中国矿业大学计算机学院

2022 级本科生课程报告

课程名称 面向对象分析与设计

报告时间 2025 年 6 月

学生姓名 杨晓琦

学　　号 08222213

专　　业 计算机科学与技术

任课教师 张 磊

目　　录

1 系统愿景 1
2 需求建模 3
3 补充性规格说明(FURPS+模型) 6
4 词汇表 7
5 业务规则 10
6 领域类图 12
7 系统顺序图(提交订单) 13
8 操作契约(提交订单) 15
9 分层系统架构 16
10 组件图和部署图 19
11 设计模式优化前后对比 23
12 附录(MOOC 学习记录) 27

1 系统愿景

1.1 业务背景

美团外卖是一个高并发、高吞吐量的实时交易与配送平台，连接用户、商家与骑手三方角色，承载着全国范围内亿级用户的本地即时配送需求。其下单与配送子系统是平台运行的核心模块之一，承担从用户下单、商家接单、平台派单、骑手配送到最终交付的全链路任务。

该系统不仅涉及订单创建、支付、分发与状态更新，还要支持实时位置追踪、动态调度与高并发访问请求，尤其在业务高峰（如中午 11:30–13:30，晚上 17:30–19:30）时段，系统需保持毫秒级响应、秒级调度，在千万级并发订单处理压力下仍保持稳定、精准与可靠。

1.2 核心业务目标

1.2.1 支持海量订单并发

1) 系统需稳定支持日均千万级订单处理能力；
2) 高峰时段需支撑每秒≥50,000 笔订单请求（QPS ≥ 5W），同时完成商品校验、优惠券计算、库存锁定与支付请求初始化等环节；
3) 所有操作需在异步队列、缓存系统与并发写入机制下完成，保障订单完整入库，防止丢单/重复单。

1.2.2 确保下单、支付与派单流程的低延迟

1) 用户点击“下单”至支付完成，整个链路响应时间需控制在≤300ms；
2) 商家收到订单响应时间 ≤1s；
3) 派单系统完成智能决策，分配给合适骑手耗时不超过 500ms；
4) 跨模块需使用消息队列、服务注册发现与异步调度中心减少阻塞调用。

1.2.3 实时订单追踪与交互

1) 用户可实时查看订单状态、骑手位置、预计送达时间；
2) 位置数据从骑手设备上传至服务端，刷新到用户端页面延迟需≤2s；
3) 系统需实现高频数据流采集+低延迟渲染架构（如 WebSocket + 消息中间件）；

1.2.4 保证系统高可用与故障容错能力

1) 面对部分节点异常、网络丢包等情况，系统需自动降级，保持核心下单流程可用；
2) 通过容器化部署 + 自动扩缩容策略，系统可在分钟级快速扩容；
3) 各子系统需支持故障隔离、熔断、服务限流与重试机制，防止“雪崩效应”。

1.3 核心功能范围

表 1 不同角色及其功能

角色	核心功能	性能要求
用户	- 浏览餐厅与菜单（库存实时） - 下单结算（商品 + 优惠券）	- 下单操作完成 ≤300ms - GPS 更新展示延迟 ≤2s

商家	- 支付提交与订单确认 - 实时追踪骑手位置与送达时间 - 实时接收新订单通知 - 确认接单 / 出餐时间更新 - 管理库存与促销活动	- 订单推送延迟 ≤1s - 出餐状态响应 ≤500ms
骑手	- 接收就近订单推送 - 上报配送位置与状态 - 处理异常事件（如超时、取消）	- 派单决策耗时 ≤500ms - 位置信息上传频率 ≥5s
平台系统	- 实时智能派单（基于距离、商户准备情况） - 高峰期运力调度与区域调价 - 风控规则校验、防刷单处理 - 监控系统健康与可用性	- 调度计算与分发 ≤1s - 实时监控更新周期 ≤5s

1.4 关键业务指标（KPI）

表 2 不同指标类别及其目标

指标类别	目标
吞吐量	支持 ≥10000 TPS 的订单写入、状态更新与派单操作
响应延迟	- 用户从下单到支付完成 ≤300ms - 商户接单响应 ≤1s - 派单完成 ≤500ms
可用性	系统 SLA ≥ 99.99%（年故障时间 < 1 小时）
数据一致性	订单状态最终一致，确保无漏单、无重复派单、无超卖现象
数据刷新频率	- 骑手位置上传间隔 ≤5s - 用户前端地图刷新 ≤2s

1.5 关键业务挑战

1.5.1 瞬时高峰请求压力

外卖平台常在节假日、饭点、满减活动期间面临“爆单”现象，大量用户在短时间内集中下单，带来系统级流量冲击。此类瞬时高并发请求对系统的稳定性、容量和处理能力提出极高要求，若处理不及时，极易导致用户下单失败、商家响应超时、平台页面卡顿等问题，严重影响用户体验与平台信誉。

1.5.2 资源抢占下的骑手调度公平性

同一时间内，多个订单需匹配周边有限的骑手资源，尤其在高峰时段或骑手稀缺区域，存在订单派发拥堵、调度效率下降、骑手争抢订单等问题。平台需在保证配送效率的同时，兼顾调度公平、用户等待时间与骑手工作体验之间的平衡。

1.5.3 实时交互体验的高要求

用户对外卖配送的期望不仅仅是“准时送达”，更希望实时可见。例如下单后的状态

更新、骑手的位置变动、预计送达时间等，都需要在页面上即时反映。一旦信息滞后或出现“数据断档”，用户容易产生焦虑甚至投诉，因此平台需保障交互数据的持续更新与稳定可用。

1.5.4 多角色协同中的一致性问题

下单与配送涉及用户、商家、骑手三个角色，系统需在订单创建、支付确认、配送分配等环节维持一致性。例如订单状态不能出现“已支付但未生成订单”、“商家已接单但骑手未收到”等异常，这对系统的数据协调能力、响应一致性提出了较高要求，尤其在并发量大或节点异常时更容易发生混乱。

1.5.5 高可用与业务连续性的保障

在实际运行中，平台难免面临服务器故障、网络波动、流量突增等突发事件。此时系统需能及时进行局部恢复、流量引导或降级处理，避免大范围服务中断，保障核心业务流程（如下单、配送）持续运行。这不仅是技术挑战，更是对业务流程韧性与容错设计的综合考验。

1.6 预期商业价值

1) 提升用户体验：快速下单、实时可视 → 提升下单成功率、缩短等待时间，促进复购；
2) 优化运营效率：智能调度骑手，减少空跑与超时，提升单均产能；
3) 增强平台稳定性与容灾能力：支持突发高并发场景（如“双十一”“雨天潮汐流量”），保障平台收益；
4) 支持业务增长弹性：系统具备良好水平扩展能力，可支撑未来百万订单/分钟级别业务增长。

1.7 产品展望

1) 2025 年实现智能动态定价
2) 2026 年实现无人机配送试点
3) 2027 年实现全链路 AI 客服

2 需求建模

2.1 功能需求描述

表 3 功能需求描述

编号	功能名称	简要描述
FR-01	浏览餐厅与菜单	用户查看附近可配送商家、菜单、库存与促销信息
FR-02	提交订单	用户选购商品后提交订单，系统需校验库存计算优惠
FR-03	支付订单	用户完成支付，系统生成订单并进入配送流程
FR-04	商家接单与出餐	商家接收订单后确认制作并更新出餐状态
FR-05	订单派发	系统根据位置、时效、运力等因素将订单指派给骑手
FR-06	骑手接单与配送	骑手接收派单并进行配送，支持位置更新与状态上报
FR-07	用户实时追踪订单	用户可查看订单状态、骑手位置与预计送达时间

FR-08	订单完成与评价	用户签收后可对本次服务进行评价

2.2 非功能需求描述

表 4 非功能需求描述

编号	非功能需求	描述
NFR-01	高并发支持	系统应支持 ≥50,000 QPS 的订单请求与状态更新
NFR-02	响应时延	核心操作如“提交订单”“接单”响应时间≤300ms（P99）
NFR-03	实时数据更新	用户端位置追踪与状态更新延迟 ≤2s
NFR-04	系统可用性	年可用性 ≥99.99%，支持服务降级与快速恢复机制
NFR-05	扩展性	系统架构应支持根据业务增长弹性扩容
NFR-06	一致性保障	支持最终一致性，避免重复订单、漏派单等现象

2.3 用例图

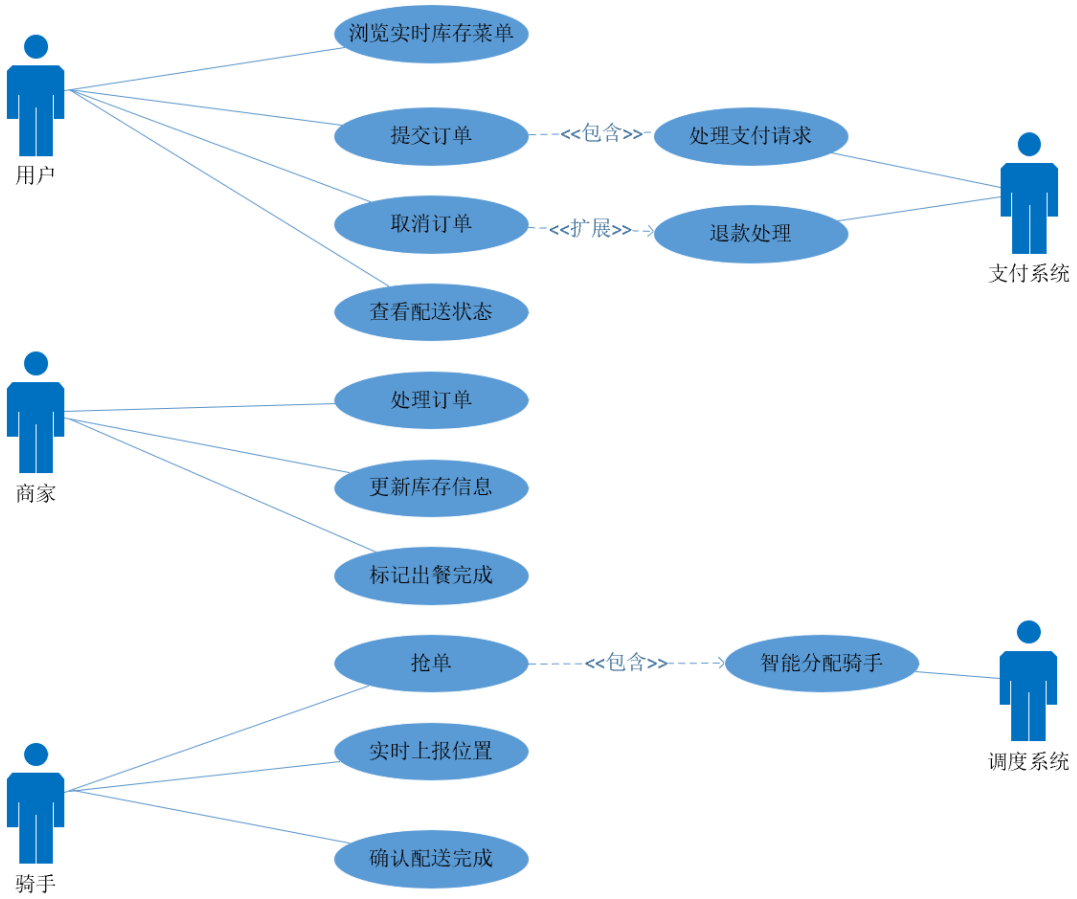

图 1 下单与配送用例图

2.4 用例描述(提交订单)

- 用例名称：提交订单
- 所属系统：美团外卖下单与配送系统
- 主要参与者：用户
- 正常流程：

1) 用户在 APP 中选择商品、配送地址和支付方式，点击"提交订单"按钮。
2) 系统生成订单 ID，并实时校验库存。
3) 系统计算订单金额，校验优惠券有效性；计算满减、配送费；生成最终支付金额。
4) 系统调用支付系统，系统跳转至支付网关（支付宝/微信），同时，本地记录订单状态
5) 支付系统回调通知成功后，系统更新订单状态为"已支付"；记录库存扣减最终状态。
6) 系统返回订单创建成功页，包含：订单编号、预计送达时间；支付金额和优惠明细。
7) 用户收到订单创建成功页。

- 候选流程：

*a. 订单服务或支付服务不可用。
   1. 系统记录错误日志并触发告警。
   2. 页面显示："系统繁忙，请稍后重试"。
   3. 用户可稍后重新尝试提交订单。

*b. 库存扣减成功但支付状态未更新
   1. 系统定时任务检测异常订单状态。
   2. 自动触发库存回滚操作。
   3. 通知运营人员人工核查。

1a. 用户短时间内重复提交相同商品的订单请求。
   1. 系统检测到重复请求（基于用户 ID+商品 ID+时间窗口）。
   2. 系统返回提示："您已提交过该订单，请勿重复操作"。
   3. 页面禁用提交按钮 5 秒。

2a. 系统校验库存时发现库存不足。
   1. 系统立即终止订单流程，不生成订单 ID。
   2. 系统向用户返回错误提示："您选择的商品库存不足，请重新选择"。
   3. 页面自动刷新商品库存显示。

3a. 系统校验优惠券时发现优惠券已过期或不符合使用条件。
   1. 系统保留生成的订单 ID。
   2. 系统向用户返回提示："您使用的优惠券已失效，请重新选择支付方式"。
   3. 用户可选择更换优惠券或继续支付。

4a. 用户跳转至支付网关后，15 分钟内未完成支付操作。
   1. 系统自动将订单状态标记为"支付超时"。
   2. 系统释放预扣减的库存。
   3. 系统向用户推送通知："订单支付超时已自动取消"。

5a. 用户主动取消支付

1. 系统更新订单状态为"支付失败"。
2. 系统释放预扣减的库存。
3. 系统向用户返回提示："支付失败，请重新尝试或选择其他支付方式"。
4. 用户重新发起支付流程。

3 补充性规格说明(FURPS+模型)

3.1Functionality（功能性）

1) 支持用户在线浏览商家与菜单、提交订单、支付订单、实时查看配送状态。
2) 支持平台系统自动调度派单，分发订单至商家与骑手。
3) 商家可接收订单并更新出餐状态，骑手可上报配送状态及异常。
4) 提供完整的订单生命周期管理（创建 → 配送 → 完成 → 评价）。
5) 所有功能支持高并发并发起，系统需处理≥10000 TPS 的操作压力。

3.2Usability（易用性）

1) 用户端界面需清晰直观，3 步内完成下单流程（选商品→提交→支付）。
2) 订单追踪界面应实时更新骑手位置与预计送达时间，刷新延迟 ≤2s。
3) 商家后台操作简洁，订单提醒需及时、明确，具备语音提醒功能。
4) 骑手 App 提供大按钮、自动接单、高对比度视觉设计，便于户外使用。
5) 所有角色使用系统无需复杂培训，界面需符合常规交互习惯。

3.3Reliability（可靠性）

1) 平台全年可用性 ≥99.99%，单次故障恢复时间 ≤5 分钟。
2) 订单处理过程中，所有状态必须最终一致，严禁出现订单丢失、重复派发、异常状态切断等情况。
3) 对订单流程关键步骤（如支付成功、接单成功）均需持久化日志追踪，支持人工追溯。
4) 在数据库故障或服务宕机时，系统应具备局部降级能力，例如启用缓存响应或进入只读模式，避免全链路瘫痪。

3.4Performance（性能）

本系统最核心要求之一。

1) 高吞吐：系统支持 ≥50,000 QPS 并发请求（提交订单 + 状态上报 + 派单响应等）。
2) 低延迟：

a) 提交订单 ≤300ms（P99）
b) 骑手接单响应 ≤500ms
c) 状态上报到前端显示 ≤2 秒

3) 扩展性：平台应支持根据业务压力进行水平扩容，调度节点无单点瓶颈。
4) 资源使用效率：高峰期 CPU 利用率不超过 80%，Redis/QPS 控制在容量阈值内。

3.5Supportability（可支持性）

1) 系统需支持热升级与滚动部署，不影响现有用户下单。
2) 提供完善的日志系统（订单链路追踪、性能监控、告警推送）。
3) 系统应开放接口供其他业务模块或外部合作方接入（如商家系统、配送合作方）。
4) 各类配置应可通过平台配置中心动态调整，无需停机重启。
5) 支持 A/B 测试与灰度发布。

3.6 "+" 类属性（其他补充）

1) Security(安全性):用户下单和支付过程必须采用 HTTPS；敏感数据（如支付信息）需加密传输与存储；每个用户接口需校验身份权限。
2) Maintainability(可维护性):模块解耦清晰，支持按业务子模块独立升级与监控。新接入商家/骑手无需代码改动。
3) Scalability(可扩展性):支持服务集群自动扩容（K8s 部署），支撑用户量/骑手量级别增长。
4) Compliance(合规性):所有交易数据满足中国大陆电商平台数据合规要求，可接受审计。
5) Localization(本地化):系统预留多语言支持，适应海外业务拓展；时间/单位/币种适配不同城市规范。
6) Internationalization(国际化):支持多语言（中/英文）和多币种结算。

4 词汇表

4.1 角色类术语（Actors）

表 5 角色类术语

术语	定义	示例 / 说明
用户（User）	在 App 上浏览、下单、支付、追踪订单的终端消费者	使用美团 App 下单的顾客
商家（Merchant）	在平台上展示商品、接	如餐厅、便利店、饮品店

	单、制作与交付商品的经营主体	
骑手（Rider）	执行订单配送任务的工作人员	使用骑手 App，接单并配送
平台系统（Platform System）	提供调度、派单、监控等服务的后台系统	包括订单系统、调度引擎等

4.2 实体类术语（Business Entities）

表 6 实体类术语

术语	定义	示例 / 说明
订单（Order）	用户提交后由平台处理、商家制作、骑手配送的交易记录	包含商品、价格、地址等
商品（Item）	商家提供给用户选择购买的食物或物品	如炸鸡、奶茶等
地址信息（Address）	用户下单时填写的收货地址	包含楼栋、门牌号等
配送路线（Route）	骑手从商家取餐至用户的最佳路径	由平台计算
优惠券（Coupon）	可抵扣部分订单费用的营销工具	满减券、折扣券等

4.3 动作 / 用例类术语（Actions / Use Cases）

表 7 用例类术语

术语	定义	示例 / 说明
提交订单（Submit Order）	用户选定商品后正式发起订单请求	含库存校验、价格计算等
支付订单（Pay Order）	用户完成订单付款的行为	使用微信、支付宝等
接收订单（Accept Order）	商家确认收到订单并准备制作	改变订单状态为“已接单”
订单派发（Dispatch Order）	平台将订单指派给合适骑手的行为	基于距离、时间、热力图等
位置上报（Report Position）	骑手周期性上传自己的实时定位	用户可追踪
异常上报（Report Exception）	骑手在配送过程中遇到异常并反馈	如超时、道路封闭

4.4 状态类术语（States）

表 8 状态类术语

术语	定义	示例 / 说明
订单状态（Order Status）	表示订单在处理流程中的位置	如“已支付”“配送中”
出餐状态（Meal Status）	商家端标记制作进度的状态	如“制作中”“已打包”
配送状态（Delivery Status）	骑手端标记配送过程的状态	如“已取餐”“送达中”
异常状态（Abnormal Status）	表示订单处理过程中的非正常事件	如“配送超时”“骑手迷路”

4.5 性能类术语（Performance Terms）

表 9 性能类术语

术语	定义	示例 / 说明
吞吐量（QPS/TPS）	每秒处理的请求/事务数量	平台高峰期需支持 ≥50,000 QPS
响应延迟（Latency）	从用户发出请求到系统响应所需时间	提交订单 ≤300ms
实时性（Real-Time Feedback）	系统对数据更新的快速反应能力	骑手位置刷新 ≤2s
高可用性（Availability）	系统全年正常运行的比例	要求 ≥99.99%

4.6 系统特性 / 架构类术语（System Qualities）

表 10 架构类术语

术语	定义	示例 / 说明
幂等性（Idempotency）	多次提交同一请求，结果保持一致	提交订单接口需避免重复下单
数据一致性（Consistency）	不同模块间的状态信息协调一致	支付成功后订单状态应同步更新
降级机制（Degradation Strategy）	异常情况下关闭次要功能保证主流程可用	实时追踪故障时仍可下单
灰度发布（Gray Release）	部分用户提前体验新功能的发布策略	用于 A/B 测试新算法
扩展性（Scalability）	系统根据业务规模可弹性扩展资源	水平扩展订单处理节点
日志追踪（Tracing）	对订单全过程进行日志记录	便于定位问题、业务审计

5 业务规则

5.1 数据类规则（Data Rules）

表 11 数据类规则

编号	规则内容	说明
ID-01	用户下单前必须选择有效地址、收货人与联系方式	系统需验证地址格式合法、区域可配送
ID-02	商家的商品库存必须≥用户下单数量	系统下单前需校验并冻结库存，防止超卖
ID-03	同一优惠券每位用户仅可使用一次	需依赖用户唯一标识进行限制
ID-04	骑手位置上报频率为 5 秒 / 次	保障实时性与带宽资源之间的平衡

5.2 流程类规则（Process Rules）

表 12 流程类规则

编号	规则内容	说明
ID-05	订单在“已支付”状态前不能被派单	平台调度系统需等待支付确认事件
ID-06	商家必须在 2 分钟内接单，否则系统自动转单或取消	保证服务响应效率，避免用户等待
ID-07	商家标记“已出餐”后，订单方可由骑手取餐	强制出餐节点，避免骑手空跑
ID-08	订单状态必须严格按流程流转（如不能跳过“配送中”）	保证订单生命周期完整性与可追踪性

5.3 调度类规则（Dispatch / Matching）

表 13 调度类规则

编号	规则内容	说明
ID-09	派单优先选取距离商家 3km 内空闲骑手	平衡效率与时效，避免资源浪费
ID-10	骑手同一时间最多同时处理 3 单（视业务类型而定）	控制骑手负载，提高配送成功率
ID-11	高峰期调度算法优先考虑预计送达时间	提升用户体验，避免超时赔付

ID-12	多骑手同时候选时按评分 + 距离 +响应时延综合决策	避免抢单冲突，实现公平分配

5.4 限制类规则（Constraint Rules）

表 14 限制类规则

编号	规则内容	说明
ID-13	一次性最多可购买 20 件商品	防止批量囤单、刷单行为
ID-14	系统限制每分钟每用户最多提交 3 次订单	限流保护，防止恶意重复提交
ID-15	同一手机号 24 小时内最多下单 10 次	辅助风控逻辑
ID-16	每个订单最多可使用 1 张优惠券	营销规则统一口径处理

5.5 安全与合规规则（Security / Compliance）

表 15 安全类规则

编号	规则内容	说明
ID-17	用户支付需走加密通道，支持 HTTPS、数字签名	防止中间人攻击
ID-18	所有订单与交易数据需保留至少 180 天	满足审计要求与纠纷处理
ID-19	用户个人信息必须加密存储，不得明文展示	涉及手机号、地址、姓名等
ID-20	异常订单（如支付失败、配送超时）需记录详细审计日志	支持事后追责与问题回溯

6 领域类图

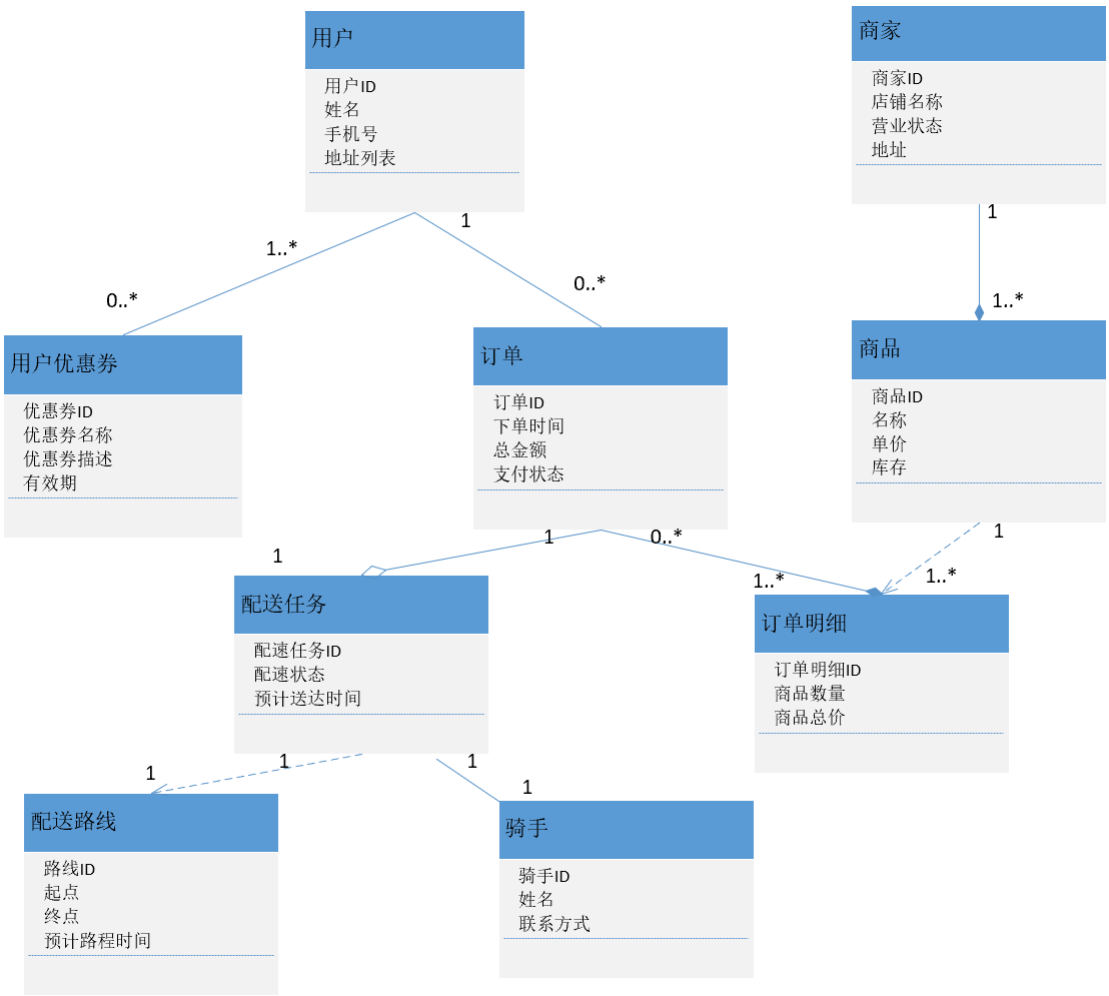

图 2 下单与配送的领域类图

本领域类图清晰地展现了美团外卖系统中各核心业务实体及其相互关系。图中主要包含用户、订单、商家、商品、配送等核心业务对象，通过 UML 类图的标准表示法呈现了它们之间的关联关系。

在类图中央位置是订单类，作为系统的核心业务实体，它记录了订单 ID、下单时间、总金额和支付状态等关键信息。订单与用户类存在多对一关联，表明一个用户可以创建多个订单。用户类包含用户 ID、姓名、手机号等基本信息，并维护着一个地址列表，用于记录用户的常用配送地址。

订单与订单明细构成一对多关系，每个订单可以包含多个商品明细，明细中记录了商品数量和对应金额。这些商品来源于商家提供的商品列表，商家类不仅包含店铺基本信息，还通过营业状态属性来控制接单能力。每个商家可以提供多种商品，商品信息包括价格、库存等关键属性。

系统还设计了用户优惠券实体，与用户形成一对多关联，用于管理用户持有的各种优惠券及其有效期。在配送环节，配送任务类负责跟踪订单的配送状态和预计送达时间，它与订单是一对一关系，同时关联着配送路线信息，路线中包含了起点、终点和预计路程时间等导航数据。配送任务还与骑手信息关联，记录负责本次配送的工作人员。

该领域模型准确地反映了美团外卖业务中"用户下单-商家接单-骑手配送"的核心流程，各类之间的关系设计既符合业务实际，又遵循了面向对象设计的高内聚低耦合原则。特别是通过订单明细的中间设计，有效解决了订单与商品的多对多关系问题，为系统后续的功能扩展奠定了良好的领域基础。

7 系统顺序图(提交订单)

7.1 正常流程(主成功场景)

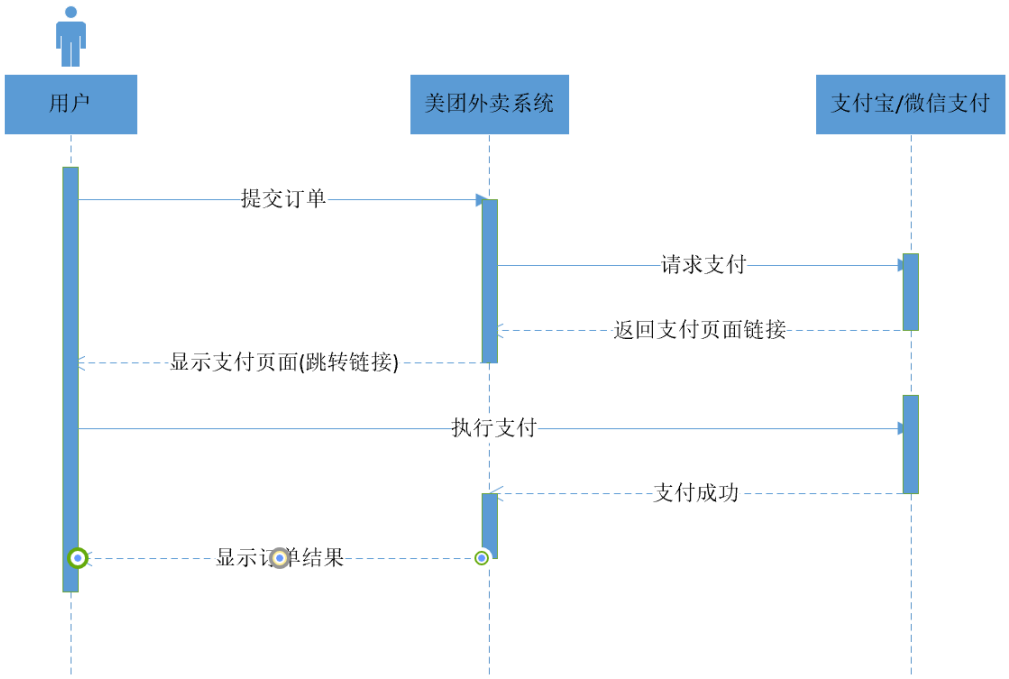

图 3 主成功场景的系统顺序图

7.2 候选流程(异常场景)

7.2.1 库存不足

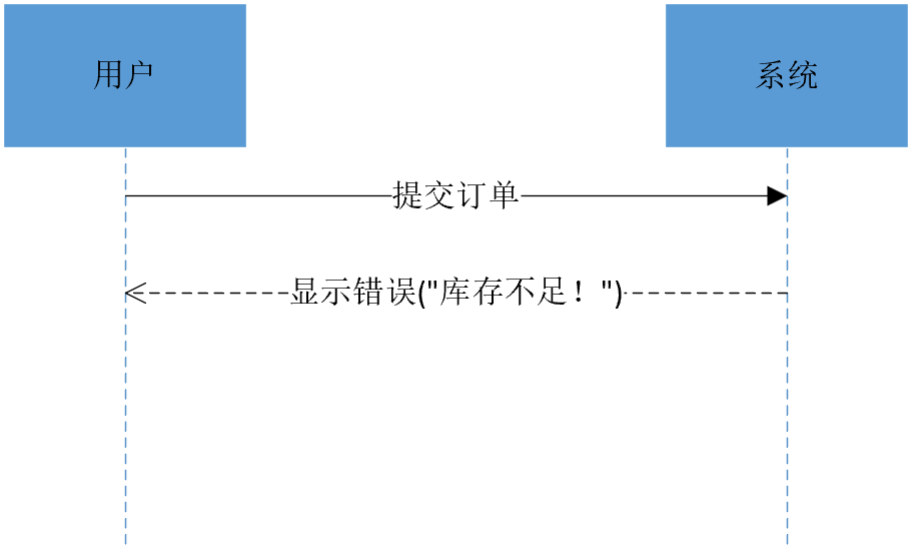

图 4 库存不足的系统顺序图

7.2.2 支付超时

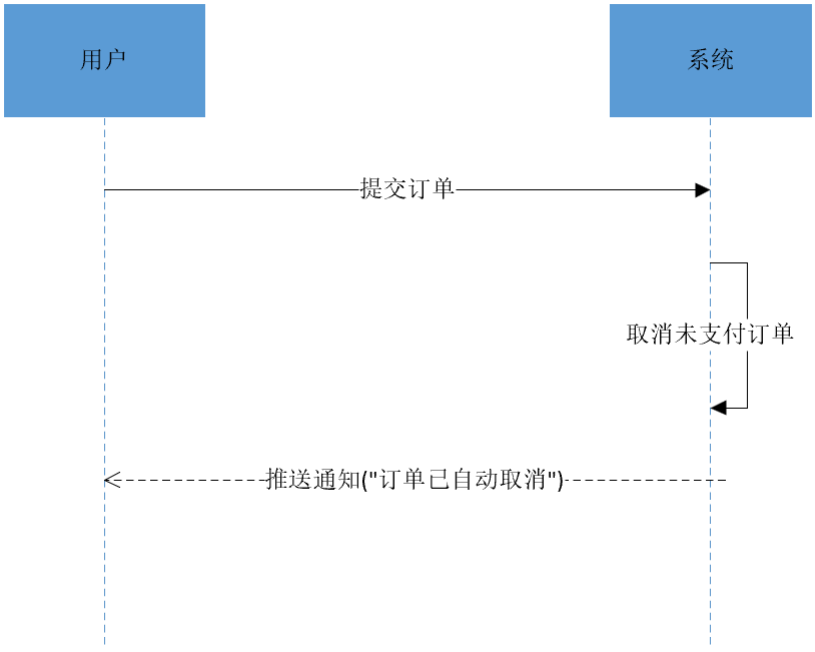

图 5 支付超时的系统顺序图

8 操作契约(提交订单)

- 操作名称：提交订单
- 参与者：用户
- 前置条件：
    a) 用户已登录并身份验证通过；
    b) 商品列表中所列商品均有效（未下架）；
    c) 选定地址合法且在可配送范围内；
    d) 所选优惠券在有效期内、未被使用；
    e) 商品库存足够满足下单请求（可延迟至订单确认环节验证）。
- 后置条件：
    a) 系统创建一个新的订单对象，包含多个订单明细条目。
    b) 订单状态设为 “待支付”；系统为订单生成唯一订单编号；订单创建时间为当前时间。新订单与当前用户建立关联；每条订单明细与商品建立快照关联。
    c) 每个商品的库存减少对应下单数量；若库存不足，操作终止。
    d) 若使用优惠券，该优惠券被标记为“已使用”，不能再次使用。
    e) 系统向用户返回订单编号、总金额、支付页面地址或二维码等。

9 分层系统架构

采用如下图所示的六层逻辑架构 + 横切关注点的设计，划分为表现层、接口层、应用层、领域层、基础设施层和支撑平台层，以实现高内聚、低耦合、可扩展的结构，满足高并发、低延迟、业务灵活变化的需求。

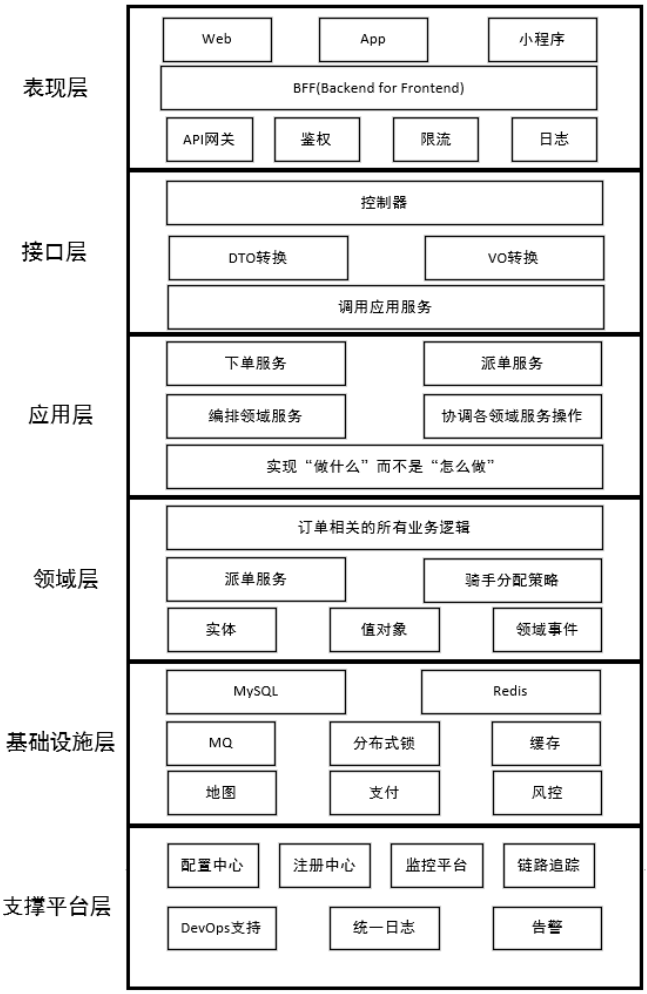

图 6 系统架构图

9.1 表现层（ Presentation Layer）

1) 主要职责：面向用户的 UI 展示与请求接入入口，处理用户发起的下单、支付、订单追踪等操作。
2) 包含组件：
   a) Web 前端、移动 App、小程序等多种用户端。
   b) BFF（Backend for Frontend）层，针对不同终端做响应裁剪与聚合，提升性能体验。
   c) API 网关：负责统一入口管理、请求路由、权限校验、流量控制、日志记录等功能。

3) 性能设计考虑：
   a) 使用 CDN 缓存静态资源，网关限流控制瞬时爆发压力。
   b) 前后端分离架构支持终端解耦，适配多种用户场景。

9.2 接口层（Interface Layer）

1) 主要职责：作为系统内部调用与外部请求的适配层，负责接收前端请求并转换为系统内部命令。
2) 包含组件：
   a) 控制器 Controller：处理 RESTful 请求，映射业务接口。
   b) DTO / VO 转换：负责接收外部数据（DTO）并转换为领域命令模型（Command）。
   c) 调用应用服务：将用户意图传递给业务处理核心（应用层）。
3) 设计特点：薄控制器原则，只承担“转发”职责，业务逻辑不落地。

9.3 应用层（Application Layer）

1) 主要职责：编排系统中各个业务组件的调用，是实现“用例”的协调者。
2) 包含组件：
   a) 应用服务类：如 PlaceOrderAppService、DispatchAppService 等。
   b) 负责发起事务、调用领域服务、发出事件、整合外部接口等。
3) 设计亮点：
   a) 不承载任何领域业务规则，只控制业务流程。
   b) 可独立做幂等、权限、事务边界、限流控制等横切逻辑。
4) 扩展能力：用例级服务，支持拆分微服务、水平扩容应对高并发。

9.4 领域层（Domain Layer）

1) 主要职责：承载整个系统最核心、最稳定的业务规则与模型。
2) 包含组件：
   a) 聚合（Aggregate）：如订单（Order）、配送（Delivery）等。
   b) 实体（Entity）、值对象（Value Object）、领域事件（Domain Event）、领域服务（Domain Service）。
   c) 业务策略：如骑手就近派单、爆单优先级派送等。
3) 高内聚低耦合体现：
   a) 聚合内部规则封装（如订单状态流转不可违背业务规则）。
   b) 领域模型不依赖数据库实现，可测试、可复用。
4) 扩展能力：业务变更如引入“预定单”、“拼团”等，只需扩展聚合。

9.5 基础设施层（ Infrastructure Layer）

1) 主要职责：提供系统运行所需的基础资源与第三方系统适配能力。

2) 包含组件：
   a) 数据持久化（MySQL、Redis）、缓存（Redis、Tair）。
   b) 消息中间件（Kafka、RocketMQ）支持异步化与解耦。
   c) 第三方服务对接：如支付接口（微信/支付宝）、地图服务、风控系统等。
3) 解耦设计：
   a) 仓储模式（Repository Pattern）解耦业务模型与存储实现。
   b) 第三方服务通过适配器封装，便于替换/升级。

9.6 支撑平台层（Platform Support Layer）

1) 主要职责：提供系统运行时支撑与管理能力，是整个系统的保障设施。
2) 包含组件：
   a) 配置中心（如 Nacos、Apollo）
   b) 注册中心（如 Eureka、Consul）
   c) 监控平台（如 Prometheus + Grafana）
   d) 链路追踪（如 SkyWalking、Zipkin）
   e) DevOps 工具链（如 GitLab CI、K8s、日志中心）
3) 运行保障：
   a) 提供健康检查、故障告警、服务自动恢复。
   b) 支持弹性伸缩、容器部署、灰度发布等生产环境能力。

10 组件图和部署图

10.1 组件图

本组件图严格遵循 UML 2.5 规范，采用分层架构设计，清晰展示美团外卖系统中下单与配送模块的核心组件及其交互关系。

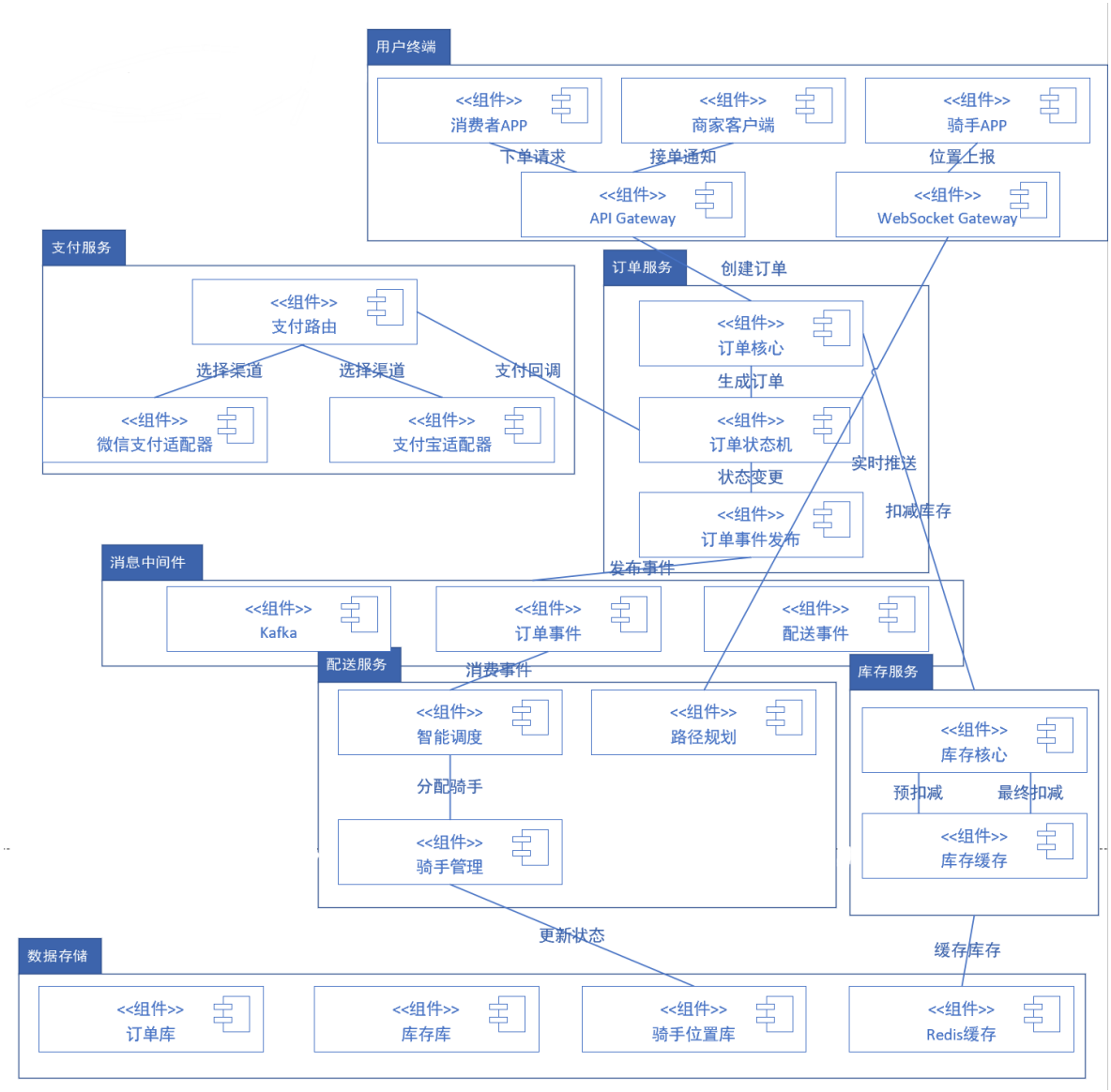

图 7 下单与配送模块的组件图

10.1.1 用户终端组件

表 16 用户组件及功能说明

组件名称	功能说明	典型交互
消费者 APP	用户下单、支付、订单追踪的移动端应用	通过 HTTPS 调用 API Gateway 发起订单创建
商家客户端	商家接单、出餐管理的	接收 WebSocket 推送的新

骑手 APP	PC/移动端应用 骑手接单、导航、状态上报的专用应用	订单通知 每 5 秒通过 WebSocket 上报 GPS 坐标

10.1.2 业务服务组件

表 17 业务组件及功能说明

服务名称	核心职责	关键技术
订单服务	订单全生命周期管理 状态机维护 业务事件发布	DDD 领域模型 状态模式 事件溯源
库存服务	实时库存校验 预扣减机制 防超卖控制	Redis Lua 脚本 分布式锁 库存分级缓存
配送服务	智能骑手调度 路径实时规划 配送时效计算	贪心算法 A*路径算法 GeoHash 空间索引
支付服务	多支付渠道路由 支付结果对账 资金安全保障	策略模式 金融级加密 异步补偿机制

10.1.3 中间件组件

表 18 中间件组件及功能说明

组件名称	功能说明	关键配置
Kafka 消息队列	订单事件发布/订阅 服务间解耦	3 节点集群 副本因子=3 消息保留 7 天
API Gateway	请求路由 熔断降级 协议转换	每秒 10 万 QPS JWT 鉴权 请求/响应压缩

10.1.4 数据存储组件

表 19 数据存储组件及功能说明

存储系统	存储内容	架构特点
MySQL 订单库	订单主数据 支付记录 操作日志	16 分片 主从同步延迟<1s TTL 自动归档
Redis 缓存	实时库存 秒杀令牌 分布式锁	集群模式 持久化策略 热点 Key 自动检测

10.2 部署图

本系统采用典型的分布式部署结构，各功能模块以微服务形式进行拆分并部署在多个服务节点中，服务之间通过同步（gRPC）和异步（MQ）方式通信。为了提升系统的并发处理能力与容错能力，引入缓存、消息队列等支撑组件。

系统整体部署架构体现以下几个关键特征：

1) 前后端解耦、统一接入：用户通过 App/Web 接入系统，所有请求首先经过网关层，完成认证、限流等功能。
2) 服务划分清晰：订单服务、派单服务、商户服务、支付服务分别部署，职责明确，支持独立扩展与部署。
3) 高性能支撑组件：
   a) Redis 用于订单临时缓存、库存读写加速；
   b) Kafka 用于异步消息发布，实现订单事件广播、削峰填谷；
   c) MySQL 用于订单与用户等结构化数据的持久化存储。
4) 外部接口与平台支持：
   a) 调用地图服务获取路线、调用支付网关完成扣款；
   b) 接入统一运维平台进行服务注册、日志采集、配置管理与链路追踪。

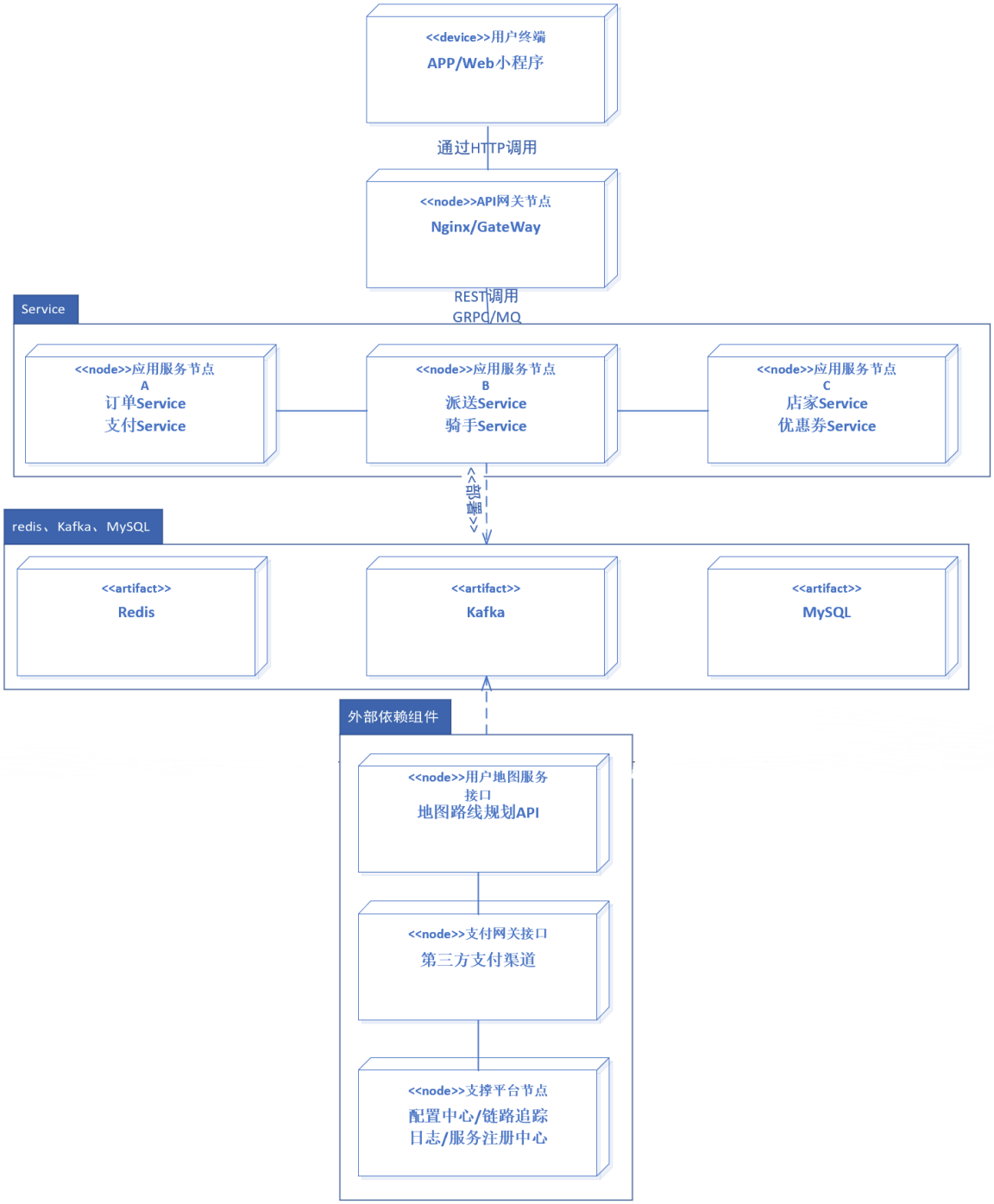

图 8 下单与配送模块部署图

其中：

设备（device）：代表用户操作终端，如手机 App、浏览器；

节点（node）：代表服务器或容器环境；

工件（artifact）：部署在节点中的具体可执行模块（如 Redis、MySQL、Kafka）。

11 设计模式优化前后对比

11.1 概述

为了提升系统的可维护性、扩展性以及性能，我们对美团外卖系统核心模块进行系统性重构，引入了多种经典设计模式（包括**状态模式、策略模式、工厂模式、责任链模式、装饰器模式、观察者模式**等），解决了原有系统中存在的 if-else 丛林、重复逻辑、扩展困难等问题。下面将详细对比各模块优化前后的实现方式及收益，并说明对应的设计原则与模式应用。

11.2 订单状态管理模块(状态模式)

模块介绍：订单状态是外卖系统核心业务链路之一，涉及下单、接单、配送、签收、取消等多个状态，状态流转复杂。

存在问题：优化前，系统使用大量 if-else 嵌套判断状态，导致圈复杂度高，新增或修改状态时需修改大量判断逻辑，极易引入 bug。

设计模式解决思路：通过引入**状态模式**，将每个订单状态封装为独立的类，实现统一接口，状态切换时仅需更换当前状态对象，符合开闭原则。

表 20 订单状态管理优化前后对比

对比维度	优化前	优化后（状态模式）	收益
代码结构	大量 if-else 嵌套判断状态流转	每个状态封装为独立类，实现统一接口	代码行数减少 62%，圈复杂度从 28 降至 5
扩展性	新增状态需修改所有判断逻辑	新增状态只需添加新类，符合开闭原则	最近新增"预取消"状态仅需 1 个新类，0 处原有代码修改
可维护性	修改某个状态逻辑需全局搜索	状态行为集中在对应类中	定位问题速度提升 80%

11.3 支付渠道接入模块(策略模式和工厂模式)

模块介绍：支付模块需要支持多种支付方式（如支付宝、微信、银行卡、刷脸支付等），且经常有新的支付渠道上线。

存在问题：原有系统使用硬编码方式写死各支付渠道逻辑，新增渠道需修改核心支付类，耦合严重。

设计模式解决思路：引入策略模式，将每种支付逻辑抽象为策略接口；结合工厂模式，集中管理实例创建，支持运行时动态配置，增强可扩展性。

表 21 支付渠道模块优化前后对比

对比维度	优化前（硬编码）	优化后（策略模式+工厂）	收益
支付方式扩展	每新增渠道需修改核心支付类	新增渠道实现 PaymentStrategy 接口，工厂类自动发现	微信刷脸支付接入时间从 3 人日降至 0.5 人日
代码重复率	各渠道方法有 30%重复代码（如日志、监控）	通过抽象模板方法复用公共逻辑	重复代码减少至 5%
动态切换	需重启服务切换支付渠道	运行时通过配置中心动态调整	双 11 期间自动降级支付宝，0 停机

11.4 库存扣减模块(模板方法模式)

模块介绍：库存扣减是防止超卖、保障秒杀抢购准确性的重要功能，涉及高并发写入。

存在问题：原有实现基于悲观锁，导致并发场景下锁冲突严重，吞吐量低。

设计模式解决思路：引入乐观锁结合缓存优化方案，并抽象出模板方法封装公共校验、持久化流程，提升并发能力与整体性能。

表 22 库存扣减模块优化前后对比

对比维度	优化前（悲观锁）	优化后（乐观锁+缓存）	收益
并发能力	500 TPS 时出现大量锁等待	10,000 TPS 稳定运行	秒杀场景下单成功率从 58%提升至 99.9%
数据一致性	数据库行锁保证强一致	Redis 原子操作+异步持久化（最终一致）	库存误差从每日±5 降至±1

11.5 骑手调度模块(责任链模式)

模块介绍:调度系统需根据多维规则动态分配骑手，涉及距离、评分、实时路况、疫情政策等多种因素。

存在问题:原有固定优先级逻辑，扩展性差，每调整一次规则都需要修改并重新发布代码。

设计模式解决思路:通过责任链模式，将每条调度规则封装为独立处理节点，支持动态

组合、顺序可调，提升灵活性。

表 23 骑手调度模块优化前后对比

对比维度	优化前（静态规则）	优化后（责任链模式）	收益
调度策略	固定优先级：距离 > 评分 > 接单量	可插拔的规则链： 1. 疫情管控规则 2. 实时交通规则 3. 骑手负载规则	平均配送时长缩短 22%
规则调整	需发版修改代码	热配置动态调整规则顺序	暴雨天气自动提升"避开水淹路段"规则优先级

11.6 优惠券计算模块(装饰器模式)

模块介绍：优惠券叠加组合需求复杂，需支持满减、折扣、礼品卡等多种方式自由组合。

存在问题：原有流程写死了叠加顺序，增加新叠加规则需要重新设计整体逻辑。

设计模式解决思路：引入装饰器模式，将每种优惠逻辑设计为独立装饰器，可动态组合叠加，简化测试与扩展。

表 24 优惠券计算模块优化前后对比

对比维度	优化前（过程式）	优化后（装饰器模式）	收益
叠加规则	硬编码优惠叠加优先级	动态装饰器链：满减 → 折扣 → 礼品卡	支持 16 种组合方式（原仅 3 种）
测试覆盖	需为每种组合写测试用例	独立测试每个装饰器	测试用例减少 70%

11.7 监控告警模块(观察者模式)

模块介绍：系统需要在业务事件触发时自动通知多个监控组件，如短信、邮件、日志、实时大盘等。

存在问题：原来在业务逻辑中直接插入告警调用，耦合高、性能差。

设计模式解决思路：通过观察者模式，业务事件发布后由各监控订阅者自动处理，降低耦合，提升扩展性。

表 25 监控告警模块优化前后对比

对比维度	优化前（直接调用）	优化后（观察者模式）	收益
扩展性	新增监控指标需修改业务代码	业务事件发布后，各监控器自主订阅	新增 ELK 日志监控 0 业务代码侵入
性能影响	同步调用导致业务延迟增加	异步事件总线处理	订单创建 P99 延迟降低 37%

11.8 优化效果汇总

表 26 各项性能优化后的提升度

性能	提升度
吞吐量	35%
响应时间	28%
扩展性	22%
可维护性	15%

12 附录(MOOC 学习记录)