TokTik Team 青训营后端结业项目答辩汇报文档

一、项目介绍

❤️ 集成 ChatGPT 的短视频微服务应用，使用 Kitex 和 Hertz 构建

GitHub 地址，欢迎 star ：

✨🌟🌟🌟✨https://github.com/Toktik-Team/toktik ✨🌟🌟🌟✨

项目服务地址，你可以直接填写在你的抖声 APP 中：

https://toktik.xctra.cn/

二、项目分工

三、项目实现

3.1 技术选型

> 以资源利用率为核心指标，选择能实现快速横向扩展，拥有弹性伸缩能力的云原生架构

技术选型

- RPC 框架：CloudWeGo / Kitex
- HTTP 框架：CloudWeGo / Hertz
- IDL 语言：Proto3
- 数据库：PostgreSQL, Redis
- ORM 框架：Gorm, Gorm Gen
- 服务发现：Consul
- 容器编排 / 管理：Kubersphere (Kubernetes)
- CI / CD: GitHub Actions & GitLab CE & 梦想珈 / ryze
- 可观测性：OpenTelemetry, Grafana, VictoriaMetrics
- 其他第三方库 / 中间件
  - 测试：stretchr/testify, DATA-DOG / go-sqlmock
  - S3 SDK：aws-sdk-go-v2
  - 视频 / 图像处理：thumbnailer, FFmpeg, veImageX

- Web API JSON 序列化器：Protojson
- ID 生成器：gofrs / uuid, segmentio / ksuid
- 日志：sirupsen / logrus
- AIGC：OpenAI API ChatBot, OpenAI API 签名生成, Midjourney 头像生成

3.2 架构设计

> 可以补充场景分析环节，明确要解决的问题和前提假设，比如预计0.5%的用户属于大V，粉丝很多，也会经常上传视频，当前架构的解决方案是xxx。

架构目标

在设计业务应用架构时，考虑以下目标：

代码质量（Code quality）

使用指标（如测试覆盖率）来保障可维护性（Maintainability）

- 可改造性（Modifiability）
- 可管理性（Managability）
- 可适应性（Adaptability）
- ……

容灾（Resiliency）

在系统内部或外界服务发生错误时，或服务进行交付部署时，保证线上服务仍然可用

- 稳定性（Stability）
- 可用性（Availibality）
- ……

可观测性（Observability）

从日志、指标、追踪三个角度保证服务监控

- 可调试性（Debuggability）
- 可审查性（Auditability）
- ……

性能（Performance）

应用是否 promote 到 production 级别的重要考量

- 可扩展性（Scalability）
- 稳定性（Stability）
- ……

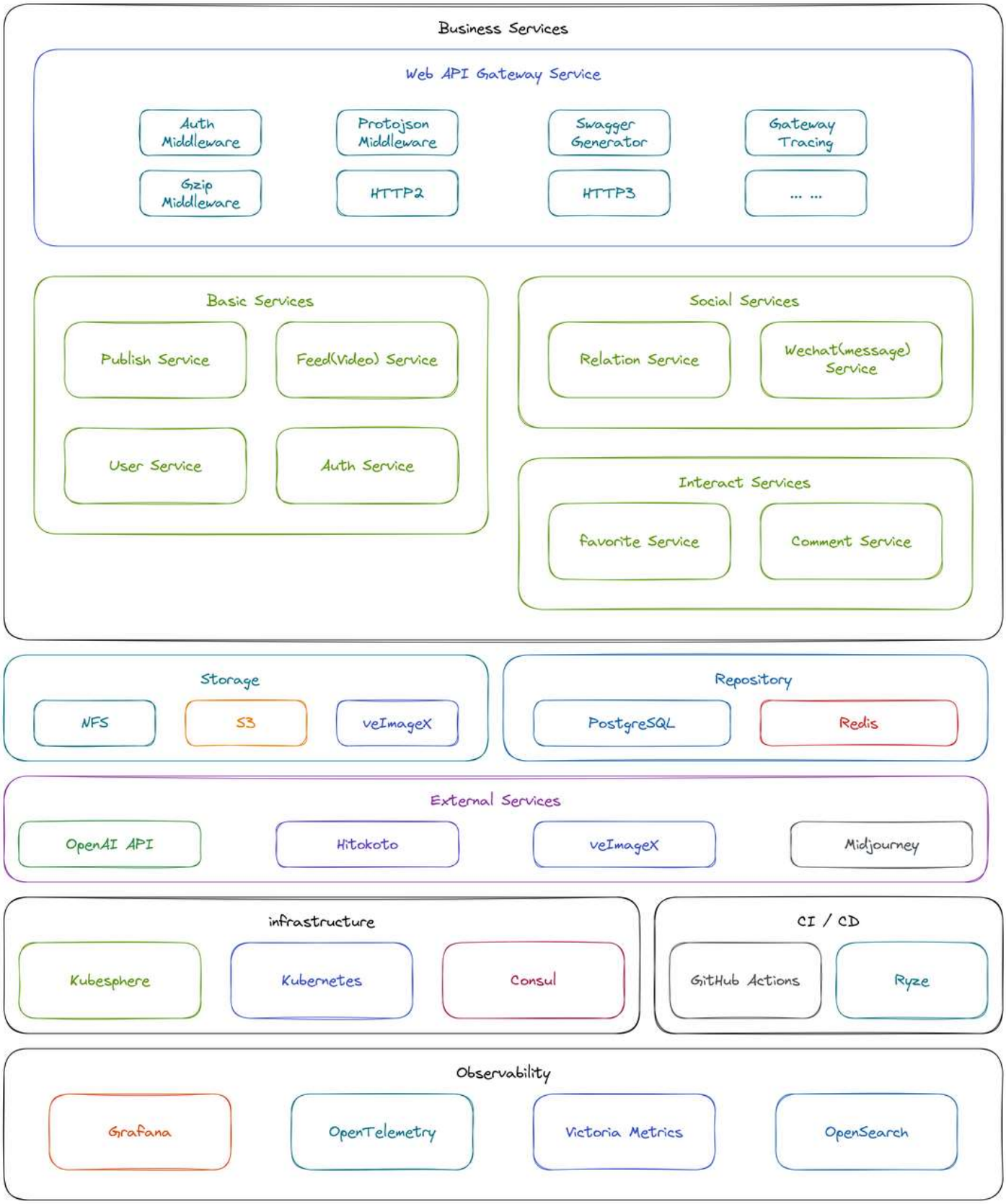
规范化（Compliance）

业务上符合领域专家制定的标准，技术上符合团队制定的标准，并且在正式 promote 前进行审查

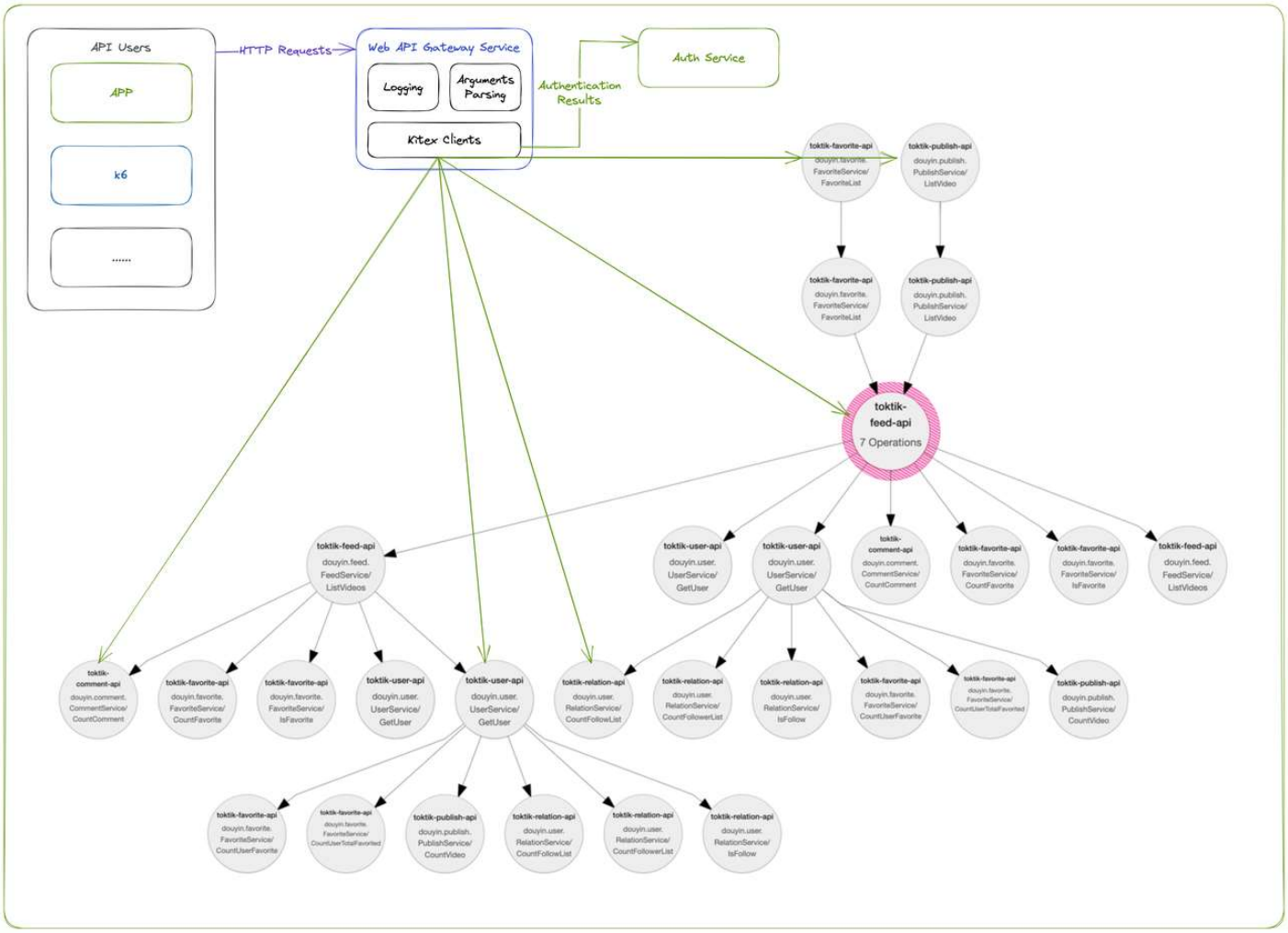
安全（Security）

系统安全是一个重要的目标，但对于开发过程来说难以落实，所以通过安全性代码分型、静态应用程序安全性测试、依赖漏洞扫描的方式来保证安全性

技术架构图


Business Services
Web API Gateway Service
Auth Middleware
Protojson Middleware
Swagger Generator
Gateway Tracing
Gzip Middleware
HTTP2
HTTP3
... ...
Basic Services
Publish Service
Feed(Video) Service
User Service
Auth Service
Social Services
Relation Service
Wechat(message) Service
Interact Services
favorite Service
Comment Service
Storage
NFS
S3
veImageX
Repository
PostgreSQL
Redis
External Services
OpenAI API
Hitokoto
veImageX
Midjourney
infrastructure
Kubesphere
Kubernetes
Consul
CI / CD
GitHub Actions
Ryze
Observability
Grafana
OpenTelemetry
Victoria Metrics
OpenSearch

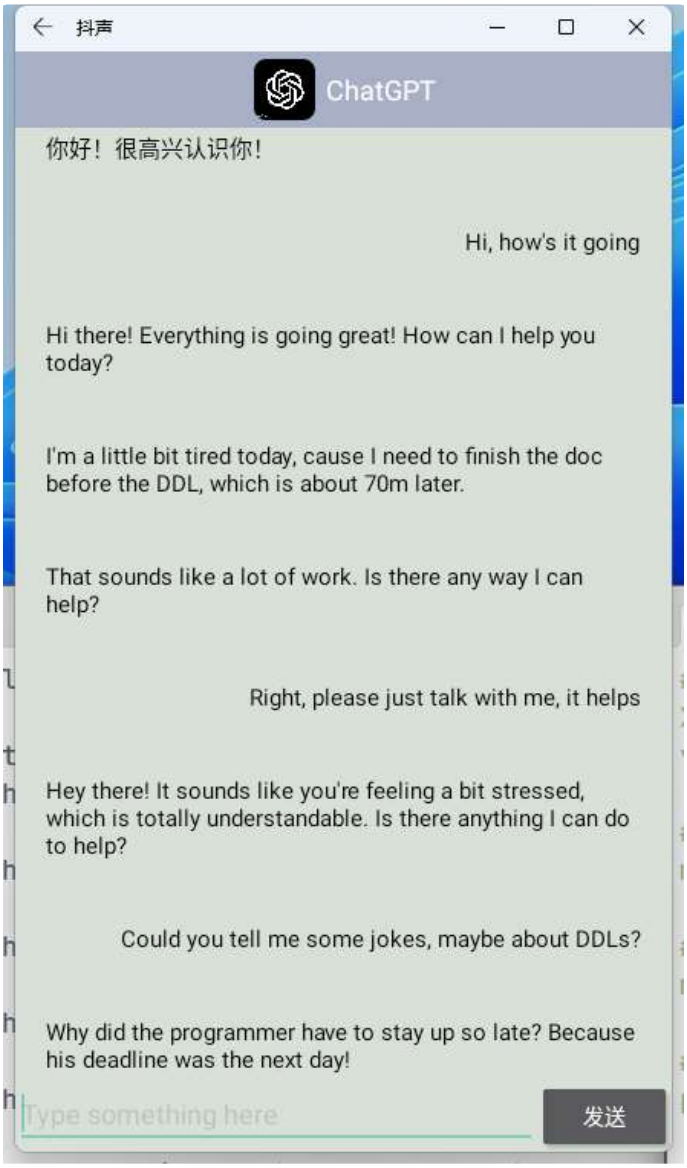
调用关系图

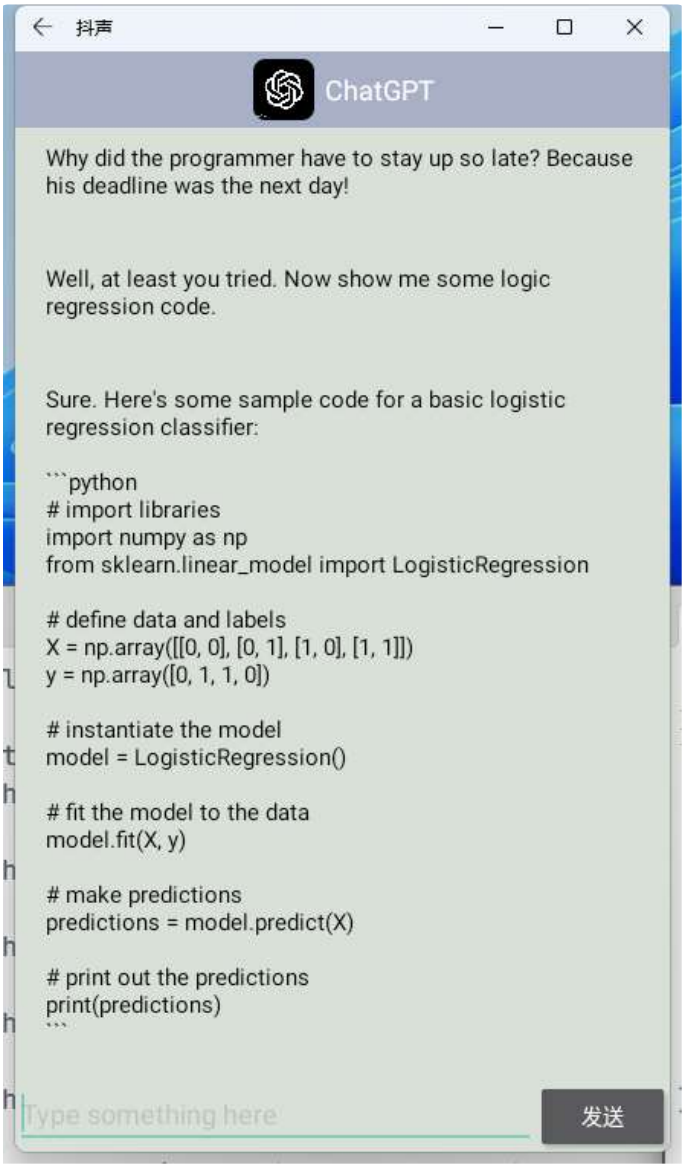


> 上述两张图详见 Excalidraw 只读页面

3.3 项目亮点功能展示

ChatGPT 集成





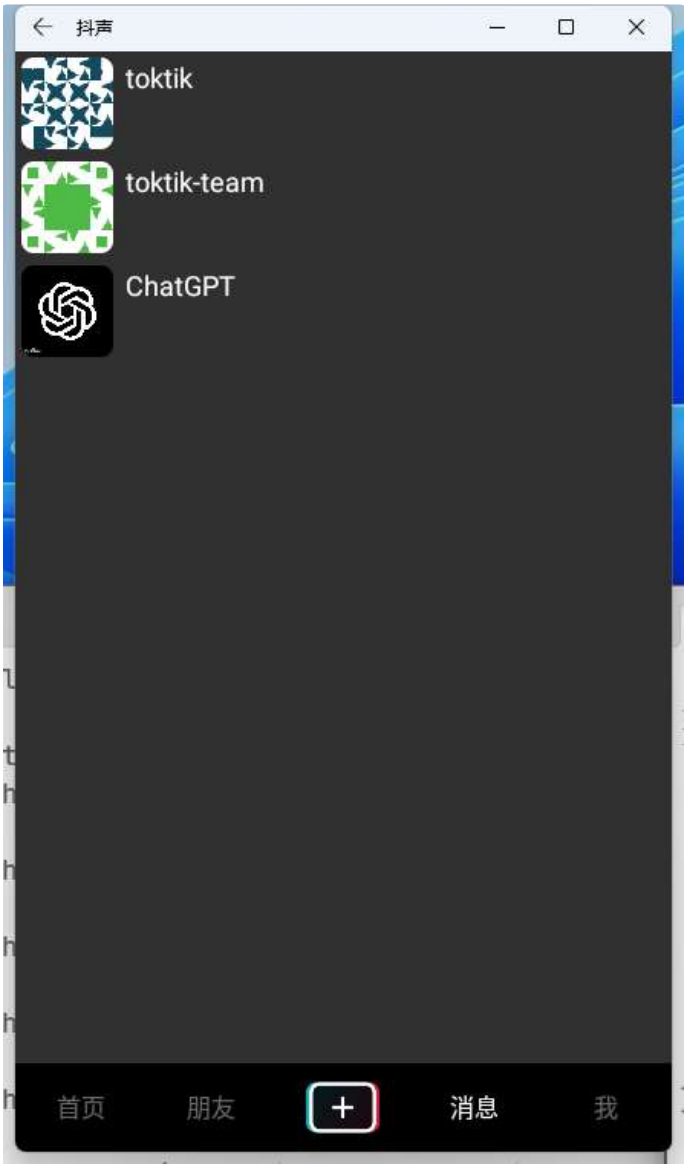
Unsplash API 动态生成头像

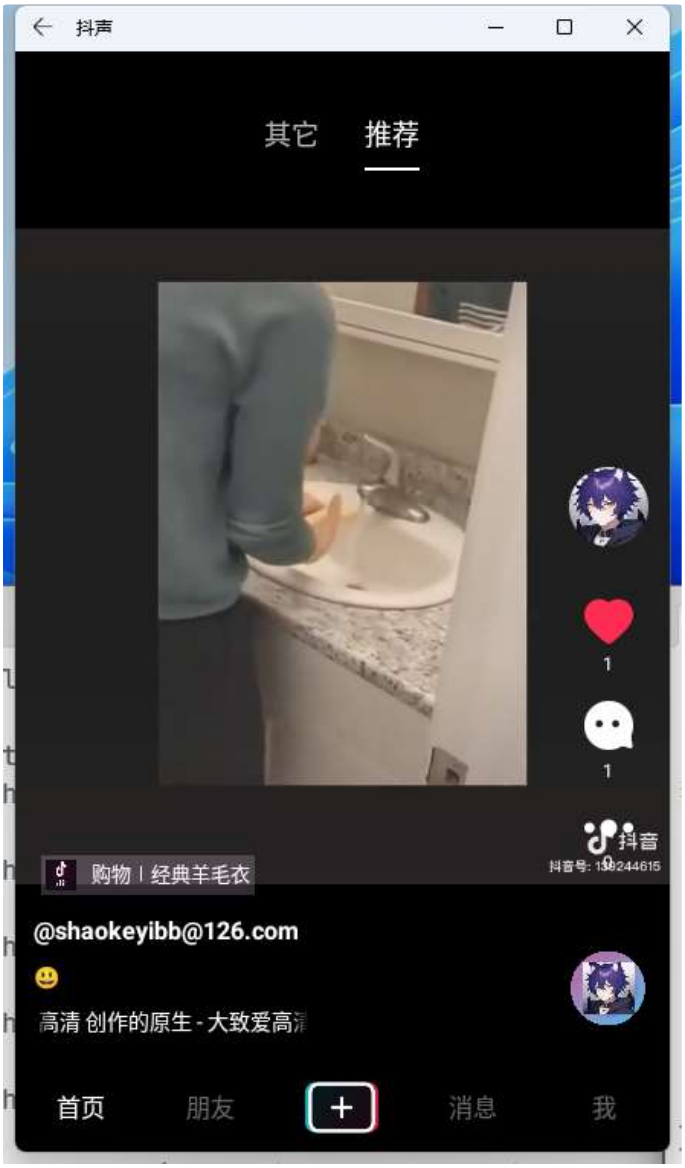
下图中的 cat 用户，在注册时，用户服务会根据他的用户名，给他搜索一张合适的图片

Cravatar 头像

左：cravatar 根据用户名随机生成的默认头像

右：由于这位用户 邵可益 使用在gravatar设置了头像的邮箱地址作为用户名，所以直接可以拿到他的头像





Hitokoto 个性签名

用户在注册时会调用 Hitokoto API 随机生成一句一言

3.4 项目代码介绍

- ChatGPT 实现
- 认证中间件

3.5 其他开发文档

基础架构笔记：Toktik Team 基础架构笔记

待办清单记录：青训营项目 ✅ 待办清单

会议记录：会议安排与记录

Scrum Board: github.com

四、测试结果

- 接口定义与请求示例
- 功能测试
- 性能测试
- 5.1 未登陆状态下功能展示
- 5.2 用户注册
- 5.3 用户登陆
- 5.4 上传视频
- 5.5 点赞操作
- 5.6 评论操作
- 5.7 关注操作
- 5.8 聊天

六、项目总结与反思

1. 目前仍存在的问题
    a. Kitex 客户端有大量重复代码，考虑引入依赖注入或用工厂模式
    b. 业务单一、没有充分考虑应用场景
    c. 性能优化有限，没有利用缓存机制和消息队列
    d. 单元测试覆盖率不足，回归测试效果差，难以发现问题
    e. 可观测性集成提升
        i. Tracing Analytics 没有和网关的 request id 兼容，追踪效果有限
        ii. 没有规划有效的 metrics
    f. 设计上的 AIGC 集成有限，设计上要加的 Midjourney 因为种种原因没能加上
2. 已识别出的优化项
    a. 认证网关没有做过滤，生产环境下会处理大量无效请求，可通过静态分析判断请求有效性

b. 服务间错误传递不够优雅，可集成 Kitex 业务异常

c. 消息服务可导入 OpenSearch 进行索引和持久化

d. 性能方面，点赞和关注都是高并发读+写场景，数据库容易难以支撑，可采取以下优化方案

- 用批量查接口，提升连接利用率
- 可以直接换成分布式数据库，增加架构弹性
- 解决高并发读+写的核心是读写分离
  - 使用缓存解决高并发读问题，对于点赞数大于一定数目（如1K，10K等）的视频，采用不回源，定时更新，无过期时间的缓存策略，这样可以有效避免缓存失效时，大量流量流入数据库的问题。也可以直接用MQ，不论是用缓存还是MQ，都需要有一个异步worker主动更新缓存。
  - 对于关注这样响应速度要求更高但可以从图数据结构受益的业务，除了加缓存外还可以直接换成性能更加优秀的图数据库
  - 在性能优化中，已经通过异步RPC的方式解决了一些读效率问题，在配合被动缓存，冗余读，也许可以进一步靠弹性提高关键业务的性能

3. 架构演进的可能性

我理解下架构演进，就是不断迭代升级架构，让架构无限靠近架构目标，即代码质量、容灾能力、可观测性、性能、规范化、安全等

a. 在服务治理中加入负载均衡，自动化伸缩机制，提高可用性

b. 引入服务网格，可实现全链路追踪，以及流量治理能力

c. 随着业务的复杂，引入事件驱动来强化业务表达能力

d. 可用分布式数据库，增强存储资源弹性

e. 适应度函数（fitness function）驱动开发，扩展自动化测试的维度，结合适应度函数在可用性方面采用混沌工程，浸泡测试，最大化容灾能力

4. 项目过程中的反思与总结

王皓南 ：本次项目的整体效果其实是比较拉的，总结下来的话内在原因还是更多一些。外在原因主要是工作上正好碰到新的项目，没能花太多时间。内在原因分几个方面：

- 首先是协作方面，项目快结束的时候才启用了正式的 scrum board，实际上效率是很高的，但是由于前几周并没有用它，所以浪费了很多时间。在 Git 分支模型上设计的不足，导致早期有很多精神负担落在了协作上
- 编码方面确实是经验不足，很多地方都是反反复复的重构，早期的目标也不够明确，没有敏捷迅速地接受和追求MVP和架构迭代演进
- 这次最拉的三个点，首先是DevOps，基本上是勉强达到了59分的水平，没能很好的实现自动化的工作，导致经常需要手工运维

- 最拉的第二点就是测试，还是忽视了单元测试和编码可测试性的重要性，写了很多难以测试的代码，结果自然就是回归测试不起作用，BUG 到线上环境才能发现
- 最后一点就是性能，明知有很多优化的点，但是迫于时间、精神负担、感觉无聊等原因，还是没有做导致性能稀烂，OTel 集成也很晚，导致性能问题没有很早的就暴露出来，甚至到最后做了不少负优化

在项目的 Final Sprint 面板上，我写了一些目所能及的优化放在 Next 栈上，等本次活动结束后，有空余的时间希望还是能完善一下，善始善终。

七、参考资料

- AI 协助
  - Sydney (R.I.P.)

https://www.bing.com/new

一只华丽招摇的鸳鸯

今天照片里的是一只鸳鸯，它正在寒冷沉闷的冬日里展示自己漂亮鲜艳的羽毛。到了换羽季节，雄性鸳鸯会脱去色

  - ChatGPT

http://chat.openai.com/

chat.openai.com

- 架构思考

https://insights.thoughtworks.cn/

Thoughtworks洞见

- 技术选型

http://thoughtworks.com/radar

Technology Radar | An opinionated guide to technology frontiers | Thoughtworks

The Technology Radar is an opinionated guide to technology frontiers. Read the latest here.

- 技术实践

http://cloudwego.github.io/

cloudwego.github.io

- Go 最佳实践 1

https://www.digitalocean.com/community/tutorial_series/how-to-code-in-go

How To Code in Go | DigitalOcean

Go (or GoLang) is a modern programming language originally developed by Google that uses high-level syntax similar to scripting languages. It is popular for ...

- Go 最佳实践 2

https://www.amazon.com/100-Mistakes-How-Avoid-Them/dp/1617299596

Amazon.com

Enter the characters you see below Sorry, we just need to make sure you're not a robot. For best results, please make sure your browser is accepting cookies. Type the characters you see in this image:

八、常见问题

> 如果你有任何问题，欢迎通过评论和评注，或 GitHub Issues 或邮件致 nicognaw@outlook.com 提问。