软件设计描述（SDD）：一体化民宿业务管理系统

1. 引言

1.1 目的

本软件设计描述（SDD）旨在为"一体化民宿业务管理系统"提供完整、结构化的设计表示。它记录了系统的软件架构、组件设计、接口定义及设计原理，服务于以下目的：

- 为开发人员提供实现指南；
- 为测试人员提供验证依据；
- 为项目管理人员提供设计评审基础；
- 为运维人员提供部署与维护参考。

1.2 范围

本文档描述“一体化民宿业务管理系统”的软件设计，该系统为基于云原生微服务架构的 SaaS 平台，整合预订管理、前台接待、房务管理、库存控制、POS 消费、客户关系管理（CRM）及经营分析等核心业务流程。本 SDD 涵盖软件需求规范（SRS）[RD2]中定义的所有功能需求（FR-01 至 FR-12）和非功能需求（NFR-01 至 NFR-18），包括：

- 使用微服务架构的高层设计，以实现可扩展性和高可用性；
- 选定的设计视点（部署视点、包视点、构件视点、架构视点、数据流视点），针对识别的设计关切（如模块化 DC-01、性能与可扩展性 DC-02、数据一致性 DC-03、安全性 DC-04、集成与互操作性 DC-05、业务灵活性 DC-06、用户体验与可用性 DC-07）；
- 接口设计、领域模型、关键业务流程交互（如预订、入住、房态更新、外部支付集成）；
- 安全机制（RBAC、JWT）和部署策略（Docker、Kubernetes），确保符合 PIPL 等合规要求。

本文档不包括详细代码实现、单元测试用例、用户操作手册或超出部署映射的硬件规格。

1.3 系统背景

系统开发基于 SRS[RD2]中定义的民宿运营 SaaS 解决方案背景。它运行于云环境，并集成外部系支付网关（支付宝、微信支付）。设计假设云基础设施稳定，外部服务可用性高，与 SRS 第 2.6 节假设一致。

1.4 系统概述

"一体化民宿业务管理系统"是一个基于云原生架构的 SaaS 平台，为民宿提供从预订、入住、房务、库存、消费到客户关系管理和经营分析的全流程数字化解决方案。系统采用微服务架构，具备高可用性、可扩展性及合规性（符合 PIPL 等法规要求）。

1.5 参考文献

[RD1] IEEE Std 1016™-2009: IEEE Standard for Information Technology—Systems Design—Software Design Descriptions, 2009.

[RD2] 《一体化民宿业务管理系统软件需求规范（SRS）

1.6 术语与缩略语

术语	定义
SDD	软件设计描述
SRS	软件需求规范
SaaS	软件即服务
RBAC	基于角色的访问控制
JWT	JSON Web Token
PIPL	个人信息保护法

2.设计利益相关者及其关切

2.1 利益相关者

利益相关者类别	角色描述	设计关切（关联 DC 编号）
项目管理人员	项目监督者，负责整体进度、资源分配和风险管理	架构清晰度、集成与互操作性、业务灵活性（DC-01，DC-05，DC-06）
开发人员	软件实现者，负责编码、集成和维护	模块化与可维护性、技术选型、集成与互操作性（DC-01，DC-05）
测试人员	质量保证者，负责功能验证和缺陷追踪	接口可测试性、数据一致性与可靠性、用户体验与可用性（DC-03，DC-05，DC-07）
系统运维人员	系统维护者，负责部署、监控和故障恢复	部署架构、高可用性、性能与可扩展性、安全性（DC-02，DC-04）
民宿经营者	系统核心用户，负责整体运营、成本控制、员工管理和决策	系统可用性、业务扩展性、数据安全、性能与可扩展性、业务灵活性（DC-02，DC-04，DC-06，DC-07）
前台员工	日常操作者，处理预订、入住、退房和收银，常兼任房务	操作便捷性、支付安全性、数据一致性与可靠性、用户体验与可用性（DC-03，DC-04，DC-07）
房务员工	维护执行者，负责清洁、房态更新和物资申领，常在弱网环境下操作	操作流畅性、数据一致性与可靠性、用户体验与可用性（DC-03，DC-07）
宾客（间接）	系统使用者，通过官网/小程序进行预订和自助服务	操作简便性、支付安全性、用户体验与可用性（DC-04，DC-07）

2.2 设计关切

设计关切按以下分类组织：

- **架构类**：关注系统整体结构、组件组织和集成，确保可扩展、易维护和支持业务演变。

- **质量类**：关注系统非功能属性，如性能、安全和可靠性，确保系统稳定运行和数据保护。
- **用户类**：关注用户交互、操作效率和业务适应性，确保不同角色（如高频操作的前台员工、低频使用的宾客）的需求得到满足。

关切编号	分类	关切描述	关联需求	关联视点
DC-01	架构类	模块化与可维护性：确保系统组件逻辑分解清晰，支持独立修改与复用。	NFR-15	构件视点、包视点
DC-02	质量类	性能与可扩展性：确保系统在高并发下响应迅速，支持云环境弹性扩展。	NFR-01, NFR-03	部署视点
DC-03	质量类	数据一致性与可靠性：确保业务数据在流程中完整、一致，支持事务与备份。	FR-01, NFR-05	数据流视点
DC-04	质量类	安全性：确保系统符合 PIPL 等法规，实现身份认证、数据加密与访问控制。	NFR-07~NFR-10	架构视点
DC-05	架构类	集成与互操作性：确保与外部系统（支付）无缝对接。	外部接口需求	构件视点
DC-06	用户类	业务灵活性：支持价格策略、房型管理等业务变更，无需大规模重构。	FR-01, FR-07, FR-12	架构视点
DC-07	用户类	用户体验与可用性：确保界面直观、操作流畅，支持多角色与弱网环境。	NFR-11~NFR-13	数据流视点、构件视点

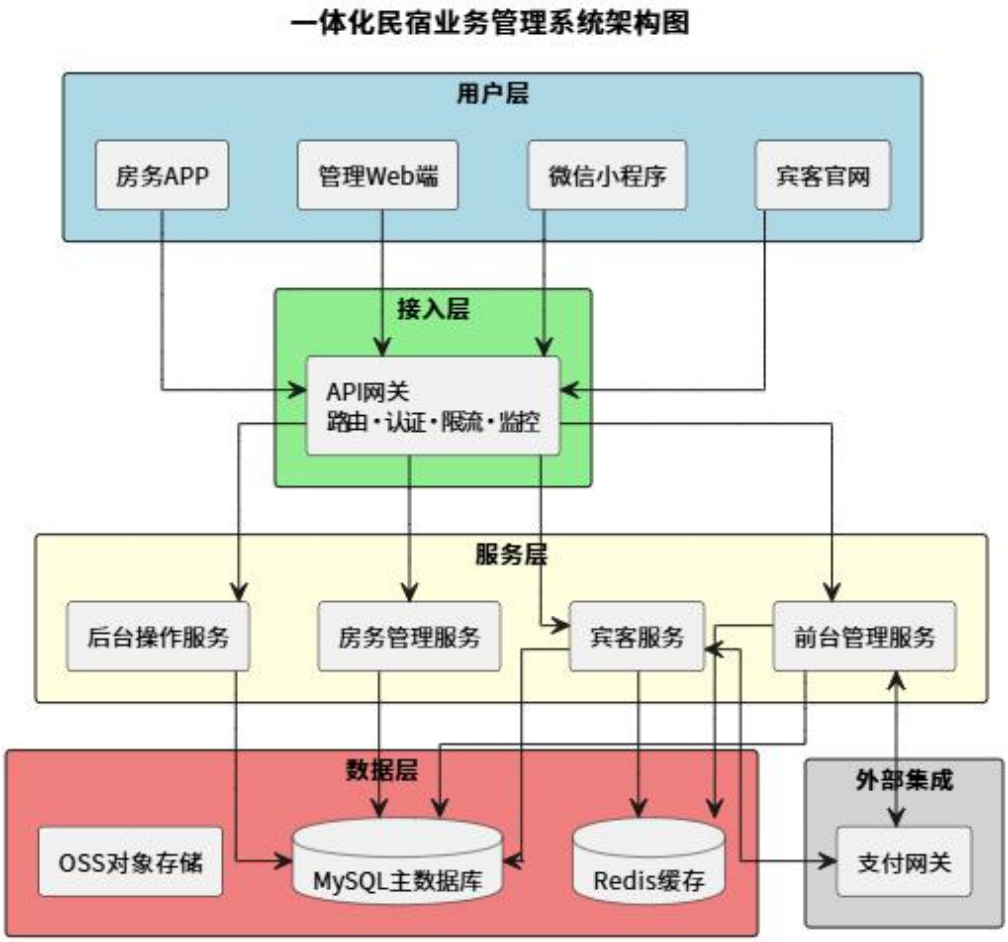
3. 设计视点与设计视图

设计视点 1：架构视点

- **视点名称**：架构视点
- **关注的设计关注点**：DC-01（模块化与可维护性）、DC-04（安全性）、DC-05（集成与互操作性）、DC-06（业务灵活性）
- **设计元素类型**：层、模块、连接（数据流、调用）。
- **建模方法**：分层分析（定义层间责任）、模块耦合评估。
- **设计语言**：架构块图、文本描述。
- **一致性规则**：层间交互必须向下（Presentation -> Business -> Data）；所有模块必须映射到微服务，无跨层直接访问。

设计视图 1：架构视图

高层：客户端层（Web/小程序）-> API 网关 -> 微服务层（预订等）-> 数据层（MySQL/Redis）。外部集成：支付网关。



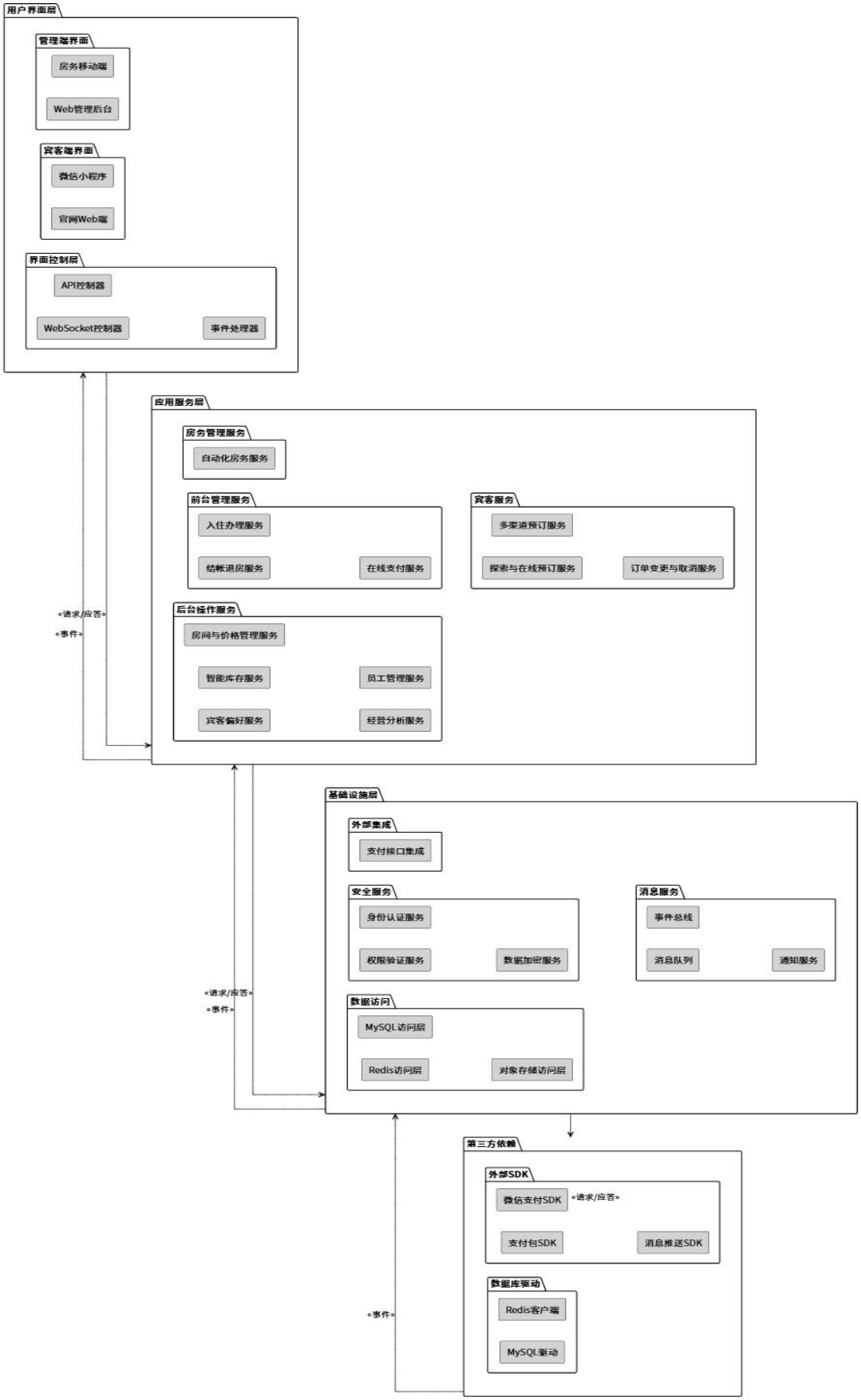
设计视点 2：包视点

- **视点名称**：包视点
- **设计关切**：DC-01（模块化与可维护性）、DC-06（业务灵活性）
- **设计元素类型**：包、子包、类/接口、依赖关系。
- **建模方法**：分组分析（将类分组到包）、依赖影响评估。
- **设计语言**：UML 包图、Java 包结构描述。
- **一致性规则**：包间依赖必须单向（无循环依赖）；所有包必须有唯一命名空间，符合 Java 命名规范。

设计视图 2：包视图

核心包：booking（预订逻辑）、inventory（库存管理）、common（共享工具）。依赖：booking <<use>> inventory。

一体化民宿业务管理平台 - 分层包图

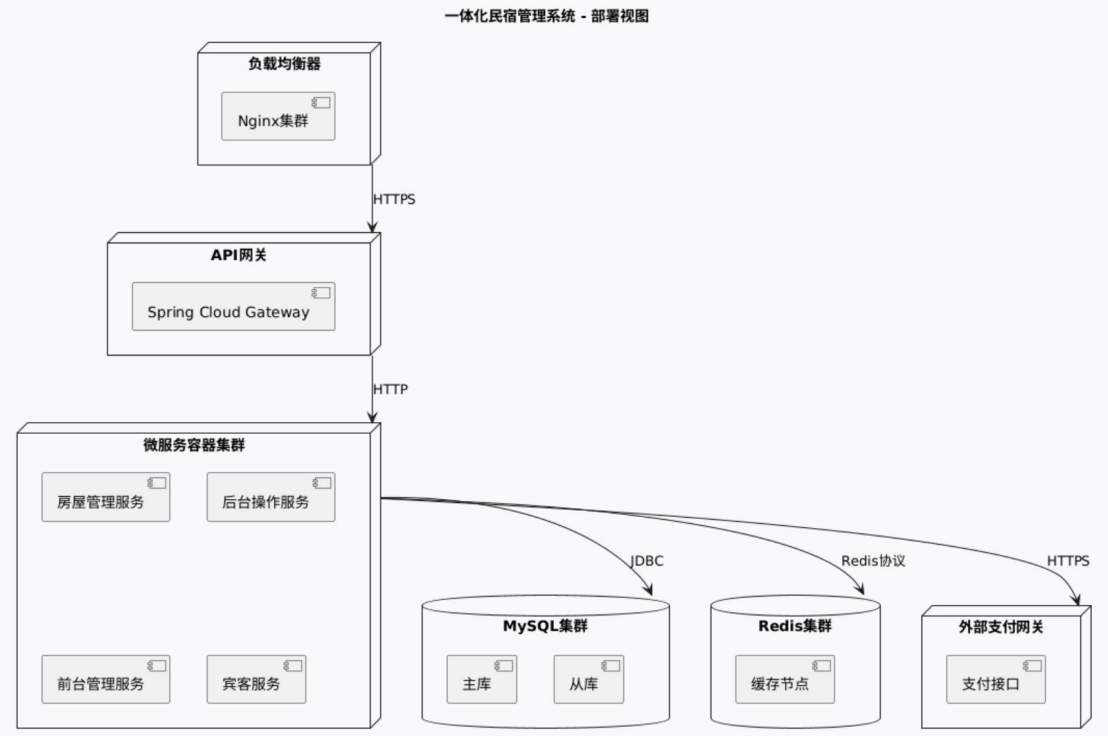


设计视点 3：部署视点

- **视点名称**：部署视点
- **设计关切**：DC-02（性能与可扩展性）、DC-04（安全性）
- **设计元素类型**：节点（服务器、容器）、工件（JAR 文件、数据库实例）、关系（网络连接、依赖）。
- **建模方法**：分配分析（将软件工件映射到物理节点）、负载均衡模拟。
- **设计语言**：UML 部署图、文本描述。
- **一致性规则**：所有节点必须指定通信协议（HTTPS）；工件部署必须符合微服务独立性，无单点故障。

设计视图 3：部署视图

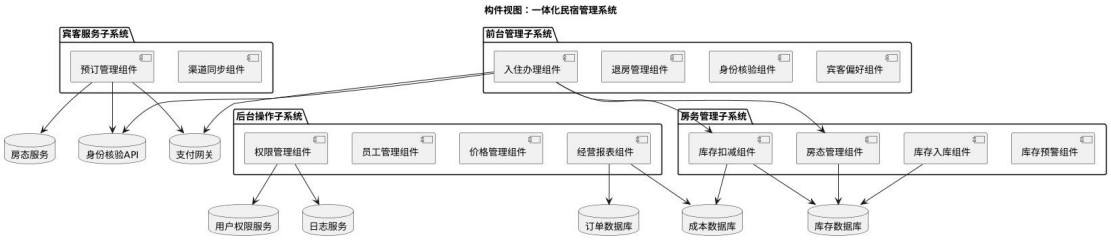
系统部署在云集群：API 网关节点连接负载均衡器，微服务（预订、库存等）部署在 Docker 容器，主从 MySQL 和 Redis 节点。外部：支付 API 通过 HTTPS 连接。



设计视点 4：构件视点

- **视点名称**：构件视点
- **设计关切**： DC-01（模块化与可维护性）、DC-05（集成与互操作性）、DC-07（用户体验与可用性）
- **设计元素类型**：构件、接口、端口、连接器。
- **建模方法**：组件分解（将系统拆分为可组装构件）、接口合同验证。
- **设计语言**：UML 构件图。
- **一致性规则**：每个构件必须定义提供/要求接口；连接器必须匹配协议（如 REST/HTTPS），无未定义端口。

设计视图 4：构件视图

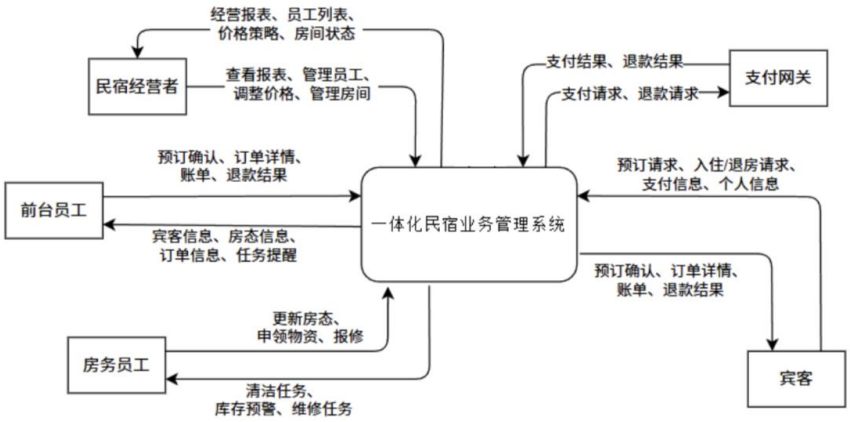


设计视点 5：数据流视点

- **视点名称**：数据流视点
- **设计关切**：DC-03（数据一致性与可靠性）、DC-07（用户体验与可用性）
- **设计元素类型**：进程、数据存储、外部实体、数据流箭头。
- **建模方法**：数据流分析（追踪数据从源到目标）、瓶颈识别。
- **设计语言**：数据流图 (DFD)。
- **一致性规则**：所有数据流必须有源和目标；进程必须定义输入/输出类型，无数据丢失或冗余。

设计视图 5：数据流视图

外部实体（宾客）-> 进程（预订请求）-> 数据存储（订单 DB）-> 进程（房态更新）。



4. 设计理由

架构决策

- 采用微服务架构：支持高可用性和扩展（SRS 2.5，NFR-01/03）。备选：单体架构。权衡：提升隔离，但增加部署复杂。关联：DC-02/06，架构/部署视点。
- 使用 Docker 和 Kubernetes 部署：确保资源优化和合规（SRS 2.4）。备选：虚拟机。权衡：轻量高效，但学习曲线陡。关联：DC-02，部署视点。
- 外部 API 集成用 HTTPS：确保数据同步（EI-XX，SRS 2.5）。备选：SOAP。权衡：安全标准化，但轻微性能开销。关联：DC-05，构件/架构视点。

模块与组件决策

- 包结构分解（如 booking/inventory）：支持无循环依赖（NFR-15）。备选：平面包。权衡：提高维护性，但初始设计稍多时间。关联：DC-01，包视点。
- 组件设计（如 Booking 要求 Payment）：确保接口复用（FR-01）。备选：松耦合。权衡：减少集成错误，但增加文档。关联：DC-01/05，构件视点。

数据与流程决策

- 数据流设计（如 宾客→预订→DB→OTA）：防超售，确保一致性（FR-01，NFR-05）。备选：批处理。权衡：实时可靠，但稍增复杂。关联：DC-03，数据流视点。
- 使用 MySQL+Redis：支持并发和加密（SRS 2.5）。备选：MongoDB。权衡：事务强，但扩展不如 NoSQL。关联：DC-03/04，数据流/架构视点。

安全与用户体验决策

- 实施 RBAC 和 JWT：确保访问控制和合规（NFR-07~10，PIPL）。备选：会话认证。权衡：减负载，但管理复杂。关联：DC-04，架构视点。
- 响应式设计（Web/小程序）：支持多设备和弱网（NFR-11~13，SRS 2.3）。备选：原生 App。权衡：跨平台快，但性能不如原生。关联：DC-07，数据流/构件视点。