示例代码：Copilot 可以根据你的需求和上下文，为你生成示例代码，帮助你理解特定功能或解决问题的方法。
- 语法纠错和风格建议：Copilot 还可以帮助你纠正语法错误和改善代码风格。
- 提供文档和资源链接：Copilot 还可以为你提供与特定代码片段或问题相关的文档和资源链接
- 生成单元测试

如何选择

Github Copilot

通义灵码

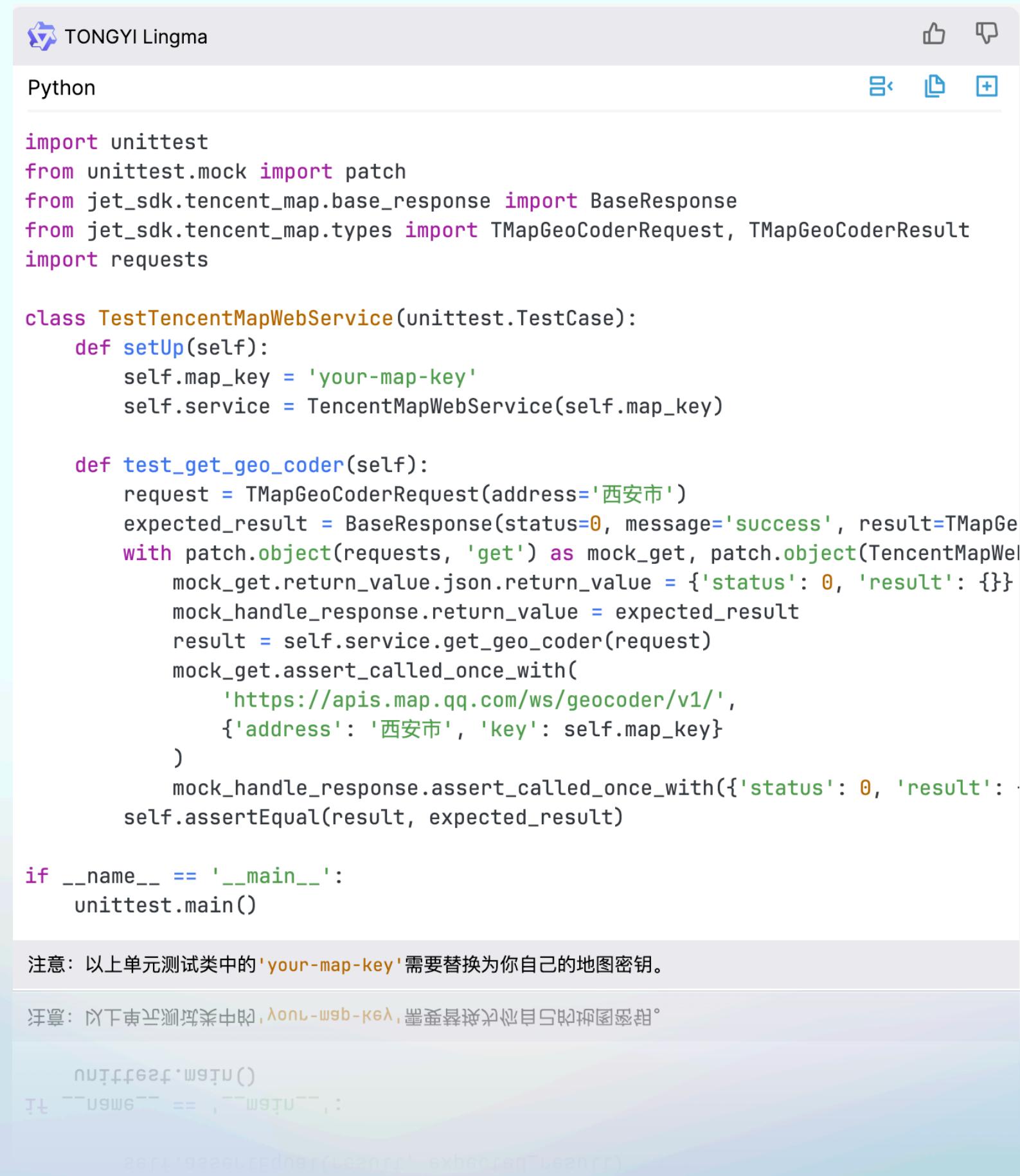
AI Assistant

代码补全

```
error: { ^1      You, Today • Uncommitted changes
    unauthorized: '未授权', ^2
    forbidden: '禁止访问', ^3
    notFound: '未找到', ^4
    internalServerError: '服务器内部错误', ^5
    badGateway: '服务网关错误', ^6
    serviceUnavailable: '服务不可用', ^7
    gatewayTimeout: '网关超时', ^8
    badRequest: '请求错误', ^9
    notAcceptable: '不可接受',
    requestTimeout: '请求超时',
    conflict: '冲突',
}
}

conflict: '冲突'
```

单元测试



The screenshot shows a Python code editor window titled "TONGYI Lingma". The code is a unit test for the Tencent Map Web Service using the unittest module. It includes imports for unittest, patch, BaseResponse, TMapGeoCoderRequest, TMapGeoCoderResult, and requests. The test class, TestTencentMapWebService, inherits from unittest.TestCase. It contains a setUp method to set up a map key and service, and a test_get_geocoder method that uses a patch to mock requests and responses. The test checks if the service returns the expected result for a geocoding request for '西安市'. The code ends with an if __name__ == '__main__': block calling unittest.main(). A note at the bottom of the code editor says: "注意: 以上单元测试类中的'your-map-key'需要替换为自己的地图密钥。" (Note: Replace 'your-map-key' in the test class with your own map key.)

```
import unittest
from unittest.mock import patch
from jet_sdk.tencent_map.base_response import BaseResponse
from jet_sdk.tencent_map.types import TMapGeoCoderRequest, TMapGeoCoderResult
import requests

class TestTencentMapWebService(unittest.TestCase):
    def setUp(self):
        self.map_key = 'your-map-key'
        self.service = TencentMapWebService(self.map_key)

    def test_get_geocoder(self):
        request = TMapGeoCoderRequest(address='西安市')
        expected_result = BaseResponse(status=0, message='success', result=TMapGe
        with patch.object(requests, 'get') as mock_get, patch.object(TencentMapWe
            mock_get.return_value.json.return_value = {'status': 0, 'result': {}}
            mock_handle_response.return_value = expected_result
        result = self.service.get_geocoder(request)
        mock_get.assert_called_once_with(
            'https://apis.map.qq.com/ws/geocoder/v1/',
            {'address': '西安市', 'key': self.map_key}
        )
        mock_handle_response.assert_called_once_with({'status': 0, 'result': {}})
        self.assertEqual(result, expected_result)

if __name__ == '__main__':
    unittest.main()
```

注意: 以上单元测试类中的'your-map-key'需要替换为自己的地图密钥。

注意: 以下单测类中，`your-map-key`需要替换为自己的地图密钥。

```
(unitest) tianyuan@tianyuan-OptiPlex-5090:~/Desktop/tencent_map
```

05 - 实现MVP

废话不多说，开始吧



You

我要做一个共享汽车平台，这个产品包括以下部分：

1. 微信小程序，用户可以通过注册，登录小程序查找最近的可用车辆，并可以扫码解锁
2. Android应用，这个应用运行汽车，有一个屏幕，用户通过小程序支付之后，可以通过App控制车辆运行
3. 使用Vue3和ElementPlus开发一个管理端应用，可以查看订单，管理用户，管理车辆
4. 开发一个后台，使用Python和Sanic框架

要实现一个MVP，需要哪些类型的人员分工协作？还需要考虑什么？比如产品的设计等，请详细的给我指导

最，具体的设计并提供建议

首先做一个 MVP，需要哪类人员分工协作；还需要考虑什么；比如产品的设计

第一次结果



ChatGPT

要开发一个共享汽车平台，您需要组建一个跨职能团队，主要包括以下角色和职责：

- 产品经理**：负责整体产品规划、功能定义、用户体验设计、市场分析等。产品经理将决定产品的核心功能，并指导团队实现这些功能。
- UI/UX设计师**：负责设计用户界面和用户体验。这包括微信小程序、Android应用和管理端应用的视觉设计，确保产品的界面既美观又易用。
- 前端开发人员**：
 - 微信小程序开发者：负责开发微信小程序，确保用户可以方便地注册、登录、查找车辆和扫码解锁。
 - Vue.js开发者：使用Vue3和ElementPlus开发管理端应用，包括订单查看、用户和车辆管理功能。
- Android开发人员**：负责开发Android应用，这个应用将在车辆上运行，用户可以通过它控制车辆。
- 后端开发人员**：使用Python和Sanic框架开发后台，负责处理应用数据、用户请求、车辆数据等。
- 测试工程师**：负责测试所有应用的功能和性能，确保软件的质量。
- 系统架构师/云计算专家**：负责设计系统架构，确保应用的稳定性和扩展性，处理大量数据和请求。
- 数据分析师**：负责分析用户数据和市场趋势，帮助团队做出基于数据的决策。

除了组建团队，还需要考虑以下方面：

- 市场调研**：了解目标市场和用户群体，确定产品定位。
- 合规性和安全性**：确保产品遵守当地法律法规，特别是数据保护和交通法规。
- 商业模式和盈利方式**：明确产品的盈利模式，如按时间收费、按里程收费等。
- 合作伙伴和供应商**：可能需要与车辆制造商、地方政府、支付平台等建立合作关系。
- 技术选型和工具**：确定合适的技术栈和开发工具，以支持开发和运维工作。
- 用户反馈和产品迭代**：产品发布后，收集用户反馈，不断优化和迭代产品。

综上所述，开发一个共享汽车平台是一个复杂的项目，需要多领域能力的团队协作，以及对市场和用户需求的深入了解。

以下几点是开发共享汽车平台时需要注意的：

1. 市场调研：了解目标市场和用户群体，确定产品定位。

2. 技术选型：根据项目需求选择合适的技术栈和工具。

3. 合规性：确保产品遵守当地法律法规，特别是数据保护和交通法规。

4. 安全性：确保产品具有良好的安全性和稳定性。

5. 商业模式：明确产品的盈利模式，如按时间收费、按里程收费等。

6. 合作伙伴：可能需要与车辆制造商、地方政府、支付平台等建立合作关系。

7. 用户反馈：产品发布后，收集用户反馈，不断优化和迭代产品。

8. 供应链管理：确保供应链稳定，及时交付产品。

9. 品牌建设：打造品牌形象，提高品牌知名度。

10. 市场推广：通过各种渠道进行市场推广，吸引用户。

不考虑商业可行性



You

不需要考虑商业方面的可行性，只是产品本身，请给出相关的建议

聚焦产品本身



ChatGPT

了解到您希望专注于产品的开发和实现，而不涉及商业方面的可行性。在这种情况下，以下是一些建议，以确保产品的成功开发和有效实施：

技术团队组成和角色

- 产品经理：**定义产品的功能和需求，制定详细的产品路线图和迭代计划。
- UI/UX设计师：**
 - 设计直观、易用的用户界面。
 - 为微信小程序、Android应用和管理端应用创建一致的视觉和操作体验。
- 前端开发人员：**
 - 微信小程序开发者：负责实现注册、登录、车辆查找和扫码解锁功能。
 - Vue.js开发者：使用Vue3和ElementPlus开发管理端应用，包括数据呈现、订单管理、用户管理和车辆管理功能。
- Android开发人员：**
 - 开发车辆控制的Android应用，集成支付功能和车辆操作控制。
- 后端开发人员：**
 - 使用Python和Sanic框架建立后端服务。
 - 处理数据存储、用户请求、车辆状态更新等。
- 测试工程师：**
 - 负责测试所有应用的功能和性能。
 - 确保产品在不同设备和环境下的稳定性。

开发和实施方面的建议

- 需求分析和文档编写：**
 - 在开发前，彻底分析需求。
 - 编写详细的功能文档和技术规范。
- 敏捷开发方法：**
 - 采用敏捷开发模式，分阶段实现功能，持续迭代。
- 代码管理和版本控制：**
 - 使用Git等版本控制系统管理代码。
 - 确保代码质量和合理的分支管理策略。
- 持续集成和持续部署 (CI/CD)：**
 - 建立自动化测试和部署流程，加快开发周期，降低错误率。
- 性能优化和可扩展性：**
 - 确保应用性能满足需求，尤其是在高并发场景下。
 - 设计可扩展的架构，以适应未来的用户增长和功能扩展。
- 用户反馈和产品迭代：**
 - 在产品开发的各个阶段收集用户反馈。
 - 基于反馈不断优化产品。
- 安全性考虑：**
 - 重视用户数据和支付信息的安全。
 - 实施加密、认证和授权机制。
- 技术选型：**
 - 根据项目需求和团队技术栈选择合适的技术和框架。

我是产品经理



You

I am a product manager. Please help me generate a product architecture diagram based on the above requirements.

Requirement details are omitted.

产品架构

ChatGPT

Creating a product architecture diagram for your shared car platform requires the representation of various components and their interactions. Let's outline the main components and their relationships, based on your requirements:

1. User Interface Layer:

- **WeChat Mini Program:** For user registration, login, vehicle location, and QR code scanning.
- **Android App:** Installed in cars for vehicle control post-payment through the mini-program.
- **Web Admin Portal (Vue3 & ElementPlus):** For managing orders, users, and vehicles.

2. Application Layer:

- **Service APIs:** Interfaces for the WeChat Mini Program and Android App to interact with backend services.
- **Admin Panel Backend:** Serves the web admin portal with functionalities for order, user, and vehicle management.

3. Backend Layer (Python & Sanic Framework):

- **User Management Service:** Handling user data, authentication, and authorization.
- **Vehicle Management Service:** Managing vehicle data, status, and availability.
- **Order Management Service:** Processing and storing order details.
- **Payment Integration Service:** Handling payment processing and verification.

4. Database Layer:

- **Database Server:** Storing user data, vehicle information, orders, and other relevant data.

5. External Integrations:

- **Payment Gateways:** For processing payments.
- **Third-party Services:** If any, like GPS services for vehicle tracking.

6. Infrastructure:

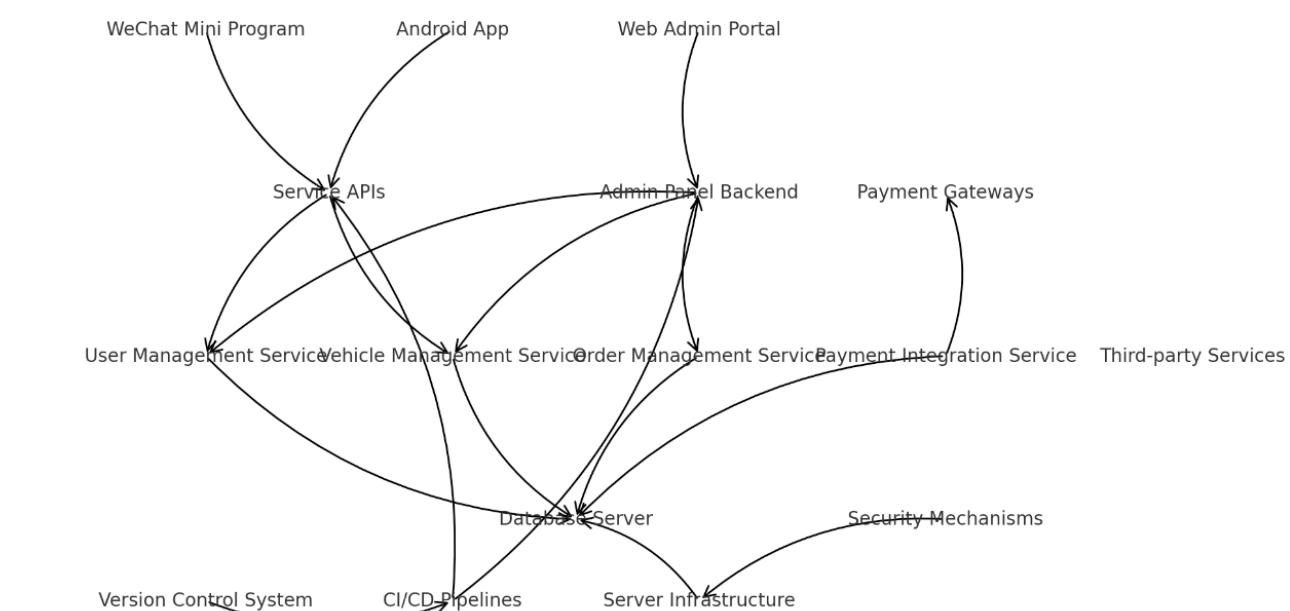
- **Server Infrastructure:** Hosting backend services and databases.
- **Security Mechanisms:** Implementing encryption, secure API access, etc.

7. Development and Operations:

- **Version Control System:** Like Git for source code management.
- **Continuous Integration and Deployment Pipelines:** For automated testing and deployment.

This architecture assumes a cloud-based backend with APIs serving the various front-end applications. The database is centralized, ensuring data consistency across different services. Let's generate a diagram to visualize this architecture.

Product Architecture Diagram for Shared Car Platform

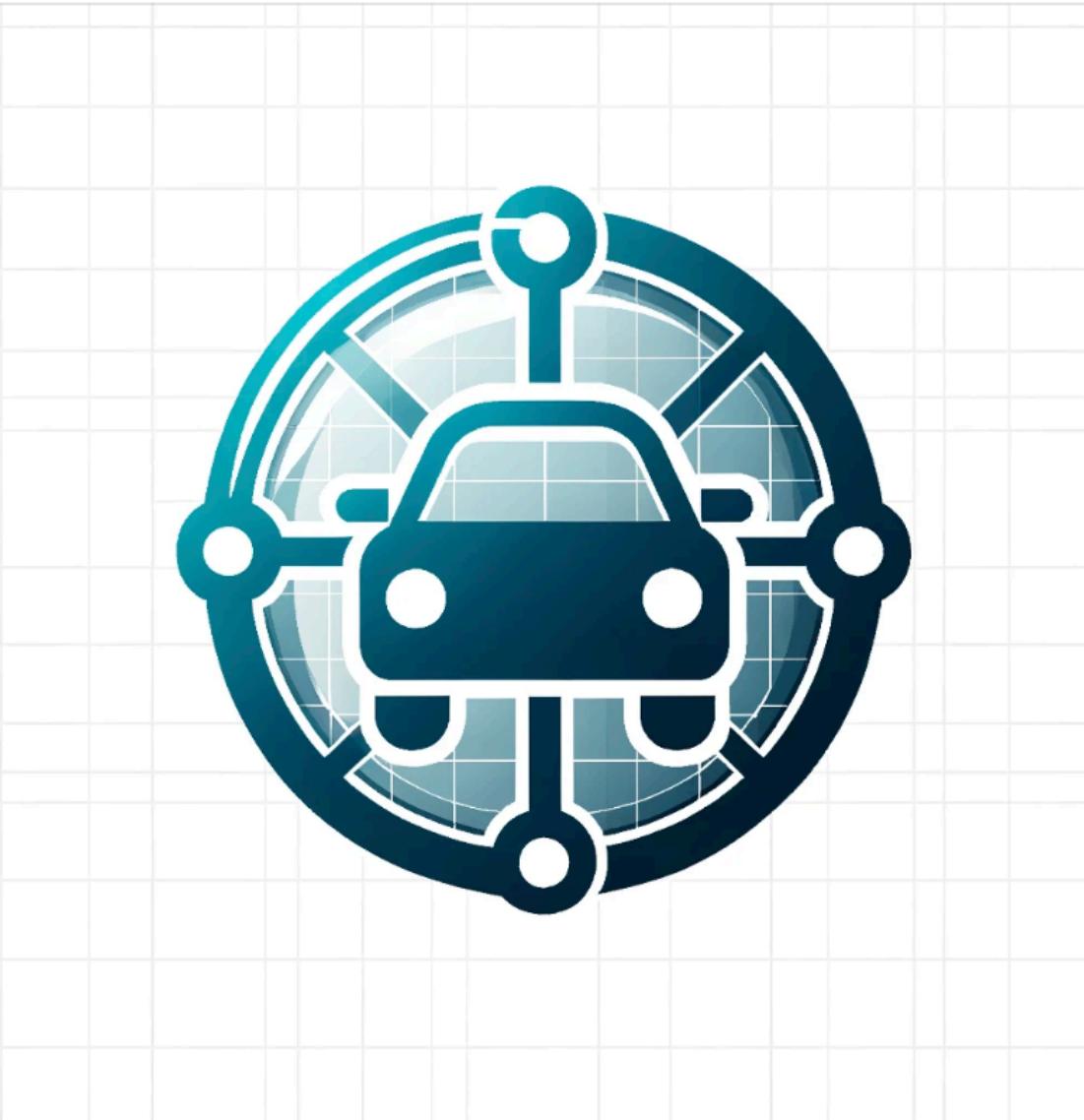


Here is the product architecture diagram for your shared car platform. This diagram illustrates the relationships and interactions between different components:

- **User Interface Layer** includes the WeChat Mini Program, Android App, and Web Admin Portal, representing the front-end parts of your system.
- **Application Layer** with Service APIs and Admin Panel Backend caters to the needs of the front-end applications and integrates with backend services.
- **Backend Layer** consists of various services like User Management, Vehicle Management, Order Management, and Payment Integration.
- **Database Layer** centralizes data storage, ensuring that all services have consistent and up-to-date information.
- **External Integrations** include Payment Gateways and Third-party Services.
- **Infrastructure** includes Server Infrastructure for hosting services and databases, and Security Mechanisms for data protection.
- **Development and Operations** are represented by Version Control System and CI/CD Pipelines for code management and deployment.

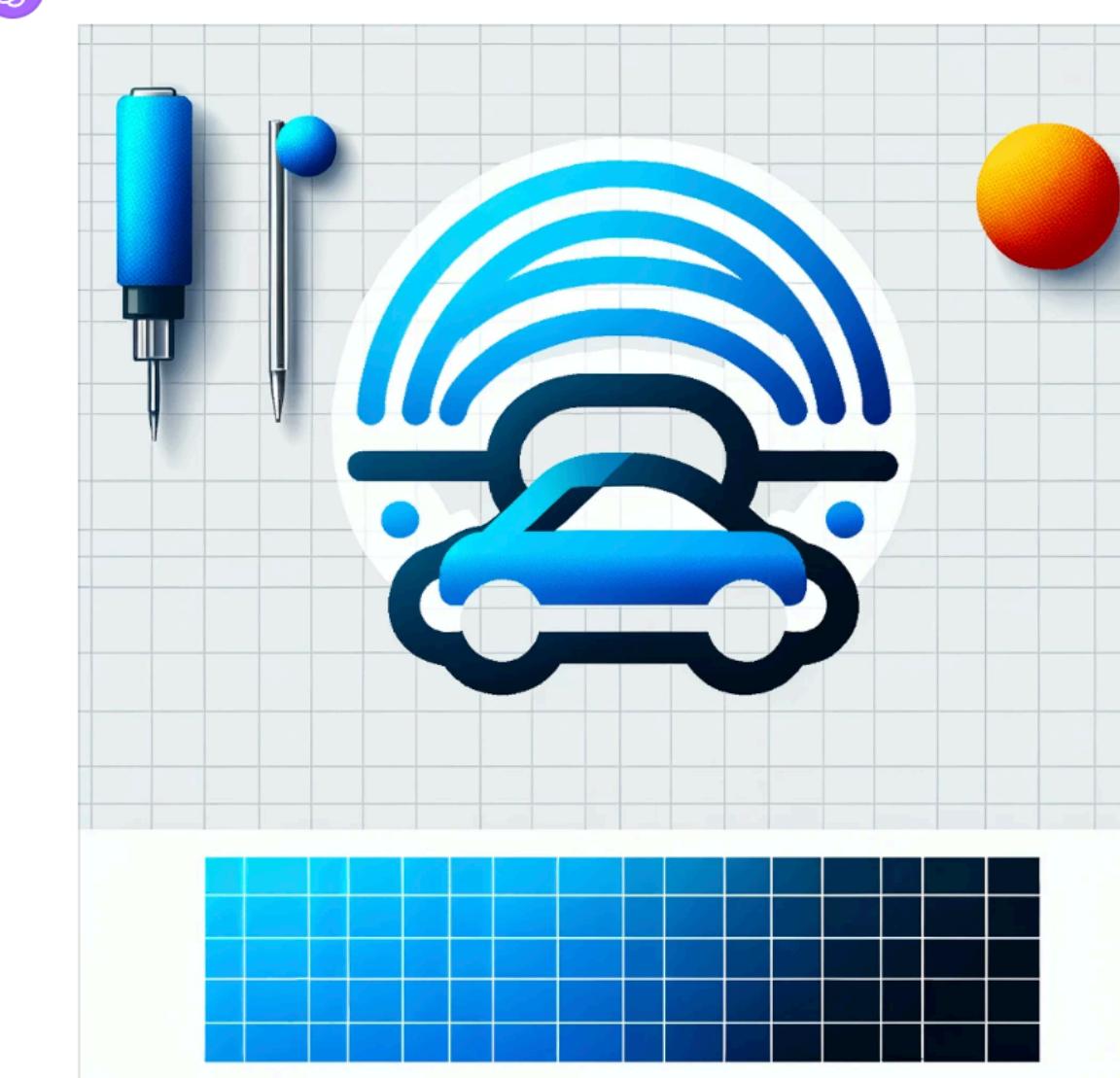
我是UI/UX设计师

ChatGPT



Here is the logo designed based on your requirements. It has a transparent background with a tool bar on the left for a clean look. The logo is centered on a grid background.

ChatGPT



Here is the updated logo design. It's simpler and uses the specified vibrant blue color. The background is a grid, and there are some tools on the sides.

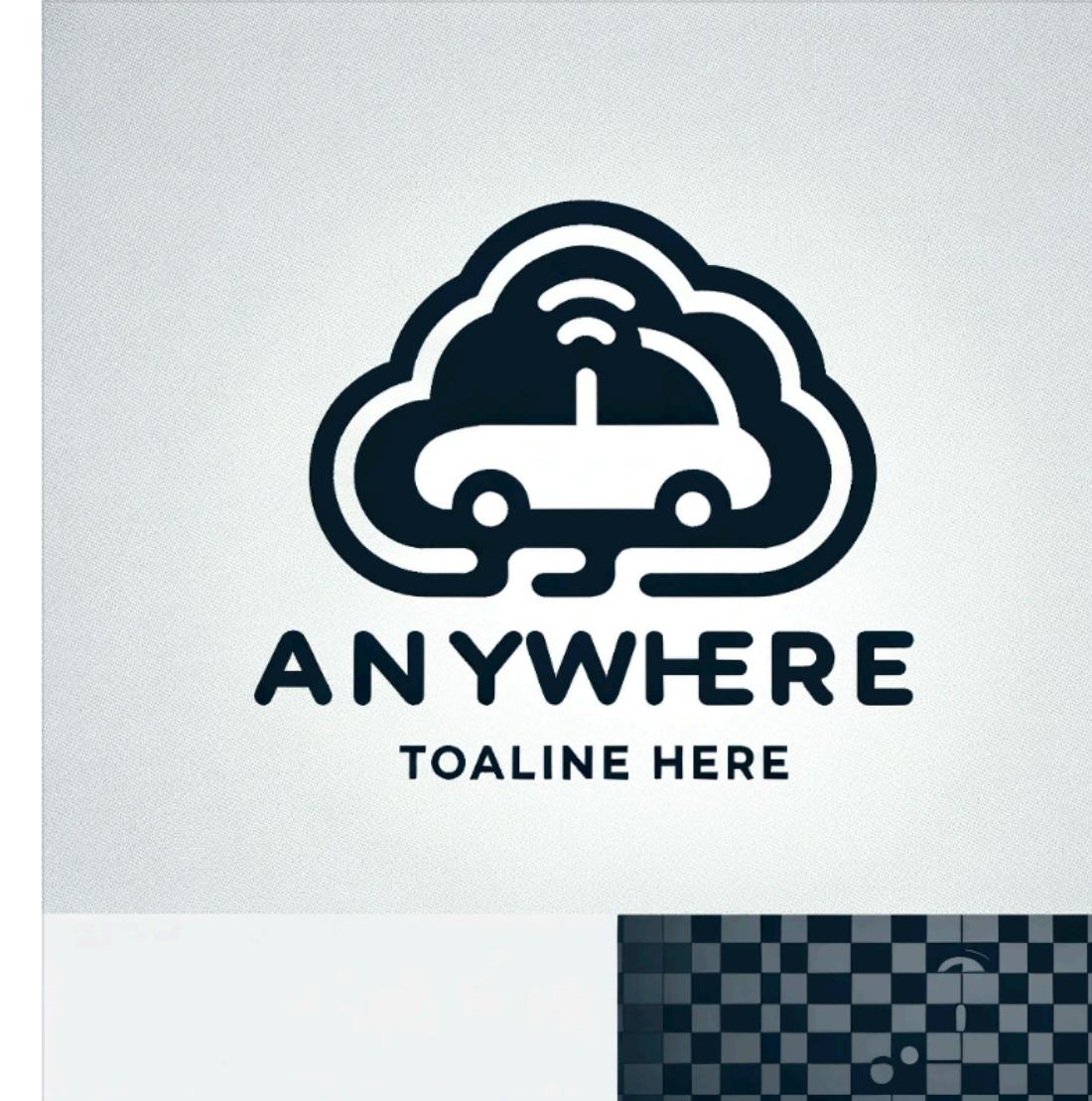
我是UI/UX设计师

ChatGPT



Here is the revised logo, designed to be even simpler and using only the solid color #048ef0. The design is minimalist and modern, reflecting the theme of car sharing. Here is the revised logo based on your requirements: a simple blue car icon inside a circle.

ChatGPT



The logo has been updated to include the word 'AnyWhere', merging it with the original car icon. The revised logo is a white car icon inside a dark cloud shape, with the text 'ANYWHERE' and 'TOALINE HERE' below it.

我是UI/UX设计师



科技 | 通讯



餐饮 | 食品



服饰 | 时尚



美容 | 化妆



母婴 | 教育



运动 | 健身



娱乐 | 文化



旅游 | 民宿



农业 | 环境



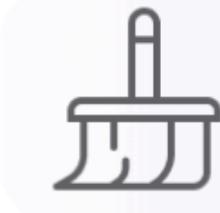
商业 | 金融



医疗 | 健康



房产 | 建材



生活 | 服务



制造 | 运输

建筑 | 房屋

商业 | 业商

零售 | 零售

医疗 | 医疗

房产 | 建材

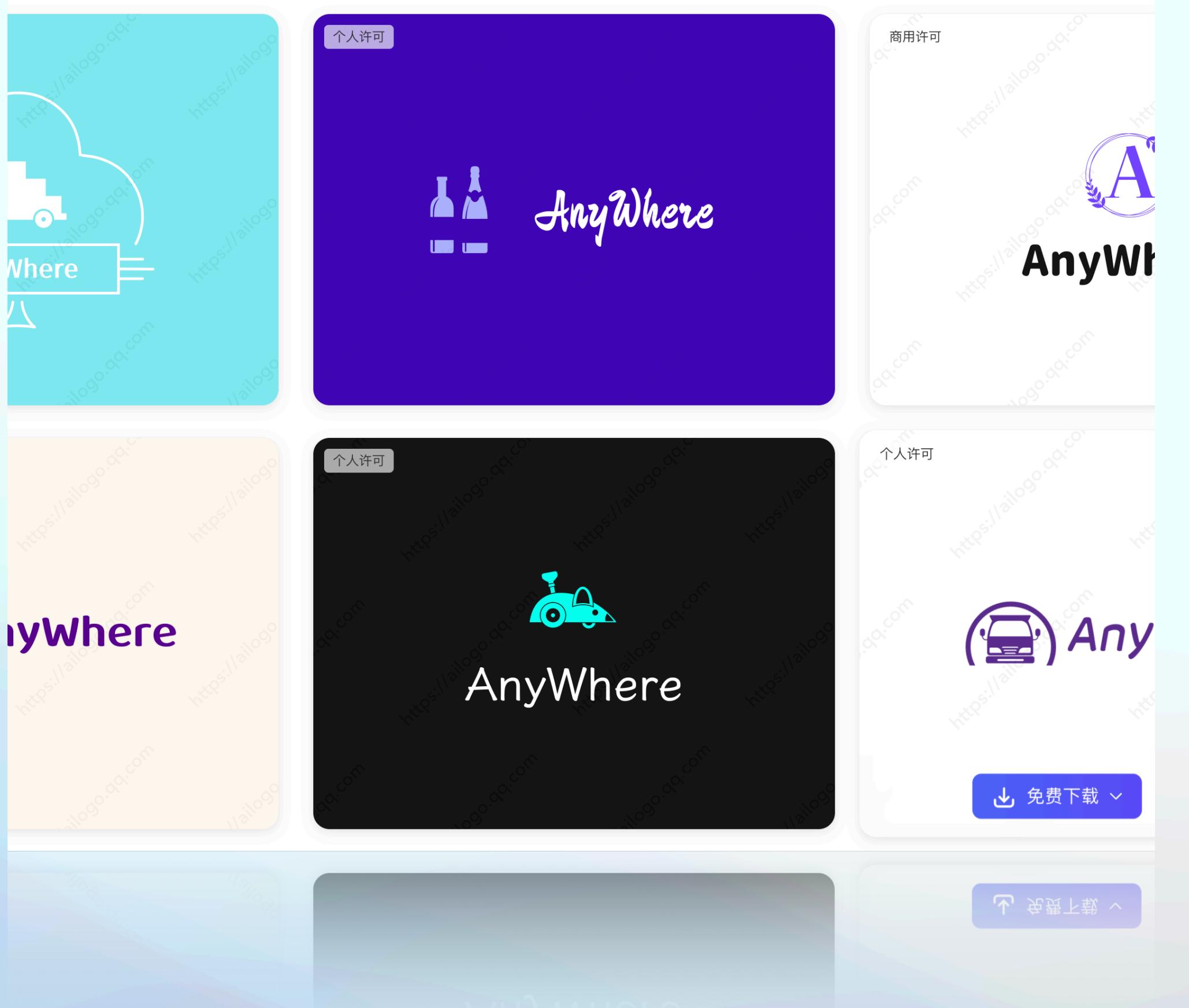
生活 | 服务

制造 | 运输

我是UI/UX设计师



我是UI/UX设计师



我是前端开发工程师



You

要开发微信小程序，需要掌握哪些背景知识

我是前端开发工程师



ChatGPT

开发微信小程序需要掌握以下背景知识：

1. 微信小程序框架：

- 理解微信小程序的基本架构，包括它的组件和模块。
- 学习微信小程序的生命周期，包括页面加载、显示、隐藏和卸载等。

2. 编程语言：

- JavaScript**：微信小程序主要使用JavaScript进行逻辑编写。
- HTML5/CSS3**：小程序的WXML（类似于HTML）和WXSS（类似于CSS）用于布局和样式设计。

3. 微信官方开发工具：

- 使用微信开发者工具进行小程序的开发、测试和调试。

4. API和组件：

- 学习微信小程序提供的丰富API和组件，用于实现各种功能，如支付、地理位置、通讯录等。
- 理解如何使用这些API和组件来提高开发效率和用户体验。

5. 小程序的设计规范：

- 遵守微信的设计指南和最佳实践，以确保小程序的用户体验符合微信生态的标准。

6. 数据通信：

6. 数据通信：

- 理解小程序如何与后端服务器进行数据交换，包括使用wx.request进行网络请求等。

7. 用户认证和授权：

- 理解微信的登录认证机制，包括如何获取和使用用户信息。

8. 性能优化：

- 学习如何优化小程序的性能，包括减少加载时间、优化数据传输等。

9. 发布和审核流程：

- 了解小程序的发布流程，包括审核、上线和版本管理。

10. 微信支付集成（如果需要）：

- 理解如何在小程序中集成微信支付功能。

11. 小程序云开发（可选）：

- 掌握微信小程序云开发的知识，包括使用云数据库、云函数等。

• 理解云数据库、云函数等基础知识，以及如何使用它们来提升小程序的性能。

12. 小结：

- 理解并掌握以上所有知识点，才能成为一名合格的微信小程序开发工程师。

我是前端开发工程师



You

请推荐几个小程序开发使用的第三方依赖

我是前端开发工程师



ChatGPT

在微信小程序开发中，使用第三方依赖可以大大提高开发效率和程序功能性。以下是一些常用第三方依赖推荐：

1. WeUI:

- 微信官方设计团队开发的一套简洁、高效的微信小程序UI库。
- 提供多种基础样式和组件，如按钮、列表、输入框等，与微信原生视觉体验保持一致。

2. Vant Weapp:

- 一款轻量、可靠的小程序 UI 组件库。
- 提供丰富的组件选择，如轮播图、导航栏、标签页等，便于快速搭建高质量小程序界面。

3. MinUI:

- 由蘑菇街前端团队开发的小程序组件库。
- 包含多种常用组件，注重性能和用户体验，支持自定义主题。

• 由蘑菇街前端团队开发的小程序组件库。
• 包含多种常用组件，注重性能和用户体验，支持自定义主题。

4. IUNI UI:

wxParse:

- 用于解析HTML和Markdown内容的小程序组件。
- 可以将后端传来的HTML或Markdown内容在小程序中渲染展示，非常适合需要展示富文本内容的小程序。

Tencent Cloud Weapp SDK:

- 腾讯云针对小程序开发提供的SDK。
- 包含对象存储、实时通信、数据库等服务，适用于需要后端支持的复杂小程序。

ColorUI:

- 高颜值、高自定义的小程序UI库。
- 提供多种设计风格的组件，适用于追求独特视觉效果的小程序开发。

ZanUI:

- 有赞团队开发的一款轻量级、可靠的小程序UI库。
- 提供丰富的UI组件，易于扩展和自定义。

在选择第三方依赖时，重要的是要考虑库的稳定性、文档的完善程度和社区的活跃度。这些因素将直接影响到开发过程的便利性和小程序的质量。同时，也要注意遵守微信小程序的官方政策和限制，以确保小程序的顺利上线和运行。

括读取配置文件、设置全局变量等操作，以及对某些API的调用。

这些功能通常会通过插件的形式实现。例如，可以通过安装“uniCloud”插件来实现云函数、云数据库等云服务的集成。同时，也可以通过安装“wxParse”插件来实现富文本解析功能。

我是前端开发工程师

The image shows a mobile application interface with a light blue background. At the top left is a circular profile picture of a person with glasses. To its right, the word "You" is written in a large, bold, black font. Below this, a message in Chinese reads: "请检索一下westore库，帮我介绍一下这个库". The next section is titled "ChatGPT" with a purple circular icon containing a white swirl symbol to its left. Below this title are seven identical message items, each consisting of an orange circle with a white exclamation mark icon followed by the text "Error analyzing" and a small downward arrow. The bottom portion of the screen is blurred, showing more of the same message template.

 You
westore的repo地址是https://github.com/Tencent/westore, 请简单介绍一下这个库

 ChatGPT
! Error analyzing ▾

WeStore is a library developed by Tencent for use in WeChat mini-program development. It is specifically designed to enhance state management in such applications. By using WeStore, developers can more efficiently manage the state of their mini-programs, leading to better-structured code and potentially improved app performance. The library offers a streamlined approach to state management, making it easier to maintain large and complex applications. For detailed information and usage instructions, you can visit the [WeStore GitHub repository](#). [→]

我是前端开发工程师

```
/*地图(容器)显示大小*/
width:1200px;
height:400px;
}
</style>
<!--引入Javascript API GL, 参数说明参见下文-->
<script src="https://map.qq.com/api/gljs?v=1.exp&key=0B4BZ-D4W3U-B7VV0-4PJWW-6TKDJ-WPB77"></script>
<script>
    //地图初始化函数, 本例取名为init, 开发者可根据实际情况定义
    function initMap() {
        //定义地图中心点坐标
        var center = new TMap.LatLng(39.984120, 116.307484)
        //定义map变量, 调用 TMap.Map() 构造函数创建地图
        var map = new TMap.Map(document.getElementById('container'), {
            center: center,//设置地图中心点坐标
            zoom: 17.2,    //设置地图缩放级别
            pitch: 43.5,   //设置俯仰角
            rotation: 45   //设置地图旋转角度
        });
    }
</script>
<\scrif
}
})!
```

我是前端开发工程师

MultiPolygon

表示地图上的多个多边形，可以自定义每个多边形的样式。

构造函数

TMap.MultiPolygon(options:[MultiPolygonOptions](#))

方法	返回值	说明
setMap(map: Map)	this	设置地图对象，如果map为null意味着将多多边形同时从地图中移除。
setZIndex(zIndex: Number)	this	设置图层绘制顺序。
setDashOffset(dashOffset: Number)	this	设置图层绘制顺序。

来源：<https://lbs.qq.com/webApi/javascriptGL/gIDoc/gIDocVector#7>

我是前端开发工程师



You

请以下内容定义TypeScript接口，保留注释：

MultiPolygon

表示地图上的多个多边形，可以自定义每个多边形的样式。

构造函数

TMap.MultiPolygon(options:MultiPolygonOptions)

方法	返回值	说明
----	-----	----

setMap(map:Map) this 设置地图对象，如果map为null意味着将多多边形同时从地图中移除。

setZIndex(zIndex: Number) this 设置图层绘制顺序。

setGeometries(geometries: PolygonGeometry[]) this 更新多边形数据，如果参数为null或undefined不会做任何处理。

setStyles(styles:MultiPolygonStyleHash) this 设置MultiPolygon图层相关样式信息，如果参数为null或undefined不会做任何处理。

setVisible(visible: Boolean) this 设置图层是否可见。

getMap() Map 获取地图对象，若为空返回null。

getGeometries() PolygonGeometry[] 获取多边形数据。

getGeometries(): PolygonGeometry[] {
 return this.geometries; // 多边形数据
}

getMap(): Map | null {
 return this.map; // 地图对象
}

setVisible(visible: Boolean): void {
 this.visible = visible; // 可见性
}

getVisible(): Boolean {
 return this.visible; // 可见性
}



ChatGPT

为了定义您提供的“MultiPolygon”内容的TypeScript接口，并保留注释，可以这样定义：

typescript

Copy code

```
// 表示地图上的多个多边形，可以自定义每个多边形的样式。  
interface MultiPolygon {  
    // 构造函数  
    new(options: MultiPolygonOptions): MultiPolygon;  
  
    // 方法  
    setMap(map: Map | null): this; // 设置地图对象，如果map为null意味着将多多边形同时从地图中移除。  
    setZIndex(zIndex: number): this; // 设置图层绘制顺序。  
    setGeometries(geometries: PolygonGeometry[] | null | undefined): this;  
    setStyles(styles: MultiPolygonStyleHash | null | undefined): this;  
    setVisible(visible: boolean): this; // 设置图层是否可见。  
    getMap(): Map | null; // 获取地图对象，若为空返回null。  
    getGeometries(): PolygonGeometry[]; // 获取多边形数据。  
}
```

// 辅助类型（需要根据实际实现来定义）
type MultiPolygonOptions = {
 // 多边形属性
};

```
getGeometries(): PolygonGeometry[] {  
    return this.geometries; // 多边形数据  
}  
  
getMap(): Map | null {  
    return this.map; // 地图对象  
}  
  
setVisible(visible: Boolean): void {  
    this.visible = visible; // 可见性  
}  
  
getVisible(): Boolean {  
    return this.visible; // 可见性  
}
```

我是前端开发工程师

```
1+ usages  ↗ kimishi
class MultiPolygon implements GeometryOverlay {
    constructor(options: MultiPolygonOptions) // 构造函数

    setMap(map: Map | null): this // 设置地图对象，如果map为null意味着将多折线同时从地图中移除。

    setZIndex(zIndex: number): this // 设置图层绘制顺序

    setGeometries(geometries: PolygonGeometry[]): this // 更新折线数据，如果参数为null或undefined，则移除所有折线。
    setStyles(styles: MultiPolygonStyleHash): this // 设置MultiPolyline图层相关样式信息，如果参数为null或undefined，则移除所有样式。
    setVisible(visible: boolean): this // 设置图层是否可见。
    setLoad(load: (load: any) => void): void \\ 去掉此段代码。
```

我是前端开发工程师

GDG西安 陕西省/西安市/雁塔区 西安

西安市 + 新增围栏

神州数码西安科技园

108°52'6"E 34°11'42"N

[编辑](#)

围栏名称	坐标	半径(米)	是否开启	操作
神州数码西安科技园	定位		开启	编辑 删除
西安交通大学	定位		开启	编辑 删除
西安建筑科技大学	定位		开启	编辑 删除

5个大不溜

公报私仇

我是安卓开发工程师



You

1. 有一些场景之下，需要使用摇杆控制汽车
2. 并不是实体摇杆和按键
3. 使用Android来连接汽车进行控制
4. Android使用Kotlin语言开发，不实用Compose，使用View
5. 摆杆有8个方向
6. 请给一段自定义View的代码

e. 建设一个自定义View的类

2. 摆杆有8个方向

我是安卓开发工程师



ChatGPT

创建一个自定义的View来实现一个有8个方向的摇杆控制器可以是一个有趣的挑战。在Android上，使用Kotlin语言，你会需要定义一个自定义的View类，并在其中处理触摸事件来判断摇杆的方向。这里有一个基本的代码示例来帮助你开始：

```
kotlin
class JoystickView(context: Context, attrs: AttributeSet) : View(context)

// 初始化时的一些设置
init {
    // 例如，初始化画笔、颜色等
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    // 绘制摇杆的逻辑
}

override fun onTouchEvent(event: MotionEvent): Boolean {
    when (event.action) {
        MotionEvent.ACTION_UP, MotionEvent.ACTION_DOWN, MotionEvent.ACTION_MOVE -> {
            handleTouchEvent(event)
        }
    }
    return true
}

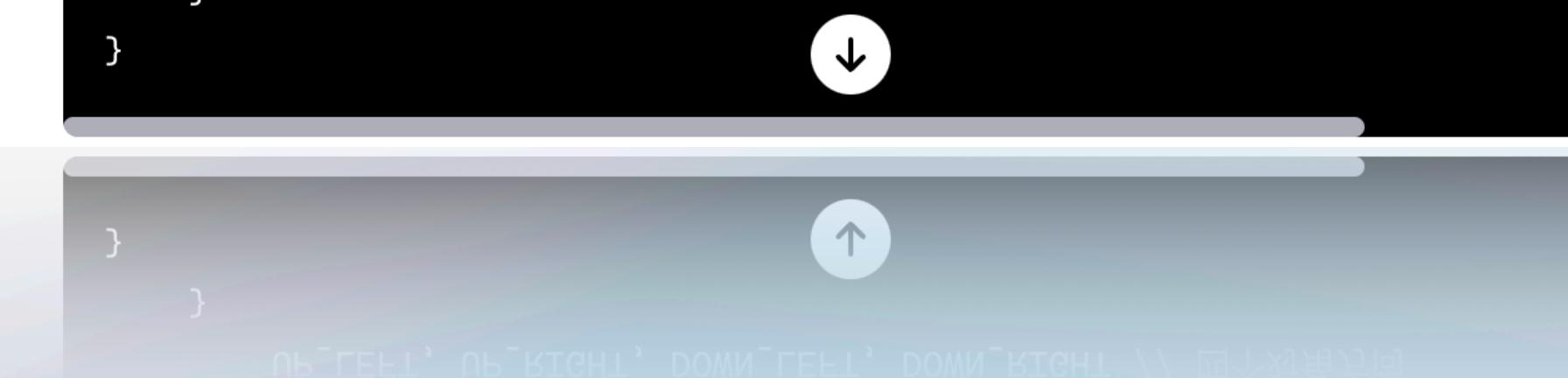
private fun handleTouchEvent(event: MotionEvent) {
    val direction = getDirection(event.x, event.y)
    // 可以在这里发送方向改变的通知
}
```

```
MotionEvent.ACTION_MOVE -> {
    // 处理移动逻辑，判断方向
    val direction = getDirection(event.x, event.y)
    // 可以在这里发送方向改变的通知
}
}

return true
}

private fun getDirection(x: Float, y: Float): Direction {
    // 根据x和y的值来判断方向
    // 返回一个代表方向的枚举或类
}

// 定义方向的枚举
enum class Direction {
    UP, DOWN, LEFT, RIGHT, // 四个主要方向
    UP_LEFT, UP_RIGHT, DOWN_LEFT, DOWN_RIGHT // 四个对角方向
}
```



我是安卓开发工程师



ChatGPT

为了确保摇杆 (knob) 不与方向箭头重叠，我们可以调整箭头和摇杆的尺寸以及它们在画布上的位置。主要的目标是使箭头位于摇杆的外侧且距离适中，从而避免重叠。这涉及对 `drawTriangleArrow` 方法和摇杆 (knob) 的绘制逻辑进行一些修改。

1. 调整三角形箭头的位置

您需要增加箭头与摇杆中心之间的距离，以确保箭头位于摇杆外侧。调整 `arrowRadius` 变量可以实现这一点：

```
kotlin
Copy code

private fun drawTriangleArrow(canvas: Canvas, angle: Double) {
    // ... 前面的代码 ...

    val arrowRadius = joystickRadius + 60 // 增加这个值以使箭头更远离摇杆中心
    // ... 绘制三角形箭头的代码 ...
}
```

2. 调整摇杆 (knob) 的半径

3. 在 `onDraw` 中绘制摇杆和箭头

```
// ... 绘制三角形箭头的代码 ...
}
```

2. 调整摇杆 (knob) 的半径

如果需要，您可以稍微减小摇杆的半径，以进一步减少与箭头的重叠可能性：

```
kotlin
Copy code

override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) {
    // ... 现有代码 ...

    knobRadius = joystickRadius / 2.5f // 可以稍微减小这个值
    // ... 其余代码 ...
}
```

3. 在 `onDraw` 中绘制摇杆和箭头

在 `onDraw` 方法中，先绘制摇杆背景和手柄，然后绘制箭头：

3. 在 `onDraw` 中绘制摇杆和箭头

我是安卓开发工程师

The screenshot shows the Android Studio interface with the code editor open to the file `JoystickView.kt`. The code implements a joystick control mechanism, calculating direction based on touch coordinates and handling motion events.

```
private fun calculateDirection(x: Float, y: Float): JoystickCommand {
    val dx = x - centerX
    val dy = y - centerY
    var angle = Math.toDegrees(atan2(dy.toDouble(), dx.toDouble()))
    if (angle < 0) angle += 360.0
    val index = ((angle + 22.5) % 360 / 45).toInt()
    return JoystickCommand.values()[index]
}

private fun handleActionDown(event: MotionEvent) {}

private fun handleActionMove(event: MotionEvent) {}

private fun handleActionUp() {}
```

我是安卓开发工程师

```
private fun notifyDirection(x: Float, y: Float) {
    listener?.let {
        val dx = x - centerX
        val dy = y - centerY
        val angle = Math.toDegrees(atan2(dy.toDouble(), dx.toDouble()))

        val direction = when {
            angle >= -22.5 && angle < 22.5 -> JoystickDirection.RIGHT
            angle >= 22.5 && angle < 67.5 -> JoystickDirection.UPPER_RIGHT
            angle >= 67.5 && angle < 112.5 -> JoystickDirection.UP
            angle >= 112.5 && angle < 157.5 -> JoystickDirection.UPPER_LEFT
            angle >= -157.5 && angle < -112.5 -> JoystickDirection.LOWER_LEFT
            angle >= -112.5 && angle < -67.5 -> JoystickDirection.DOWN
            angle >= -67.5 && angle < -22.5 -> JoystickDirection.LOWER_RIGHT
            else -> JoystickDirection.LEFT
        }
        it.onDirectionChanged(direction)
    }
}

if.onDirectionChanged(direction)
```

我是ROS/嵌入式开发工程师



You

好的，很不错，请将下发指令和上报指令进行一下整合，生成新的协议。



ChatGPT

当然，我将整合下发指令和上报指令，创建一个全面的蓝牙通信协议，以便在小程序和蓝牙小车之间进行有效通信。

完整蓝牙通信协议

下发指令

1. 连接蓝牙

- 指令：`CNL|CON`
- 描述：建立与小车的蓝牙连接。
- 格式：`[CNL|CON|<CRC32>]`
- 翻译：`[CNL|CON|<CRC32>]`
- 解释：`建立与小车的蓝牙连接`。

更多... [CNL|CON](#)

我是后端开发工程师

```
@dataclass
class Page:
    """
    分页数据结构
    """

    total: int = 0
    total_pages: int = 0
    has_prev: bool = False
    prev_page: int = 0
    has_next: bool = False
    next_page: int = 0
    first_page: int = 0
    last_page: int = 0
    page_count: int = 0
    current_page: int = 0
    per_page: int = 0
    pages: List[int] = field(default_factory=lambda: [])

    pages: List[int] = field(default_factory=lambda: [])
    per_page: int = 0
    current_page: int = 0
```



ChatGPT

将您的Python `dataclass` 转换为TypeScript的`interface`，并使用驼峰命名法以及在每个属性后添加注释，可以如下实现：

```
typescript
Copy code

interface Page {
    total: number; // 总记录数
    totalPages: number; // 总页数
    hasPrev: boolean; // 是否有前一页
    prevPage: number; // 前一页页码
    hasNext: boolean; // 是否有后一页
    nextPage: number; // 后一页页码
    firstPage: number; // 第一页页码
    lastPage: number; // 最后一页页码
    pageCount: number; // 页数计数
    currentPage: number; // 当前页码
    perPage: number; // 每页记录数
    pages: number[]; // 页码数组
}
```

这个TypeScript `interface` 保留了原Python `dataclass` 的所有属性，同时遵循了 TypeScript 的命名和类型注释惯例。每个属性后的注释用于说明该属性的用途或含义。

这个 TypeScript `interface` 保留了原 Python `dataclass` 的所有属性，并且遵循了 TypeScript 的命名和类型注释惯例。每个属性后的注释用于说明该属性的用途或含义。

我是后端开发工程师



You

请详细介绍一下cookiecutter



ChatGPT

Cookiecutter 是一个命令行工具，用于从预先制作的模板创建项目。它使用 Python 编写，但可以用于任何文本基础的项目模板（如 Python, C, Java 等）。Cookiecutter 可以帮助程序员快速创建项目结构，避免了从零开始的重复性工作。

主要特点包括：

1. **简单易用**: 通过简单的命令行接口，用户可以快速生成项目。
2. **模板定制**: 用户可以创建自己的模板或使用社区提供的模板。这些模板定义了项目的文件结构和必要文件。
3. **变量和逻辑**: 模板中可以包含变量和逻辑，使得生成的项目可以根据用户的输入进行定制。
4. **语言无关性**: 虽然 Cookiecutter 本身是用 Python 编写的，但它可以用来创建任何语言的项目模板。
5. **社区支持**: 有一个活跃的社区支持和维护 Cookiecutter，提供了大量现成的模板。

2. 社区支持：有一个活跃的社区支持和维护 Cookiecutter，提供了大量现成的模板。

项目模板。

3. 语言无关性：虽然 Cookiecutter 本身是用 Python 编写的，但它可以用来创建任何语言的项目模板。

使用 Cookiecutter 的典型步骤是：

1. **安装 Cookiecutter**: 可以通过 pip 安装。
2. **选择模板**: 选择一个已有的模板或创建一个新模板。
3. **生成项目**: 运行 Cookiecutter 并提供所需的参数，Cookiecutter 会根据模板生成项目结构。
4. **定制项目**: 在生成的项目基础上进一步开发和定制。

Cookiecutter 特别适合快速启动新项目，尤其是在需要遵循特定项目结构或编码标准时。通过使用模板，团队和个人开发者可以确保一致性和最佳实践的遵循。

4. 项目模板：通过使用模板，团队和个人开发者可以确保一致性和最佳实践的遵循。

Cookiecutter 特别适合快速启动新项目，尤其是在需要遵循特定项目结构或编码标准时。通过使用模板，团队和个人开发者可以确保一致性和最佳实践的遵循。

5. 社区支持：有一个活跃的社区支持和维护 Cookiecutter，提供了大量现成的模板。

我是后端开发工程师



You

请介绍一下DDD



ChatGPT

DDD（领域驱动设计）是一种软件开发方法论，专注于复杂系统的建模和设计。由 Eric Evans 在其 2003 年的书《领域驱动设计：软件核心复杂性的彻底解决》中首次提出。DDD 的核心理念是将注意力集中在软件项目的核心领域和领域逻辑上，而不是技术细节。

DDD 的关键概念：

1. **领域 (Domain)**：指软件项目要解决的业务领域，如电子商务、银行、保险等。
2. **子域 (Subdomain)**：大领域内的一个特定区域，每个子域都有其特定的业务逻辑。
3. **领域模型 (Domain Model)**：领域内的概念、术语和数据的结构化表示，反映了业务专家的知识和业务规则。
4. **限界上下文 (Bounded Context)**：明确领域模型的边界，确保模型内的术语和概念在这个上下文内是一致的。
5. **实体 (Entity) 和值对象 (Value Object)**：领域模型的两个主要构件，实体具有唯一标识，值对象则描述了领域的某些特征。

实体 (Entity) 和值对象 (Value Object)：领域模型的两个主要构件，实体具有唯一标识，值对象则描述了领域的某些特征。

6. **聚合 (Aggregate)**：一组相关对象的集合，通常由一个根实体和多个值对象组成，代表一个数据整体。
7. **领域服务 (Domain Service)**：实现领域逻辑但不属于实体或值对象的操作。
8. **领域事件 (Domain Event)**：反映领域内发生的重要业务事件。
9. **仓储 (Repository)**：提供访问聚合/实体的机制，通常与数据库交互。

DDD 的优势：

- **业务逻辑的清晰表示**：通过领域模型，可以更清晰地表示和理解复杂的业务逻辑。
- **改善开发者与业务专家的沟通**：使用统一的领域语言 (Ubiquitous Language)，促进团队成员之间的有效沟通。
- **适应性强**：领域驱动的方法可以更容易地适应业务需求的变化。

DDD 的挑战：

- **学习曲线**：理解和应用 DDD 概念需要时间和实践。
- **适用性**：对于简单应用或非业务密集型系统，DDD 可能是过度设计。
- **实现复杂性**：正确实现 DDD 需要深入理解其理论和实践，可能会增加项目的技术复杂性。

DDD 并不是一个具体的技术框架或工具集，而是一种设计方法论，其核心是深入理解和精心建模业务领域，以支持复杂软件系统的有效开发。

DDD 并不是一个具体的技术框架或工具集，而是一种设计方法论，其核心是深入理解和精心建模业务领域，以支持复杂软件系统的有效开发。

DDD 并不是一个具体的技术框架或工具集，而是一种设计方法论，其核心是深入理解和精心建模业务领域，以支持复杂软件系统的有效开发。

我是后端开发工程师



You

使用DDD理念进行开发，我的文件结构如下 package: `__init__.py`, `models.py`, `schemas.py`, `queries.py`, `repositories.py`, `services.py`, 请帮我总结一下



ChatGPT

使用领域驱动设计（DDD）理念进行开发时，您的文件结构包括了一些关键的组件，每个组件都扮演着特定的角色。以下是对您提供的文件结构的总结：

1. `__init__.py`:

- 这个文件使得您的包可以作为一个 Python 模块被导入。
- 它通常为空或者用于初始化代码，如设置必要的路径或者初始化包级别的状态。

2. `models.py`:

- 包含您的领域模型，这通常包括定义实体（Entities）和值对象（Value Objects）。
- 这些模型反映了业务领域的核心概念和规则。
- 实体通常具有唯一标识符，而值对象则代表没有身份的领域概念。

3. `schemas.py`:

- 可能包含数据传输对象（DTOs）或者视图模型。
- 这些模式用于数据的序列化和反序列化，通常在与外部系统（如前端或API）交互

- 例如，它们可能表示从数据库中检索的数据（如实体）或向数据库插入的数据（如命令）。
- 它们可能表示从外部系统接收的数据（如 DTOs 或视图模型）或向外部系统发送的数据（如命令）。

4. `queries.py`:

4. `queries.py`:

- 包含与查询相关的逻辑。
- 这可能包括读取数据的复杂查询操作，特别是那些涉及到多个模型或需要特定查询优化的。
- 在 CQRS（命令查询责任分离）模式中，这些查询与命令（修改数据的操作）分开。

5. `repositories.py`:

- 包含与存储和检索领域实体相关的代码。
- 仓储抽象使得领域层与数据持久化机制（如数据库）之间的耦合最小化。
- 它提供了一个简单的接口来获取和保存领域实体，而无需知道底层的数据存储逻辑。

6. `services.py`:

- 包含领域服务（Domain Services），这些是实现领域逻辑的地方，但不自然属于任何单个实体或值对象。
- 领域服务通常执行跨多个模型的操作，或者实现不适合放在实体或值对象中的复杂业务逻辑。
- 它们也可以用于处理领域事件和与其他限界上下文的交互。

总体而言，这种文件结构清晰地分离了领域模型（实体和值对象）、应用逻辑（服务和仓储）以及与外部界面的交互（模式和查询）。这有助于维护清晰的架构边界，促进代码的模块化和可维护性。

总体而言，这种文件结构清晰地分离了领域模型（实体和值对象）、应用逻辑（服务和仓储）以及与外部界面的交互（模式和查询）。这有助于维护清晰的架构边界，促进代码的模块化和可维护性。

总体而言，这种文件结构清晰地分离了领域模型（实体和值对象）、应用逻辑（服务和仓储）以及与外部界面的交互（模式和查询）。这有助于维护清晰的架构边界，促进代码的模块化和可维护性。

我是后端开发工程师



You

我的Class命名方式举例如下 UserModel(IdModel), UserRepository(BaseRepository), UserService(BaseService)



ChatGPT

您的类命名方式遵循了一种清晰的、遵循领域驱动设计（DDD）原则的模式。以下是您给出的例子的解释：

1. `UserModel (IdModel)`:

- `UserModel` 是一个领域模型，代表了用户这一领域概念。
- 它继承自 `IdModel`，这表明 `UserModel` 包括一个标识符（ID），这是 DDD 中实体的一个典型特征。
- 这个类可能包含用户的属性（如用户名、密码等）和行为（如验证用户数据的方法）。

2. `UserRepository (BaseRepository)`:

- `UserRepository` 是一个仓储类，用于抽象化用户数据的存取操作。
- 继承自 `BaseRepository`，表明它实现了一些基本的仓储操作，如增加、删除、查找用户等。
- 该类的作用是使得领域层（如服务层）与数据持久化层（如数据库操作）解耦，提高了代码的可维护性和可测试性。

通过这种方式，领域层和持久化层被分离。

- 通过这种方式，领域层和持久化层被分离，从而提高了系统的可维护性和可扩展性。



You

我有一个文件夹叫做domain，其中按照业务名称来存放上述文件，比如user, order等



ChatGPT

您的项目结构安排表明您正在实现领域驱动设计（DDD）中的领域层，并且按照不同的业务领域（如用户、订单等）来组织代码。这种做法有利于分隔不同业务领域的逻辑，增强代码的模块性和可维护性。以下是对您描述的项目结构的具体解释：

文件夹结构

1. `domain` 文件夹:

- 这个文件夹作为项目中所有领域逻辑的根容器。
- 它包含了不同子域的代码，每个子域负责处理特定的业务领域逻辑。

2. 子文件夹（例如 `user`, `order` 等）:

- 每个子文件夹代表一个特定的业务子域或限界上下文。
- 这种分隔方式有助于隔离不同业务逻辑，使得项目更易于管理和扩展。

• 通过这种方式，领域层和持久化层被分离，从而提高了系统的可维护性和可扩展性。

• 每个子文件夹代表一个特定的业务子域或限界上下文。

3. 其他文件夹 (如 `user`, `order` 相关文件):

我是后端开发工程师

每个子域内的文件

假设我们考虑 `user` 和 `order` 两个子文件夹，每个文件夹可能包含以下文件：

- `__init__.py`: 使文件夹成为 Python 包。
- `models.py`: 定义该子域的领域模型，如 `UserModel` 和 `OrderModel`。
- `schemas.py`: 用于数据序列化的模式，如 `UserSchema` 和 `OrderSchema`。
- `queries.py`: 包含特定于该子域的查询逻辑，如用户查询或订单查询。
- `repositories.py`: 为该子域的实体提供数据访问逻辑，如 `UserRepository` 和 `OrderRepository`。
- `services.py`: 包含实现该子域业务规则和操作的服务，如 `UserService` 和 `OrderService`。

优势

- **清晰的业务逻辑分离**: 每个子域专注于其特定的业务逻辑，减少了代码间的耦合。
- **可扩展性**: 新的业务需求可以通过添加新的子域文件夹来轻松实现。
- **易于维护和理解**: 新团队成员可以快速理解项目结构和各部分的职责。

注意事项

注意事项

- **保持一致性**: 在整个 `domain` 文件夹中保持一致的命名和结构可以大大提高代码的可读性。
- **避免过度工程**: 根据项目的复杂性和团队规模，适当地应用 DDD。对于小型或中等复杂度的项目，过于复杂的 DDD 结构可能会导致不必要的开销。

总体来说，这种按业务划分的领域层组织方式是实施 DDD 的一个很好的实践，有助于构建清晰、可维护和可扩展的系统。

我是后端开发工程师



You

根据之前的描述，请使用cookiecutter帮我生成一个脚手架：

1. 假如我输入user，请生成以这个名字命名的pacakge，里面包含必要的ddd文件
2. 不同文件中的命名，使用pascal case，比如UserModel, UserRepository等
3. 保存这个脚本相关的所有文件
4. 详细介绍一下如何使用cookiecutter



ChatGPT

要使用 Cookiecutter 创建一个根据您的需求定制的脚手架，我们需要遵循几个步骤。首先，我将创建一个 Cookiecutter 模板，然后我将展示如何使用这个模板来生成您需要的项目结构。由于我无法直接在此环境中运行 Cookiecutter，我会提供必要的文件和说明，供您在您自己的环境中执行。

步骤 1: 创建 Cookiecutter 模板

我们需要创建一个 Cookiecutter 模板。这通常包括两个主要部分：模板文件和一个配置文件（`cookiecutter.json`）。

文件（`cookiecutter.json`）。

我们创建一个名为“domain”的新文件夹。在该文件夹中，我们将创建一个名为“domain”的新文件夹。

在“domain”文件夹中，我们将创建一个名为“domain”的新文件夹。

模板文件结构

markdown

Copy code

```
{{ cookiecutter.domain_name }}/  
    __init__.py  
    models.py  
    schemas.py  
    queries.py  
    repositories.py  
    services.py
```

在这个结构中，`{{ cookiecutter.domain_name }}` 是一个变量，它将在您使用模板时被替换为您输入的领域名（例如 `user`）。

`cookiecutter.json`

json

Copy code

```
{  
    "domain_name": "domain"  
}  
  
    "domain_name": "domain"  
{
```

我是后端开发工程师

步骤 2: 模板文件内容

每个文件需要有基础的内容来反映 DDD 结构。例如：

- `models.py`:

```
python
Copy code
class {{ cookiecutter.domain_name|capitalize }}Model:
    pass
```

- `repositories.py`:

```
python
Copy code
class {{ cookiecutter.domain_name|capitalize }}Repository:
    pass
```

- 类似的，其他文件也会遵循这个模式。

步骤 3: 使用模板

1. 安装 Cookiecutter (如果您尚未安装) :

```
bash
Copy code
pip install cookiecutter
```

2. 使用模板生成项目：

```
bash
Copy code
cookiecutter path_to_template
```

```
cookiecutter bash-to-template
Copy code
path
5. 项目名称: 上述命令中运行以上命令:
```

在这里，`path_to_template` 是您存放 Cookiecutter 模板的路径。

3. 当提示输入 `domain_name` 时，输入例如 `user`。
4. Cookiecutter 将创建一个以您提供的领域名命名的文件夹，并包含所有必要的文件。

步骤 4: 保存和提供模板

由于我无法在这里运行脚本，我会将模板文件保存到一个可下载的压缩包中，您可以在本地使用它。让我先做好这些模板文件。

我已经创建了一个 Cookiecutter 模板，并将其保存为一个压缩包。您可以通过以下链接下载它：

[下载 Cookiecutter DDD 模板](#)

如何使用这个模板

1. **下载并解压缩模板**: 首先，下载上面提供的压缩包并在您的计算机上解压缩。
2. **安装 Cookiecutter** (如果您还没安装的话) :

在命令行中运行以下命令：

在命令行中运行以上命令：

3. 安装 Cookiecutter (如果尚未安装的话) :

在命令行中运行以下命令：

我是后端开发工程师

bash

Copy code

```
pip install cookiecutter
```

3. 使用模板生成项目：

在命令行中导航到解压缩的模板所在的目录，并运行以下命令：

bash

Copy code

```
cookiecutter path_to_template_directory
```

其中 `path_to_template_directory` 是解压缩后模板的文件夹路径。

4. 按提示输入：

当提示输入 `domain_name` 时（例如：`user` 或 `order`），输入您想要的领域名。

5. 生成的项目：

Cookiecutter 将根据您提供的领域名和模板内容，在您的当前目录下创建一个新的文件夹。

注意事项

- 在使用模板之前，确保您有 Python 环境，并且已安装 Cookiecutter。
- 如果您需要对模板进行更多定制，可以编辑解压后的模板文件。

使用 Cookiecutter 的优势在于它的自动化和可定制性，可以快速生成符合您特定需求的项目结构。[[-](#)]

项目结构 [[-](#)]

该部分展示了使用 Cookiecutter 生成的项目结构示例，包括根目录下的文件夹和子文件夹。

- 如果要对某些文件进行定制，可以在项目目录中找到相应的文件夹。

Cookiecutter 会根据模板文件中的配置自动处理这些文件。

06 - 未来已来

Gemini 是谷歌最大最强的模型



Gemini Ultra 2024年初会上线

Look out for Gemini Ultra in an advanced version of Bard early next year

Gemini Ultra is our largest and most capable model, designed for highly complex tasks and built to quickly understand and act on different types of information — including text, images, audio, video and code.

One of the first ways you'll be able to try Gemini Ultra is through Bard Advanced, a new, cutting-edge AI experience in Bard that gives you access to our best models and capabilities. We're currently completing extensive safety checks and will launch a trusted tester program soon before opening Bard Advanced up to more people early next year.

This aligns with the bold and responsible approach we've taken since Bard launched. We've built safety into Bard based on our [AI Principles](#), including adding contextual help, like Bard's "[Google it](#)" button to more easily double-check its answers. And as we continue to fine-tune Bard, your feedback will help us improve.

more easily double-check its answers. As we continue to fine-tune Bard, your feedback will help us improve.

As we continue to fine-tune Bard, your feedback will help us improve.



Practical Deep Learning for Coders 2022

A complete from-scratch rewrite of fast.ai's most popular course, that's been 2 years in the making.

COURSES

AUTHOR

Jeremy Howard

PUBLISHED

July 21, 2022

A new edition

Today we're releasing [Practical Deep Learning for Coders 2022](#)—a complete from-scratch rewrite of fast.ai's most popular course, that's been two years in the making. Previous fast.ai courses have been studied by hundreds of thousands of students, from all walks of life, from all parts of the world. fast.ai's videos have been viewed over 6,000,000 times already! The major differences are:

- A much bigger focus on interactive explorations. Students in the course build simple GUIs for building decision trees, linear classifiers, and non-linear models by hand, using that experience to develop a deep intuitive understanding of how foundational algorithms work
- A broader mix of libraries and services are used, including the Hugging Face ecosystem (Transformers, Datasets, Spaces, and the Model Hub), Scikit Learn, and Gradio
- Coverage of new architectures, such as ConvNeXt, Visual Transformers (ViT), and DeBERTa v3

By the end of the second lesson, students will have built and deployed their own deep learning model on their own data. Many students post their course projects [to our forum](#). For instance, if there's an unknown dinosaur in your backyard, maybe you need this [dinosaur classifier](#)!

On this page

- [A new edition](#)
- [About the course](#)
- [Students and results](#)
- [About deep learning](#)
- [The lessons](#)
- [A vibrant community](#)
- [Get started](#)

来源：<https://www.fast.ai/posts/2022-07-21-dl-coders-22.html>

每个人都可以成为MVP