浅谈 Web 应用架构设计

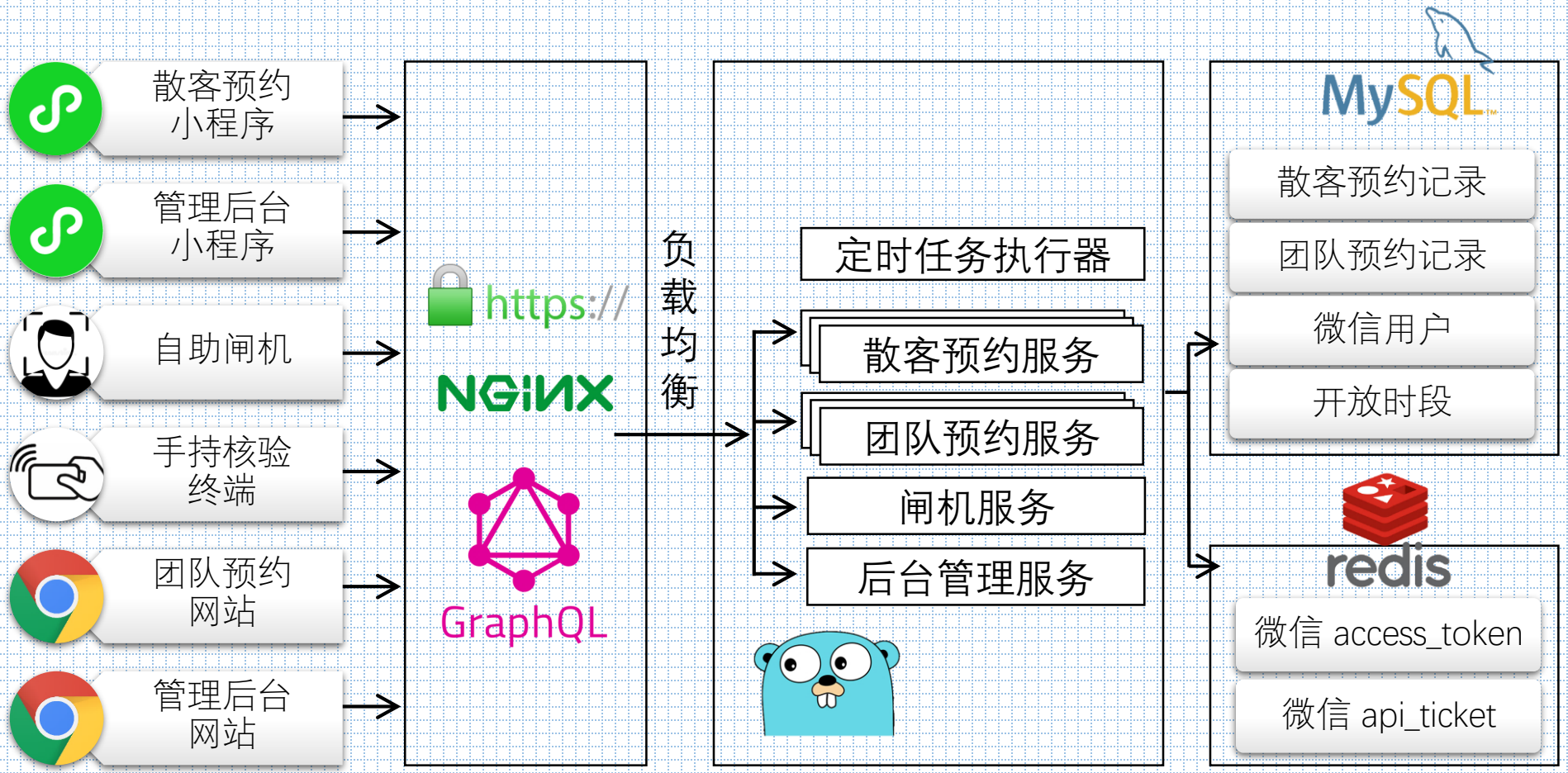
——以“参观清华”为案例

李肇阳 zhaoyang-li@outlook.com

软件工程 2018 秋

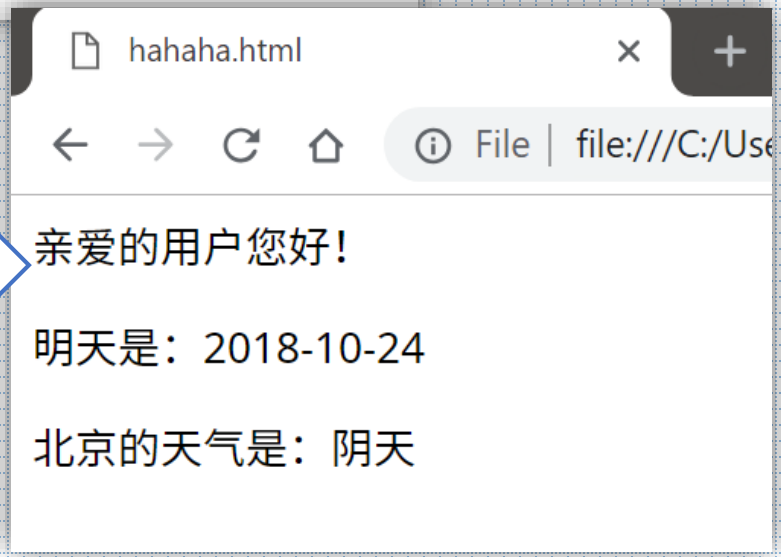
系统总体架构

客户端 网络接口 业务逻辑层 数据存储



```python
import datetime
def weather_forecast():
    tomorrow = (datetime.datetime.now() + datetime.timedelta(days=1)).date()
    return '''
        <html>
            <head></head>
            <body>
                <p>亲爱的用户您好！</p>
                <p>明天是：{date_description}</p>
                <p>北京的天气是：{weather_description}</p>
            </body>
        </html>
    '''.format_map({
        'date_description': tomorrow.strftime('%Y-%m-%d'),
        'weather_description': MAGIC_METHOD('北京', tomorrow)
    })
```

```html
<html>
    <head></head>
    <body>
        <p>亲爱的用户您好！</p>
        <p>明天是：2018-10-24</p>
        <p>北京的天气是：晴天</p>
    </body>
</html>
```



• 服务端渲染：SSR = Server-side Rendering

polls/templates/polls/index.html

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

polls/views.py

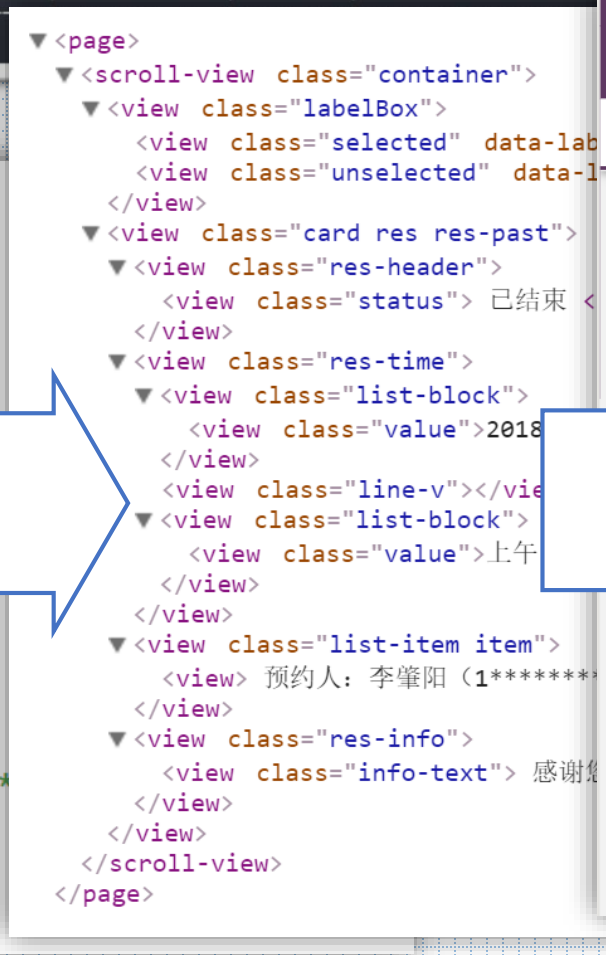
```
from django.shortcuts import render

from .models import Question


def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)
```

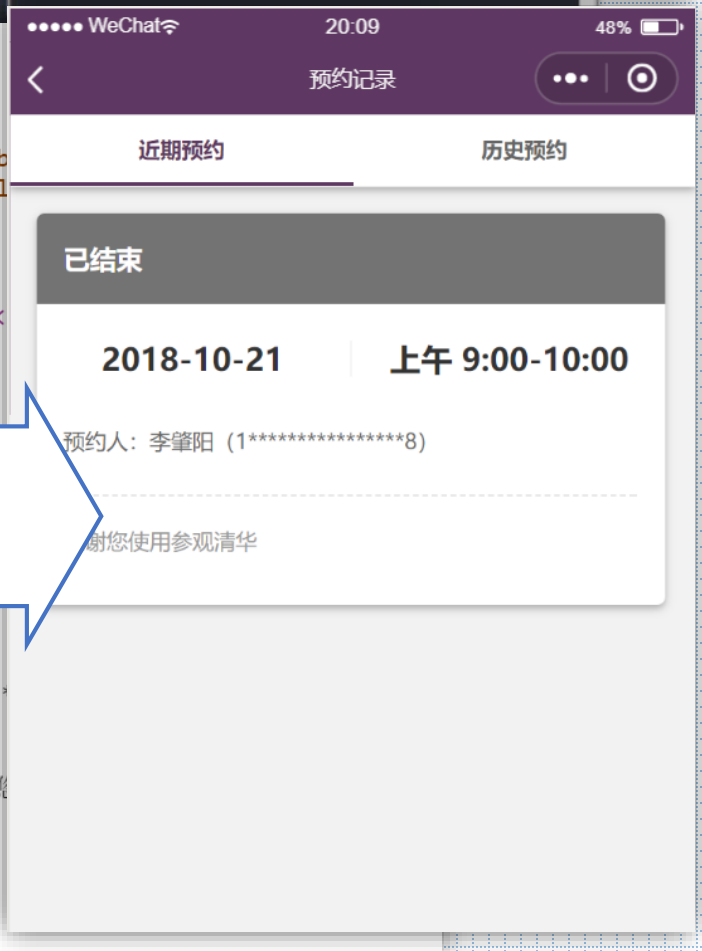
The Django template language https://docs.djangoproject.com/en/2.1/intro/tutorial03/

前后端分离

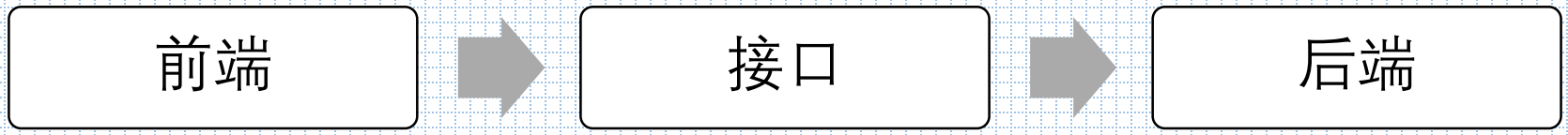

ySQL [yars]> select id,wechat_open_id,session_key,legal_name,citizen_id_hashed,created from wechat_user where id=1;
id | wechat_open_id | session_key | legal_name
1 | ohqu | sdEn | 李肇阳
row in set (0.00 sec)
ySQL [yars]> select id,time_slot_id,wechat_user_id,fine_period,time_of
id | time_slot_id | wechat_user_id | fine_period | time_of_entry | canceled | created
123589 | 170 | 1 | 9:00-10:00 | NULL | 0 | 2018-10-15 10:50:08
row in set (0.00 sec)


"reservations": [{
"reservationId": 123589,
"finePeriod": "9:00-10:00",
"timeSlot": {
"date": "2018-10-21",
"period": "上午",
"campusClosed": false,
"campusClosedFor": null
},
"visitors": [{
"entered": false,
"legalIdentity": {
"name": "李肇阳",
"citizenIdMasked": "1*
}
}]
}]

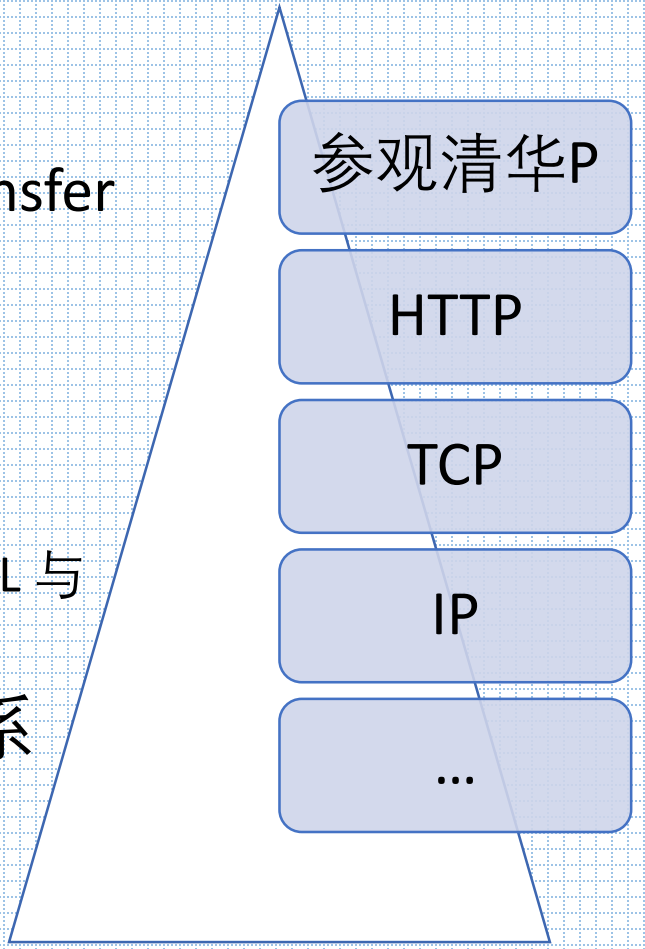

<page>
<scroll-view class="container">
<view class="labelBox">
<view class="selected" data-lab
<view class="unselected" data-l
</view>
<view class="card res res-past">
<view class="res-header">
<view class="status"> 已结束 <
</view>
<view class="res-time">
<view class="list-block">
<view class="value">2018
</view>
<view class="line-v"></vie
<view class="list-block">
<view class="value">上午
</view>
</view>
<view class="list-item item">
<view> 预约人：李肇阳（1*******
</view>
<view class="res-info">
<view class="info-text"> 感谢您
</view>
</view>
</scroll-view>
</page>


WeChat
20:09
48%
预约记录
近期预约
历史预约
已结束
2018-10-21
上午 9:00-10:00
预约人：李肇阳（1**********8）
谢您使用参观清华

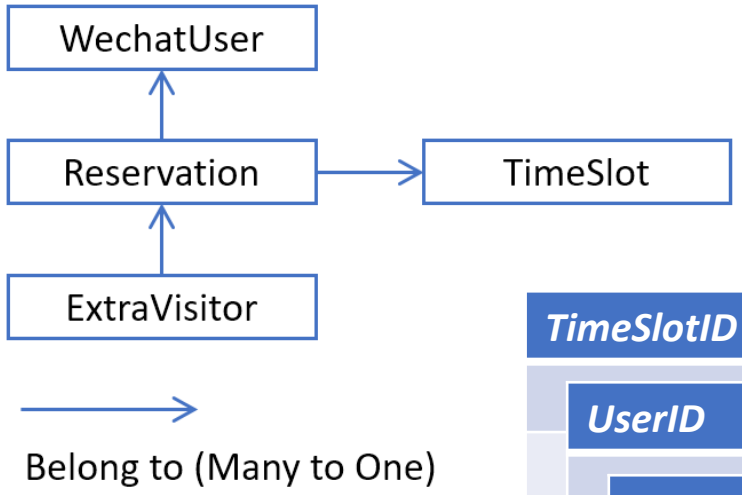
接口设计规范



- REST v.s. GraphQL
  - REST = Representational State Transfer
    - Roy Fielding，2000 年
  - GraphQL
    - Facebook，2015 年
  - 各有利弊与适用场景
    - 比如 GitHub 目前同时提供 GraphQL 与 REST 两套 API
- 与 JSON / XML / protobuf 的关系



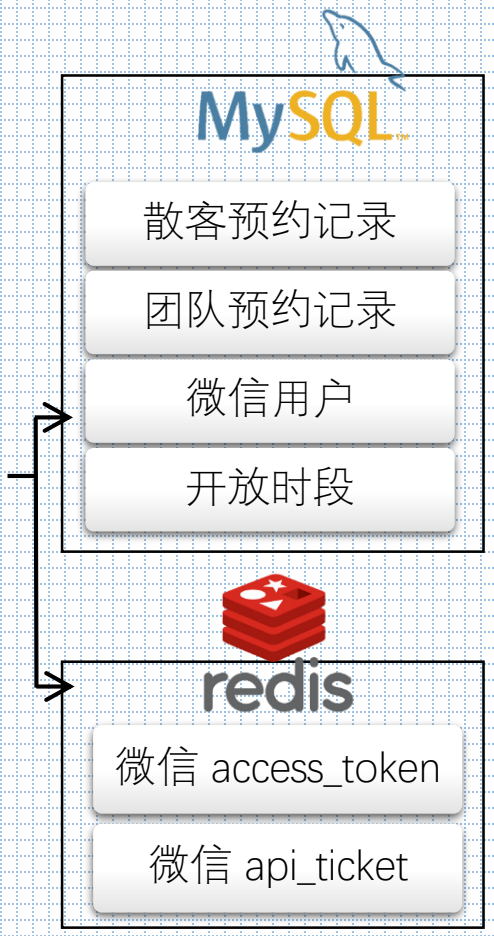
数据库



TimeSlotID	日期	名额数

UserID	证件类型	证件号

ReservationID	TimeSlotID	UserID

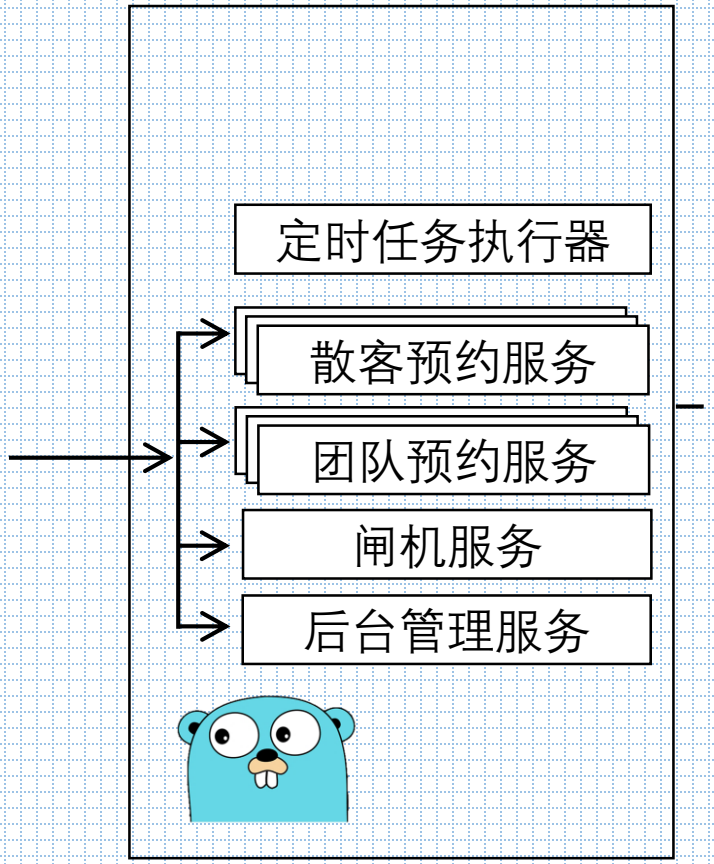


- MySQL
  - 属于关系型数据库
  - 适合结构化数据，支持 join 操作
- Redis
  - 属于非关系型数据库
  - 适合键值数据
  - 利用内存进行高速缓存

Key	Value
Access Token	…
API Ticket	…

“微服务”
Microservices

- 各组件可独立地替换、升级
- 易于快速更新

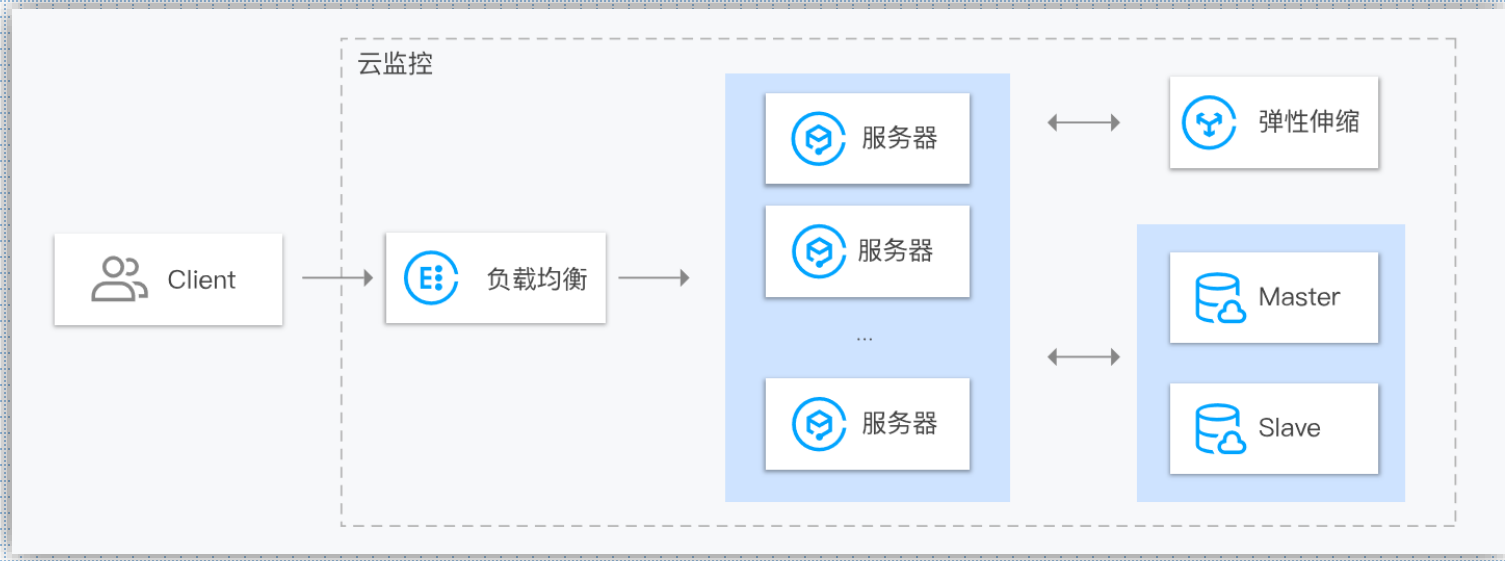


负载均衡
Load Balance

- Nginx 目前支持三种模式的负载均衡：round-robin、least-connected、ip-hash，还可指明各个上游的权重
- 更可配合弹性伸缩（Auto Scaling）

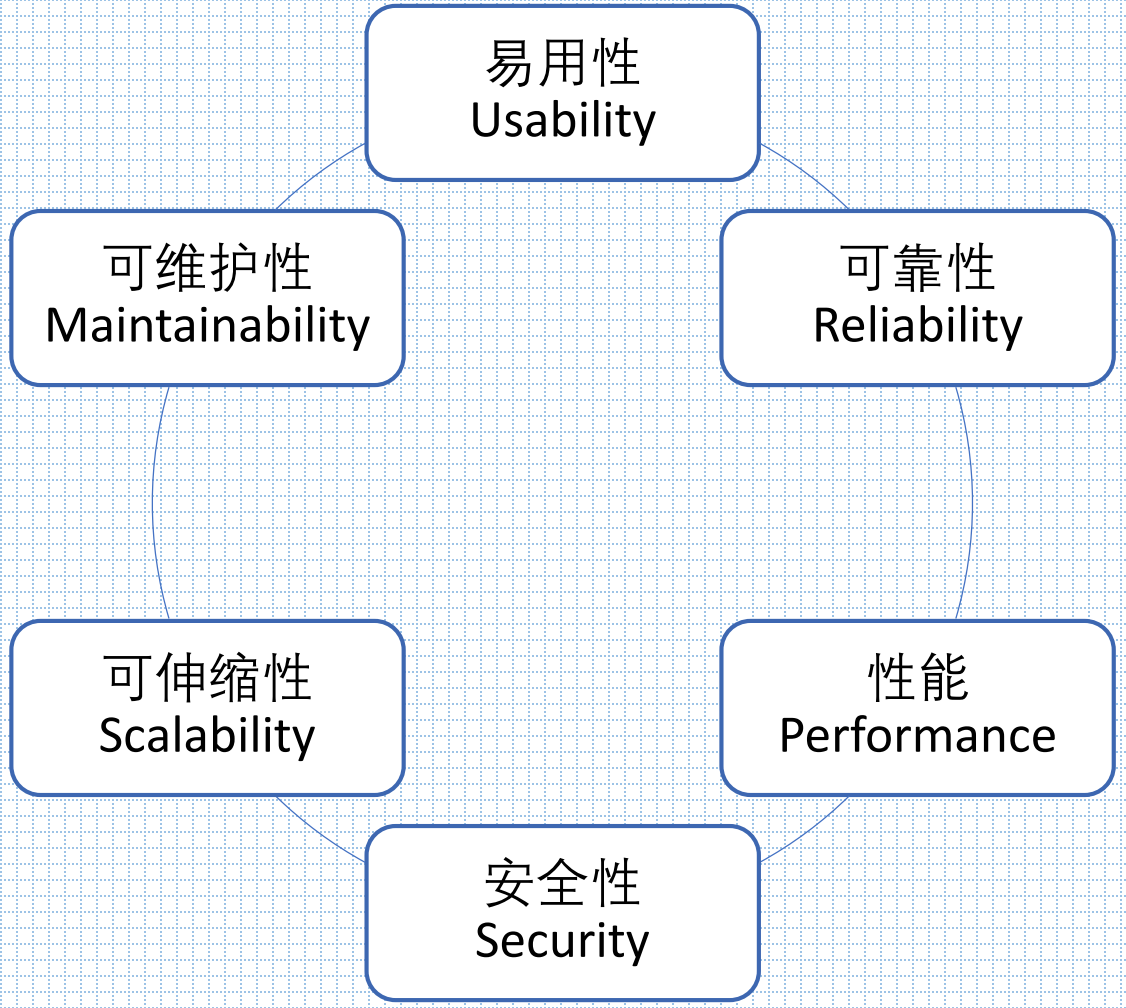
```
upstream upstream-miniapp {
    ip_hash;
    server 10.0.0.2:2333;
    server 10.0.0.10:2333;
    server 10.0.0.13:2333;
}

server {
    listen 443 ssl default_server;
    # ssl_certificate
    # ...
    location ~ ^/(login|api) {
        proxy_pass http://upstream-miniapp;
    }
}
```

→ 散客预约服务



非功能性需求

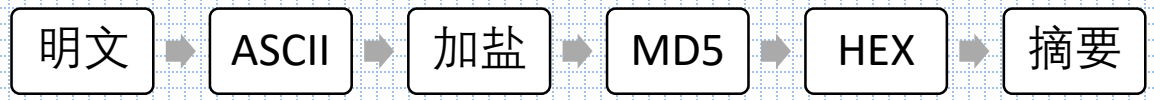
“差不多能用”
与“真的能用”
之间存在巨大的 Gap!



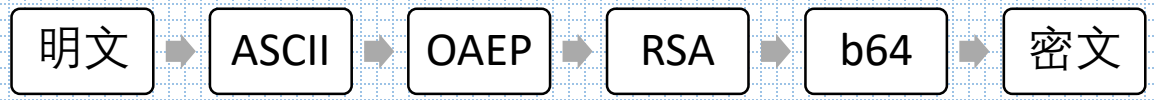
非功能性需求

- 性能优化
  - 利用 Go 语言基于协程（Goroutine）的并发支持
  - 数据库表优化：索引、冗余字段、去除外键约束
  - 设置定时任务，在低谷时段执行复杂的数据库查询
- 安全性保障
  - 身份证号加密存储

用于 1-v-N 查验：*不可逆的加盐哈希*



用于恢复明文：*可逆的非对称加密*



- 可靠性保障
  - 业务层按可伸缩设计，部署负载均衡、多机热备
  - 数据库采用主从架构，保持高可用性
  - 对闸机，设计两套接口：实时查询、定期同步，互为备份

小结