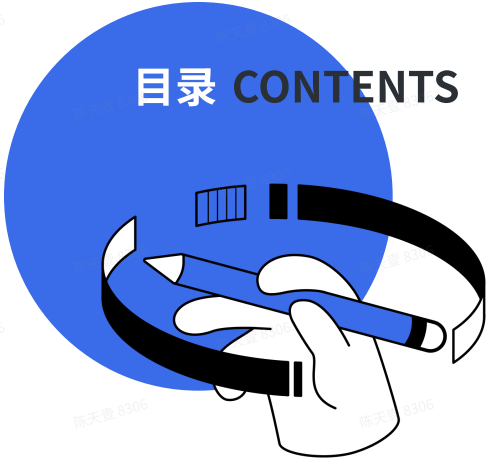
抖音电商商家客户端 Rust 实践

陈天壹（七桑）

2023.11.18

目录 CONTENTS

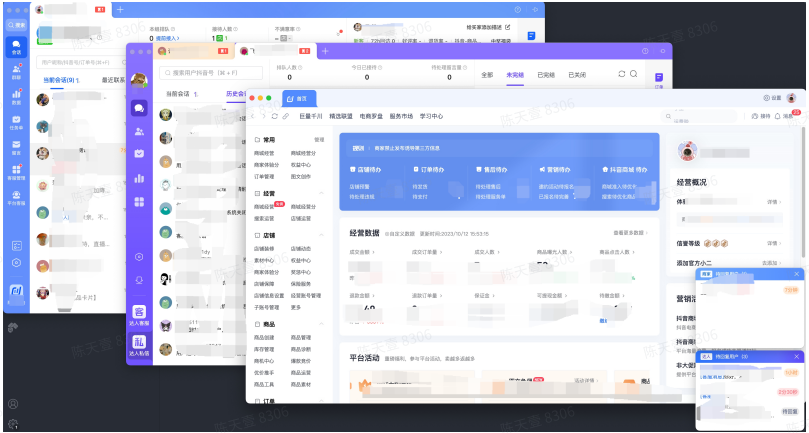


01 方案调研

抖音电商商家客户端 介绍

抖音电商商家桌面端是面向专业商家/客服多店铺高效经营、本地化等场景提供的本地PC客户端

完整的抖音电商商家店铺管理与商家客服能力，并提供多开, 多店铺、多身份、快速跳转，页面复用等功能，目标提高接待服务效率



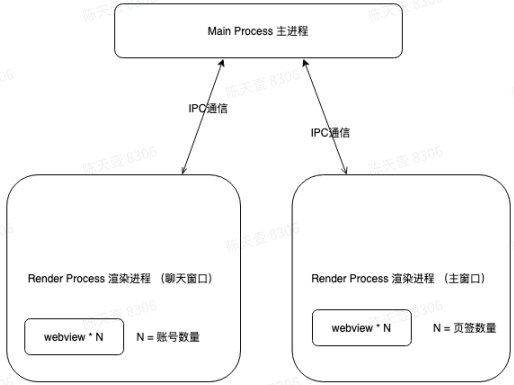
抖音电商商家客户端 业界方案对比

方案	Native+CEF+web	QT+CEF+web	Electron + Web	Tauri + Web
优点	• 可以使用Web技术构建应用程序界面 • 高性能、高天花板 • 可以直接使用更多底层功能	• 可以使用Web技术构建应用程序界面 • 用QT作为本机UI库，可实现良好的跨平台支持 • 具有丰富的QT和CEF社区支持	• 可以使用Web技术构建应用程序界面 • 具有强大的社区支持和丰富的第三方插件 • 跨平台 • 上手快、门槛低、迭代快、效率高	• 可以使用Web技术构建应用程序界面 • 跨平台 • webview2 安装包小 • 安全性高 • 可以通过Rust编程语言进行本机扩展和操作系统交互
缺点	• 开发可能相对复杂、要求高 • 应用程序性能和内存消耗可能受到Chromium引擎的限制 • 跨平台支持可能需要额外的工作	• 学习曲线可能较陡峭 • 应用程序大小可能较大 • 版权	• 应用程序大（捆绑 chromium） • 内存占用大 • 性能可能不如本机应用程序 • 安全性问题需要特别关注	• 起步阶段、生态一般 • 能力相较于 electron 还是较弱，需要不断迭代。 • 熟悉Rust编程语言

抖音电商商家客户端 当前技术栈

Web React

Electron + Nodejs



抖音电商商家客户端 架构优缺点

门槛低、前端友好

功能复用性强

跨平台兼容性强

开发体验良好

发布、更新效率高、便捷

享受 web 生态

包体积大

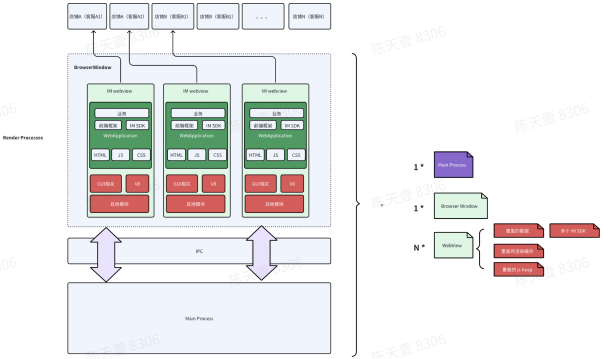
内存大户

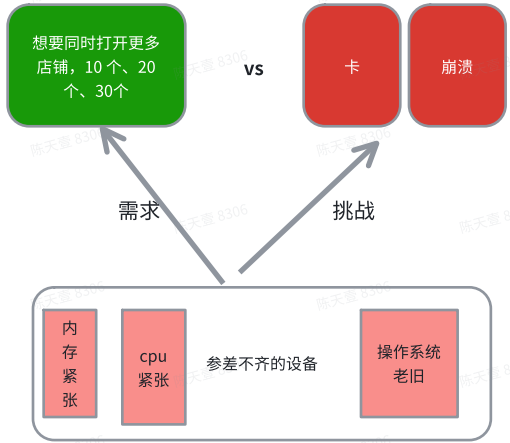
性能问题、天花板低(较原生应用)

机器配置要求高

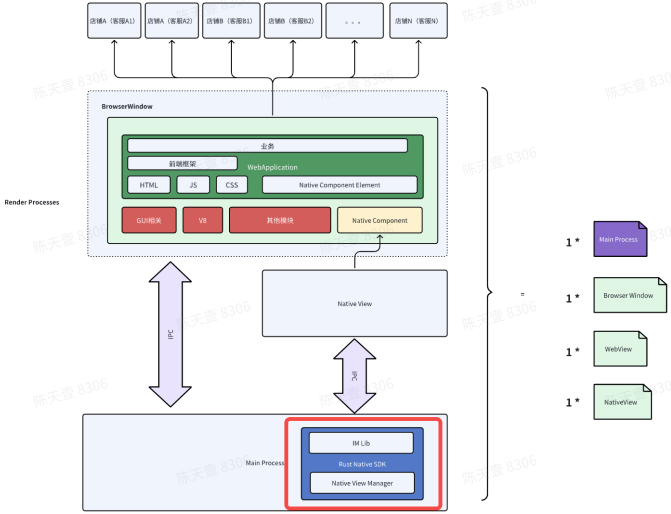
抖音电商商家客户端 问题&挑战

功能 & 性能 & 稳定性





抖音电商商家客户端 思考



技术栈渐进式改变

核心服务sdk化，native 化。
兼容各种技术栈

丰富完善 Native SDK

Native + X
去 node 、Electron

超级 APP

抖音电商商家客户端 SDK 目标

- 提供 Native 基础建设 、降低后续桌面端方案改造成本
- IM 核心服务 native 改造, 更稳定、体验更流畅
- 抬高客户端性能天花板、提供给业务更多更优的实现手段

性能、跨平台能力、ffi 、安全、生态

抖音电商商家客户端 为什么选 Rust 来实现 Native SDK

高性能

无运行时　无 GC　机器代码代码高度优化

零成本抽象　支持并发编程

内存安全

丰富的类型系统　所有权模型

强大的编译器

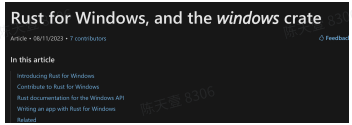
跨平台

Windows　MacOS　Linux

wasm　IOS　Android

社区环境

Last year we wrote about how moving native code in Android from C++ to Rust has resulted in fewer security vulnerabilities. Most of the components we mentioned then were system services in userspace (running under Linux), but these are not the only



抖音电商商家客户端 使用 Rust 需要注意什么?

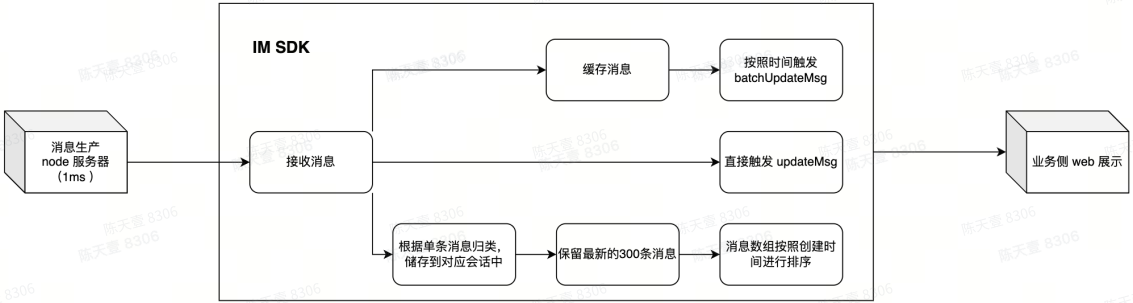
- **学习曲线**：Rust 可能对一些开发者来说有较陡峭的学习曲线，因为它引入了许多新的概念，如所有权和借用。

- **生态**：与 Nodejs 相比，Rust 的生态系统相对较小。开发者可能需要编写自己的库或使用不太成熟的库。（字节背书，有相对比较丰富的业务实践）

- **异步编程**：如果需要进行异步编程，了解 Rust 异步编程的最佳实践和库是很重要的。

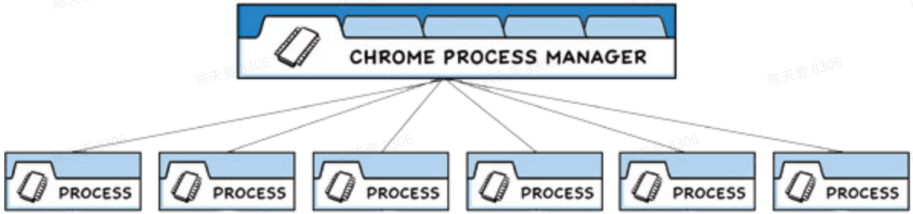
- **更广泛的知识**：网络、操作系统相关、跨平台构建及运行相关、跨语言交互等等

抖音电商商家客户端 概念验证(Proof Of Concept)



抖音电商商家客户端 概念验证(Proof Of Concept)

Electron 进程模型

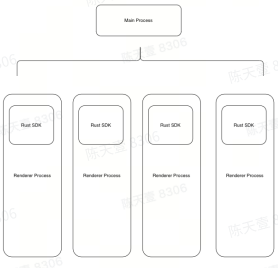


Chrome 漫画

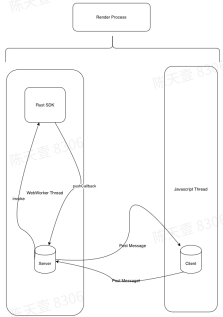
抖音电商商家客户端 概念验证(Proof Of Concept)

应用内业务 SDK 进程、线程模型

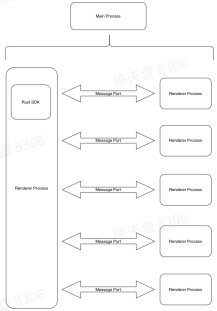
- 单进程单线程



- 单进程多线程



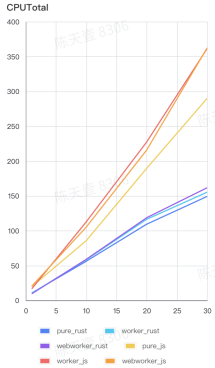
- 多进程

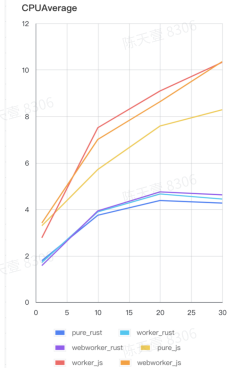


抖音电商商家客户端 概念验证(Proof Of Concept)

结论

- Rust 整体 CPU 优于 js, 随着窗口数变多，优势会放大
- 架构选型上，基于多进程的 worker 方案是最优方案



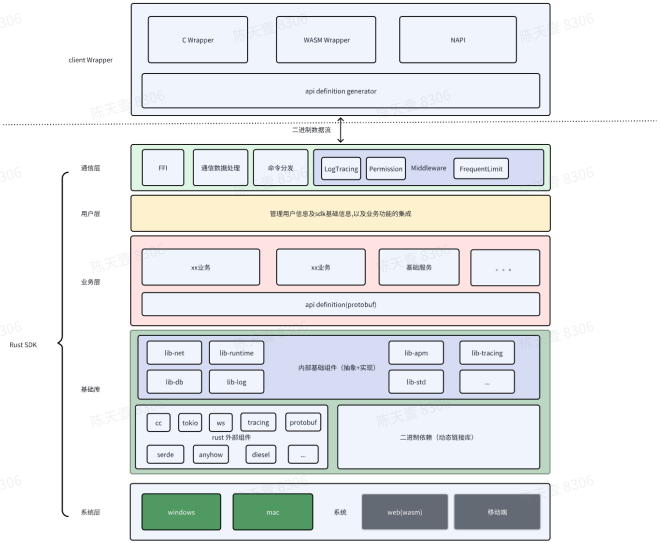


02 架构&集成

抖音电商商家客户端 SDK 架构

分层设计

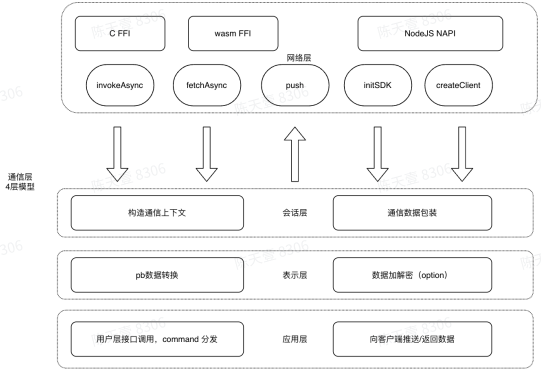
- 高内聚
- 低耦合
- 复用
- 扩展性



抖音电商商家客户端 SDK架构-通信层

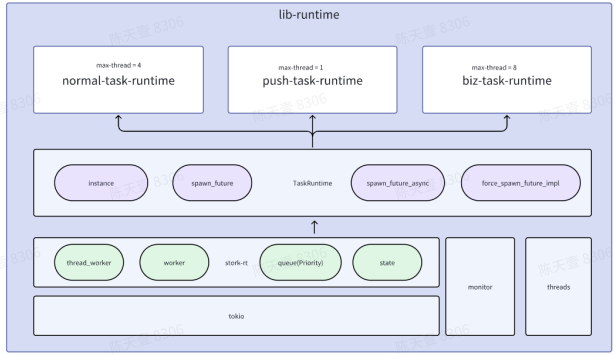
业务接口数据交互存在的问题：

- 复杂数据交互复杂，各个语言数据转化繁琐，会带来一定的门槛及降低开发效率
- 标准：需要统一的标准，降低业务开发的心智负担
- 接口多，不易收敛，难监控、难优化处理

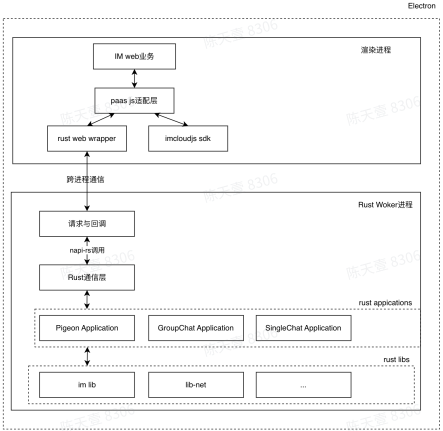


抖音电商商家客户端 SDK架构-异步运行时

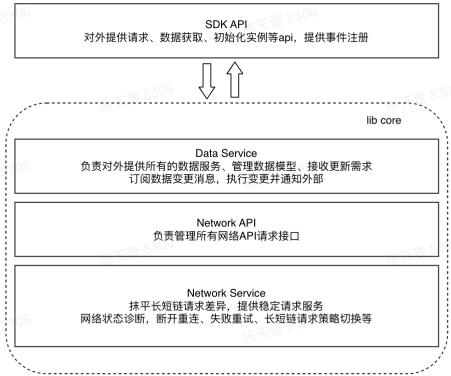
- 执行异步任务
- 分场景使用，各 runtime 不相互影响
- 线程池自动扩缩容
- 支持任务优先级



抖音电商商家客户端 IM 架构



IM 架构



IM lib 设计

抖音电商商家客户端 应用如何集成

Q: Electron 如何与 Rust 交互?

A: node ffi (NAPI)

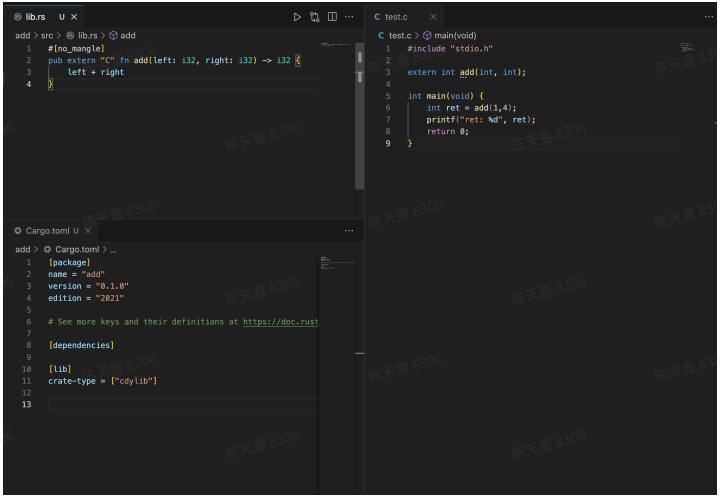
抖音电商商家客户端 FFI

FFI(Foreign Function Interface)是用来与其它语言交互的接口，在有些语言里面称为语言绑定(language bindings)，Java 里面一般称为 JNI(Java Native Interface) 或 JNA(Java Native Access)。
由于现实中很多程序是由不同编程语言写的，必然会涉及到跨语言调用，比如 A 语言写的函数如果想在 B 语言里面调用。

想在 A 语言内调用 B 语言内实现的功能。即在 Node 中调用 Rust 实现的功能。

抖音电商商家客户端 FFI-example

c call rust

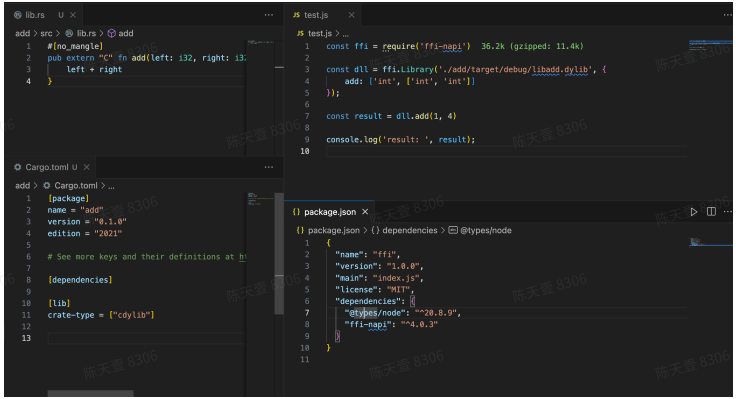



```
➜  add git:(master) cargo build
   Finished dev [unoptimized + debuginfo] target(s) in 0.00s
➜  add git:(master)
```

```
➜  ffi gcc test.c -o test -ladd -L./add/target/debug
➜  ffi ./test
ret: 5%
➜  ffi
```

抖音电商商家客户端 FFI-example

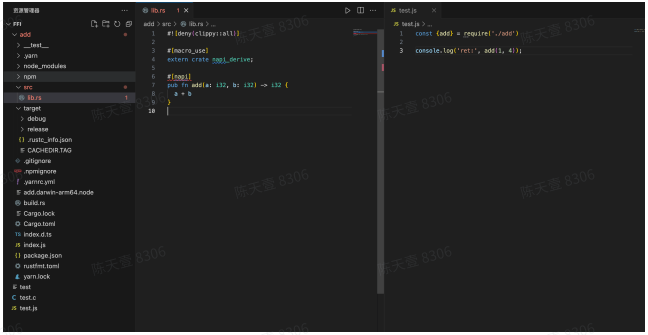
Node call rust

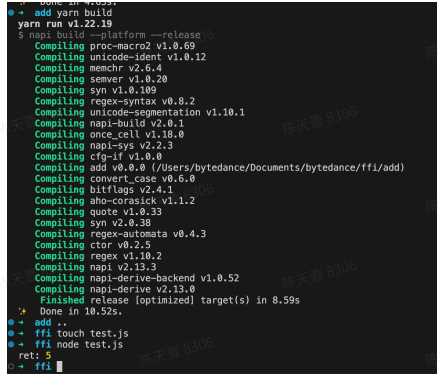


抖音电商商家客户端 FFI-napi

napi-rs

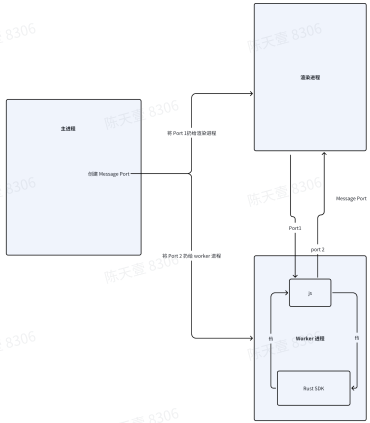
- 技术栈统一
- 支持多平台
- 使用方便，跟使用普通的 npm 包一样





抖音电商商家客户端 Electron 内使用全流程

利用 electron MessagePort API 实现 worker 进程与渲染进程直接通信



03 测试

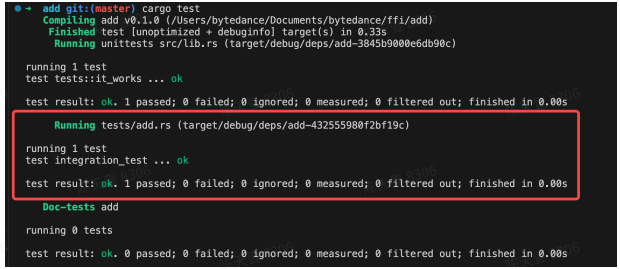
抖音电商商家客户端 单测

```rust
#![deny(clippy::all)]

#[macro_use]
extern crate napi_derive;

#[napi]
pub fn add(a: i32, b: i32) -> i32 {
  a + b
}

pub fn error_add(a: i32, b: i32) -> i32 {
  a-b
}


#[cfg(test)]
▶ Run Tests | Debug
mod tests {
  use crate::{add, error_add};

  #[test]
  ▶ Run Test | Debug
  fn test_add() {
    assert_eq!(add(1, 2), 3);
  }

  #[test]
  ▶ Run Test | Debug
  fn test_error_add() {
    assert_eq!(error_add(1, 2), 3)
  }
}
```

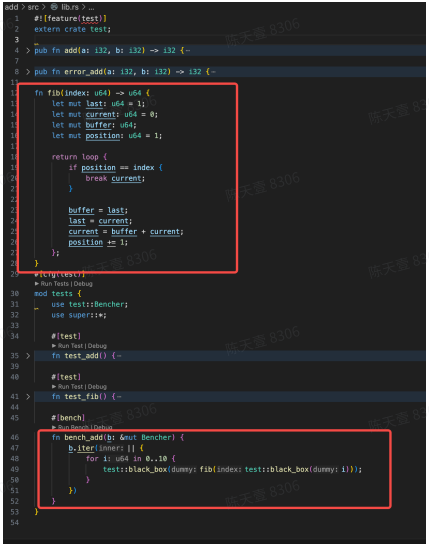
```
➜  add cargo test
   Compiling add v0.0.0 (/Users/bytedance/Documents/bytedance/ffi/add)
    Finished test [unoptimized + debuginfo] target(s) in 0.16s
     Running unittests src/lib.rs (target/debug/deps/add-20707660568c99dd)

running 2 tests
test tests::test_add ... ok
test tests::test_error_add ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

抖音电商商家客户端 集成测试

集成测试是 crate 外部的测试，并且仅使用 crate 的公共接口，就像其他使用该 crate 的程序那样。



抖音电商商家客户端 基准性能测试-Benchmark



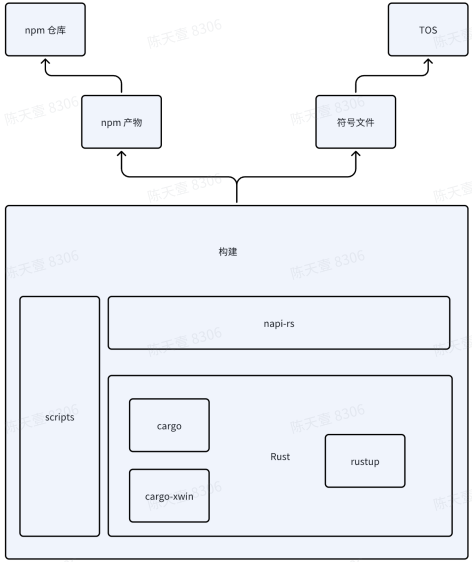

```
add git:(master)
➜  add git:(master) cargo +nightly bench
    Finished bench [optimized] target(s) in 0.00s
     Running unittests src/lib.rs (target/release/deps/add-c8577b9b8f99eb06)

running 3 tests
test tests::test_add ... ignored
test tests::test_fib ... ignored
test tests::bench_add ... bench:       5,682 ns/iter (+/- 211)

test result: ok. 0 passed; 0 failed; 2 ignored; 1 measured; 0 filtered out; finished in 4.32s
```

04 构建&发布

抖音电商商家客户端 构建&发布

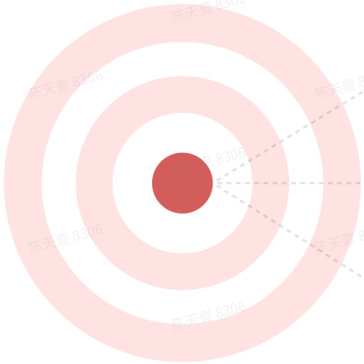
交叉编译

- cross-rs
- cargo-xwin 构建 windows



05 未来规划

抖音电商商家客户端 未来规划



1 丰富业务能力

AI能力、抖店业务 native 化 ...

2 支持更多平台

wasm、ios、android

3 更完善的端监控能力

06 Q&A