AdWeb 广告管理平台 - 设计文档

1. 系统架构设计

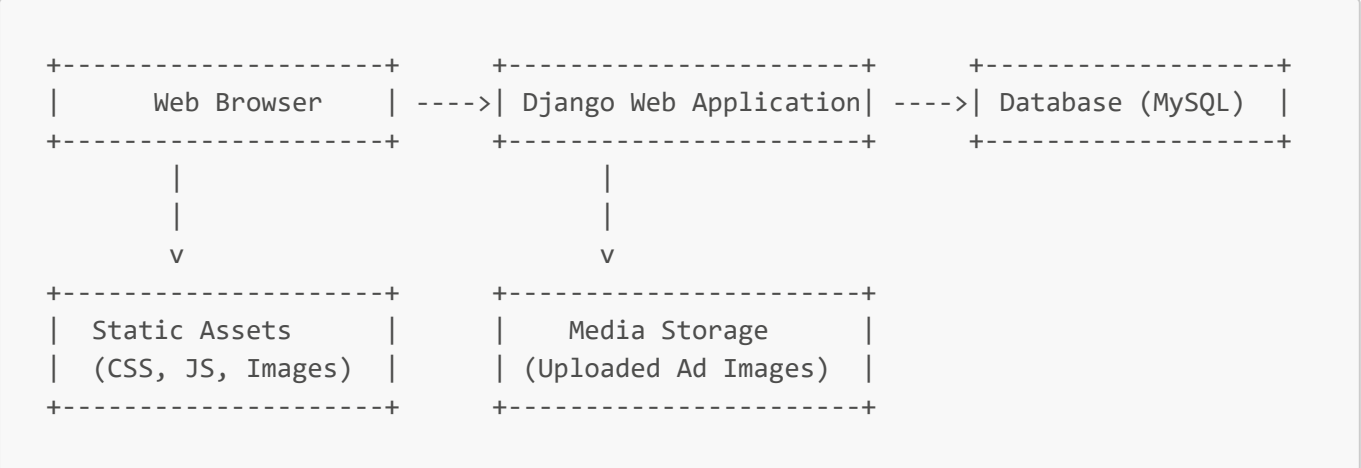
1.1 总体架构

AdWeb 广告管理平台采用 **Django MTV (Model-Template-View)** 架构模式。系统划分为多个功能模块（Django App），每个模块负责特定的业务功能。

架构层次:

- **表现层 (Presentation Layer)**
  - 使用 Django Templates 实现服务器端渲染的 Web 页面
  - 使用 HTML, CSS (Bootstrap), JavaScript 提供用户界面
  - 实现响应式设计，支持不同设备访问
- **业务逻辑层 (Business Logic Layer)**
  - Django Views 处理 HTTP 请求，执行业务逻辑
  - Service 层封装复杂业务逻辑，如 `AdService`, `PaymentService`
  - Django Forms 处理表单验证和数据处理
- **数据访问层 (Data Access Layer)**
  - Django Models 定义数据结构和关系
  - Django ORM 提供数据库操作接口
  - 数据库使用 MySQL/PostgreSQL (生产环境) 或 SQLite (开发环境)

系统组件图:



1.2 模块划分

系统按照功能领域划分为以下主要模块:

1. **Users**: 用户管理与认证
2. **AdPlace**: 广告位管理

3. **AdManage**: 广告活动和广告管理
4. **AdAudit**: 广告审核流程
5. **Payment**: 支付和财务管理
6. **DataShow**: 数据统计与展示

每个模块都被实现为独立的 Django App，具有自己的 models, views, forms 和 templates。

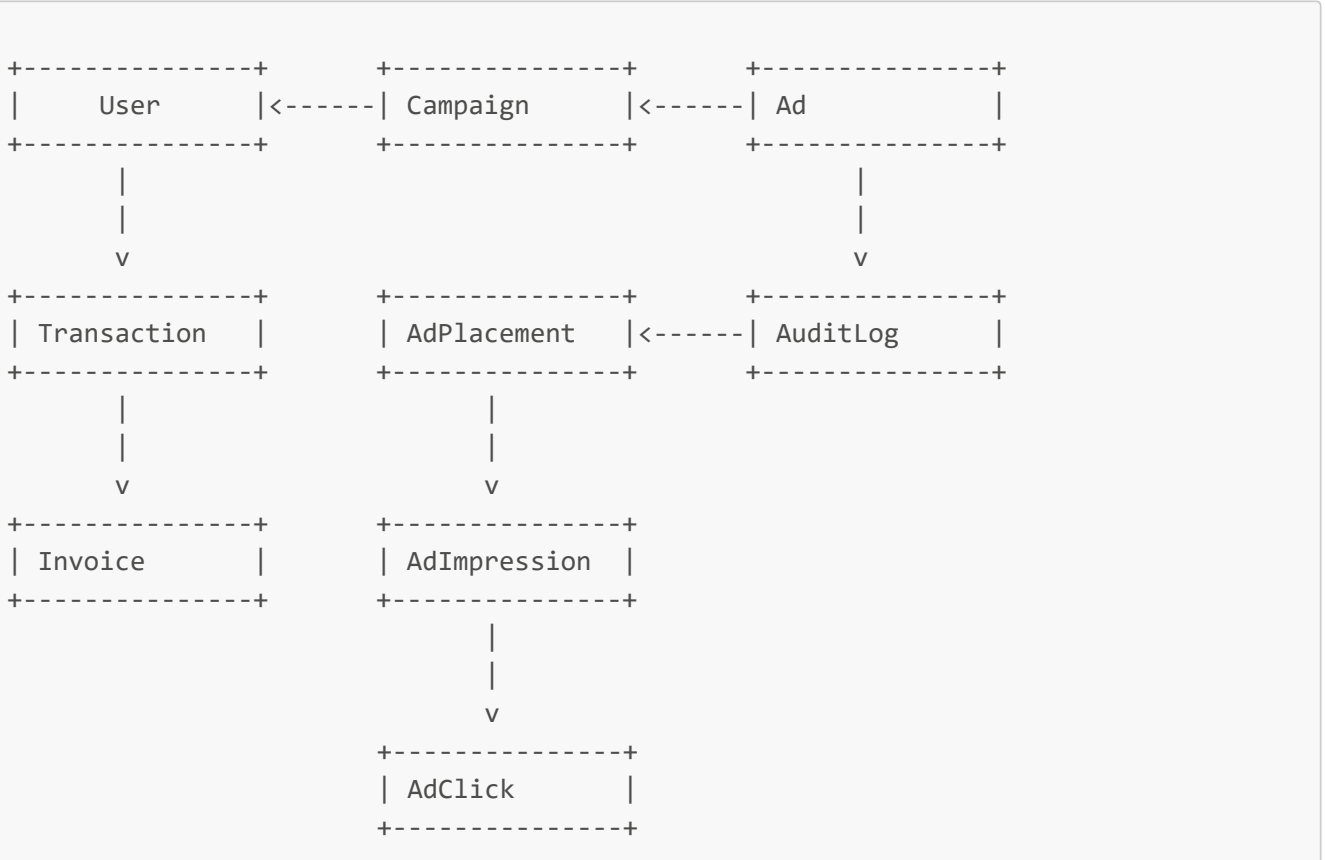
1.3 技术栈

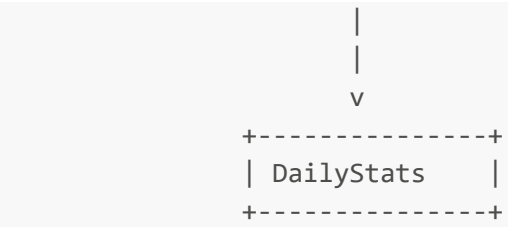
- **后端**:
  - Python 3.x
  - Django 4.x
  - Django REST framework (用于API开发)
- **前端**:
  - HTML5, CSS3 (Bootstrap)
  - JavaScript
  - Chart.js/ECharts (用于数据可视化)
- **数据库**:
  - MySQL

2. 数据库设计

2.1 ER图

系统的主要实体及其关系如下:





2.2 核心数据模型

2.2.1 用户模型 (User)

```python
class User(AbstractUser):
    USER_TYPE_CHOICES = (
        ('admin', '管理员'),
        ('advertiser', '广告主'),
    )

    AUDIT_STATUS_CHOICES = (
        ('pending', '待审核'),
        ('approved', '已批准'),
        ('rejected', '已拒绝'),
    )

    # 基本字段
    email = models.EmailField(unique=True)
    balance = models.DecimalField(max_digits=10, decimal_places=2, default=0.00)
    is_verified = models.BooleanField(default=False)

    # 审核相关字段
    audit_status = models.CharField(max_length=20, choices=AUDIT_STATUS_CHOICES,
default='pending')
    audit_time = models.DateTimeField(null=True, blank=True)

    # 用户类型
    user_type = models.CharField(max_length=20, choices=USER_TYPE_CHOICES,
default='advertiser')

    # 扩展用户资料字段
    phone = models.CharField(max_length=11, blank=True, null=True)
    company = models.CharField(max_length=100, blank=True, null=True)
    job_title = models.CharField(max_length=50, blank=True, null=True)
```

2.2.2 广告活动模型 (Campaign)

```python
class Campaign(models.Model):
    STATUS_CHOICES = [
        ('active', '进行中'),
        ('paused', '已暂停'),
```

```
        ('ended', '已结束'),
        ('deleted', '已删除')
    ]

    name = models.CharField(max_length=100, verbose_name='活动名称')
    advertiser = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='campaigns')
    budget = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='总
预算')
    start_date = models.DateTimeField(verbose_name='开始时间')
    end_date = models.DateTimeField(verbose_name='结束时间')
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='active')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

2.2.3 广告模型 (Ad)

```
class Ad(models.Model):
    STATUS_CHOICES = [
        ('pending','待审核'),
        ('active','投放中'),
        ('ended','已结束'),
        ('rejected','已拒绝'),
        ('deleted','已删除')
    ]

    name = models.CharField(max_length=100, verbose_name='广告名称')
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='pending')
    advertiser = models.ForeignKey('Users.User', on_delete=models.CASCADE,
related_name='ads')
    placement = models.ForeignKey('AdPlacement', on_delete=models.CASCADE,
related_name='ads')
    budget = models.DecimalField(max_digits=10, decimal_places=2)
    daily_limit = models.DecimalField(max_digits=10, decimal_places=2)
    image = models.ImageField(upload_to='ads/images/')
    target_url = models.URLField(verbose_name='目标链接')
    description = models.TextField(verbose_name='广告描述', blank=True)
    start_date = models.DateTimeField()
    end_date = models.DateTimeField()
    campaign = models.ForeignKey(Campaign, on_delete=models.CASCADE,
related_name='ads', null=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

2.2.4 广告位模型 (AdPlacement)

```
class AdPlacement(models.Model):
    PLACEMENT_CHOICES = [
        ('banner','横幅广告'),
        ('sidebar','侧边栏广告'),
        ('popup','弹窗广告'),
        ('text','文字广告'),
    ]
    placement_type = models.CharField(max_length=20, choices=PLACEMENT_CHOICES,
default='banner', unique=True)
    dimension = models.CharField(max_length=20, blank=True, null=True)
    price_per_day = models.DecimalField(max_digits=10, decimal_places=2)
```

2.2.5 审核日志模型 (AuditLog)

```
class AuditLog(models.Model):
    ACTION_CHOICES = (
        ('approve', '通过'),
        ('reject', '拒绝'),
    )

    ad = models.ForeignKey(Ad, on_delete=models.CASCADE, verbose_name='广告')
    admin = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name='审核
人')
    action = models.CharField(max_length=10, choices=ACTION_CHOICES,
verbose_name='审核结果')
    reason = models.TextField(verbose_name='审核意见')
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='审核时间')
```

2.2.6 交易模型 (Transaction)

```
class Transaction(models.Model):
    TRANSACTION_TYPES = [
        ('recharge', '充值'),
        ('ad_spend', '广告支出'),
        ('refund', '退款')
    ]

    STATUS_CHOICES = [
        ('pending', '处理中'),
        ('completed', '已完成'),
        ('failed', '失败')
    ]

    PAYMENT_METHODS = [
        ('balance', '余额支付'),
        ('alipay', '支付宝'),
        ('wechat', '微信支付'),
```

```
        ('bank', '银行转账')
    ]

    user = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='transactions')
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    transaction_type = models.CharField(max_length=20, choices=TRANSACTION_TYPES)
    payment_method = models.CharField(max_length=20, choices=PAYMENT_METHODS,
default='balance')
    description = models.TextField(blank=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='pending')
    ad = models.ForeignKey(Ad, on_delete=models.SET_NULL, null=True, blank=True,
related_name='transactions')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

2.2.7 数据统计模型

```
class AdImpression(models.Model):
    ad = models.ForeignKey(Ad, on_delete=models.CASCADE,
related_name='impressions')
    timestamp = models.DateTimeField(auto_now_add=True)
    ip_address = models.GenericIPAddressField(null=True, blank=True)
    user_agent = models.TextField(null=True, blank=True)
    cost = models.DecimalField(max_digits=10, decimal_places=4, default=0)

class AdClick(models.Model):
    ad = models.ForeignKey(Ad, on_delete=models.CASCADE, related_name='clicks')
    impression = models.ForeignKey(AdImpression, on_delete=models.SET_NULL,
null=True, related_name='clicks')
    timestamp = models.DateTimeField(auto_now_add=True)
    ip_address = models.GenericIPAddressField(null=True, blank=True)
    user_agent = models.TextField(null=True, blank=True)
    cost = models.DecimalField(max_digits=10, decimal_places=4, default=0)

class DailyStats(models.Model):
    ad = models.ForeignKey(Ad, on_delete=models.CASCADE,
related_name='daily_stats')
    date = models.DateField()
    impressions = models.IntegerField(default=0)
    clicks = models.IntegerField(default=0)
    cost = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

3. 接口设计

3.1 内部API接口

系统主要使用 Django 视图函数和模板系统提供Web界面，但也设计了一些内部API接口用于特定功能:

3.1.1 广告展示接口

```
GET /api/ad/{placement_id}/
```

功能: 根据广告位ID返回一个可投放的广告信息，用于前台展示广告。

参数:

- `placement_id`: 广告位ID

返回示例:

```
{
  "ad_id": 123,
  "image_url": "/media/ads/images/ad123.jpg",
  "target_url": "https://example.com/landing",
  "advertiser_id": 456
}
```

3.1.2 广告展示记录接口

```
POST /api/stats/impression/
```

功能: 记录广告展示事件

请求体:

```
{
  "ad_id": 123,
  "ip_address": "192.168.1.1",
  "user_agent": "Mozilla/5.0..."
}
```

3.1.3 广告点击记录接口

```
POST /api/stats/click/
```

功能: 记录广告点击事件

请求体:

```
{
  "ad_id": 123,
  "impression_id": 789,
  "ip_address": "192.168.1.1",
  "user_agent": "Mozilla/5.0..."
}
```

3.2 Service层接口

系统内部使用Service层封装复杂业务逻辑，主要的Service包括:

3.2.1 AdService

```
class AdService:
    @staticmethod
    def create_ad(user, data)

    @staticmethod
    def update_ad_status(ad_id, new_status)

    @staticmethod
    def get_ad_list(user=None, status=None)

    @staticmethod
    def validate_ad_time_range(start_date, end_date)
```

3.2.2 PaymentService

```
class PaymentService:
    # 计费标准
    IMPRESSION_COST = Decimal('0.01')  # 每次展示0.01元
    CLICK_COST = Decimal('0.10')       # 每次点击0.10元
    PREPAYMENT_RATIO = Decimal('0.10') # 预扣10%

    @classmethod
    def create_ad_prepayment(cls, ad: Ad) -> Transaction

    @classmethod
    def record_impression(cls, ad: Ad, ip_address: str = None, user_agent: str =
None) -> bool

    @classmethod
    def record_click(cls, ad: Ad, impression: AdImpression = None, ip_address: str
= None, user_agent: str = None) -> bool

    @classmethod
```

```
    def recharge_account(cls, user: User, amount: Decimal, payment_method: str) ->
Transaction
```

4. 组件设计

4.1 用户认证与管理组件

该组件负责用户注册、登录、权限管理等功能。

核心功能:

- 用户注册与审核
- 用户登录与认证
- 用户角色管理
- 用户资料管理

关键类/文件:

- `Users/models.py`: 定义用户模型
- `Users/views.py`: 实现注册、登录、资料管理等视图函数
- `Users/forms.py`: 定义用户表单

用户登录流程:

1. 用户访问登录页面
2. 输入用户名和密码
3. 系统验证凭据
4. 成功后重定向到对应角色的首页
5. 失败则返回错误消息

用户注册流程:

1. 用户填写注册表单
2. 系统验证表单数据
3. 创建用户记录，状态设为"待审核"
4. 管理员审核用户
5. 审核通过后用户可以登录

4.2 广告管理组件

该组件负责广告活动和广告的创建、管理等功能。

核心功能:

- 广告活动管理
- 广告创建与管理
- 广告状态更新

关键类/文件:

- `AdManage/models.py`: 定义广告活动模型

- `AdPlace/models.py`: 定义广告模型
- `AdManage/views.py`: 实现广告管理相关视图函数
- `AdPlace/services.py`: 封装广告相关业务逻辑

广告创建流程:

1. 广告主选择或创建广告活动
2. 选择广告位
3. 上传广告素材
4. 设置预算和时间范围
5. 提交审核
6. 广告状态设为"待审核"

广告状态管理:

- 待审核 -> 投放中/已拒绝
- 投放中 -> 已结束/已暂停
- 已暂停 -> 投放中/已结束

4.3 广告审核组件

该组件负责管理员对广告进行审核的功能。

核心功能:

- 待审核广告列表
- 广告审核决策
- 审核日志记录

关键类/文件:

- `AdAudit/models.py`: 定义审核日志模型
- `AdAudit/views.py`: 实现审核相关视图函数

审核流程:

1. 管理员查看待审核广告列表
2. 选择一个广告进行详细查看
3. 做出通过/拒绝决定
4. 记录审核结果和意见
5. 更新广告状态

4.4 支付与财务组件

该组件负责账户充值、费用扣除等财务功能。

核心功能:

- 账户充值
- 广告费用计算
- 交易记录管理
- 发票申请管理

关键类/文件:

- `Payment/models.py`: 定义交易、发票等模型
- `Payment/views.py`: 实现支付相关视图函数
- `Payment/services.py`: 封装支付相关业务逻辑

充值流程:

1. 用户选择充值金额和支付方式
2. 创建交易记录，状态为"处理中"
3. 模拟支付过程
4. 支付成功后更新交易状态为"已完成"
5. 增加用户账户余额

计费流程:

1. 广告展示/点击时记录事件
2. 根据计费标准计算费用
3. 从用户余额中扣除费用
4. 创建交易记录

4.5 数据统计组件

该组件负责广告展示、点击等数据的统计和展示。

核心功能:

- 记录广告展示/点击
- 统计广告效果数据
- 生成数据报表

关键类/文件:

- `DataShow/models.py`: 定义数据统计相关模型
- `DataShow/views.py`: 实现数据展示相关视图函数

数据收集流程:

1. 广告展示/点击时通过API记录事件
2. 定时任务汇总每日数据
3. 更新DailyStats记录

数据展示流程:

1. 用户选择时间范围和筛选条件
2. 系统查询对应数据
3. 生成数据图表和表格

5. 用户界面设计

系统使用Bootstrap框架实现响应式设计，主要界面包括:

5.1 通用界面

- **登录页**: 用户登录表单
- **注册页**: 用户注册表单
- **导航栏**: 根据用户角色显示不同菜单项

5.2 广告主界面

- **仪表盘**: 显示账户余额、活动广告数量、近期消费等
- **广告活动管理**: 列表显示所有活动，支持创建、编辑等操作
- **广告管理**: 列表显示所有广告，支持过滤、排序等
- **广告创建**: 分步表单，包括选择活动、广告位、上传素材等
- **充值页面**: 选择充值金额和支付方式
- **交易记录**: 表格显示所有交易历史
- **数据报表**: 图表和表格展示广告效果数据

5.3 管理员界面

- **用户管理**: 列表显示所有用户，支持审核新用户
- **广告审核**: 列表显示待审核广告，支持审核操作
- **广告位管理**: 创建和管理广告位
- **系统报表**: 展示系统整体数据

6. 安全设计

6.1 认证与授权

- 使用Django内置的认证系统
- 基于角色的访问控制 (RBAC)
- 所有敏感操作需要登录认证
- 管理员功能只对管理员角色开放

6.2 数据安全

- 密码使用哈希算法存储
- 敏感数据传输使用HTTPS
- 表单添加CSRF保护
- 输入验证防止注入攻击

6.3 业务安全

- 广告创建需要审核流程
- 交易操作具有幂等性
- 资金操作有日志记录
- 关键业务操作使用事务保证数据一致性

7. 部署架构

7.1 开发环境

- 本地开发服务器
- SQLite数据库
- Django Debug Toolbar

7.2 生产环境

- Nginx + Gunicorn/uWSGI
- MySQL/PostgreSQL数据库
- 静态资源分离存储
- 日志收集与监控

7.3 部署流程

1. 代码审查
2. 单元测试
3. 构建部署包
4. 数据库迁移
5. 静态资源收集
6. 服务重启
7. 健康检查

8. 扩展性考虑

系统设计考虑了未来的扩展需求:

- **前后端分离**: 可以逐步迁移到API + 前端框架架构
- **微服务化**: 关键模块可以拆分为独立服务
- **扩展计费模型**: 支持更多计费方式，如CPM、CPC、CPA等
- **广告定向**: 增加地域、人群等定向能力
- **API开放**: 为第三方系统提供API接口