

# Go 서버 아키텍처 만들기

근데 이제 리브랜딩을 곁들인

곽웅휘 / 챌린저스



# Speaker



**곽웅휘** challengers / Product cell.


---

 **백엔드 개발 9년 (Go 개발 5년)**

시작은 C언어 네트워크 엔진 개발자  
현재는 Go 백엔드 서비스 개발자

---

## Contact

 010-2754-4937

 gwakuh@hotmail.com

 <http://www.linkedin.com/in/gwakuh/>

# Index

---

## 01. 챌린저스가 리브랜딩한 이유와 배경

습관 형성 앱에서 뷰티 득템 앱으로

---

## 02. 새로운 서버를 구성하기 위해 했던 고민

새로운 서버를 구성하기 위한 기술 선택과 코딩 컨벤션

---

## 03. 기술 부채에 대한 이야기, 앞으로의 방향성

## 04. Q&A



# 01. 챌린저스가 리브랜딩한 이유와 배경

챌린저스 ver1.0

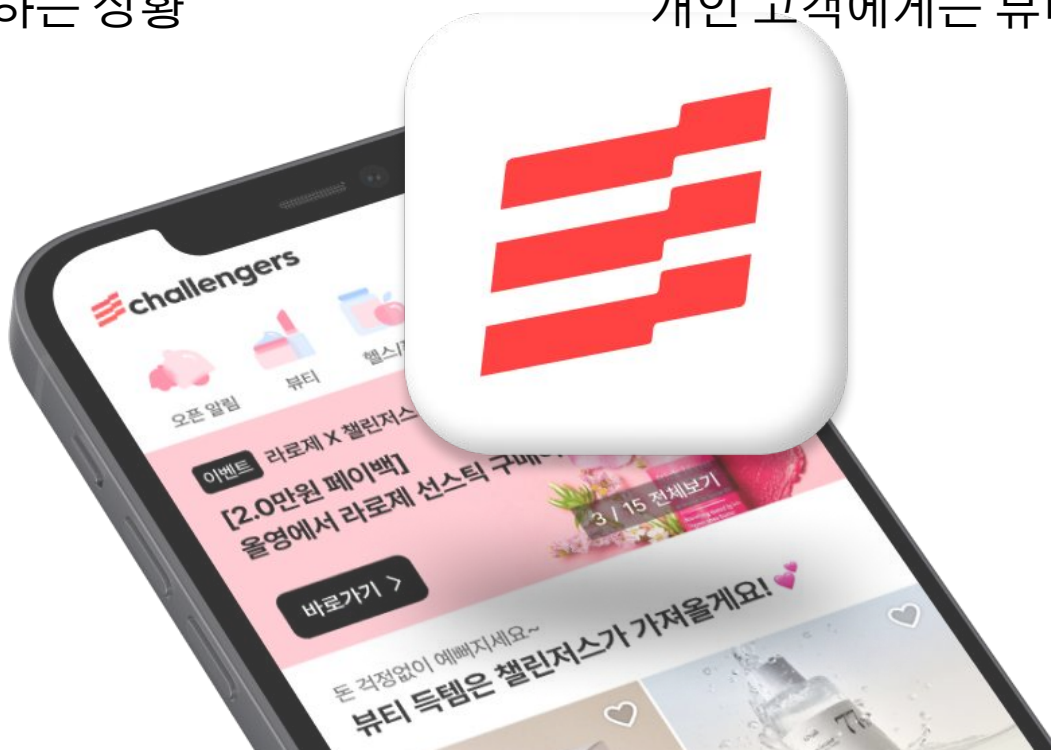
## 습관 형성 앱

많은 뷰티 기업 고객이  
올리브영 랭킹 1위를 원하는 상황

챌린저스 ver2.0

## 뷰티 득템 앱으로의 전환

기업 고객에게는 올리브영 랭킹 1위달성을 보장하고  
개인 고객에게는 뷰티 득템의 기회를 제공



## 02. 새로운 서버를 구성하기 위해 했던 고민

챌린저스 ver2.0

### 뷰티 득템 앱

- 습관 형성 앱에서 뷰티 득템 앱으로 서비스 전환을 위해 남았던 흥터
- 곳곳에 산재해있는 실패한 데이터의 흔적들
- 살아남기 위해, 빠르게 만들기 위해 남겨 놓은 기술 부채



# 고민1. 왜 Go 였을까?

챌린저스 서버 파이썬에서 고로 옮긴 주된 이유가 머에영?

10:24



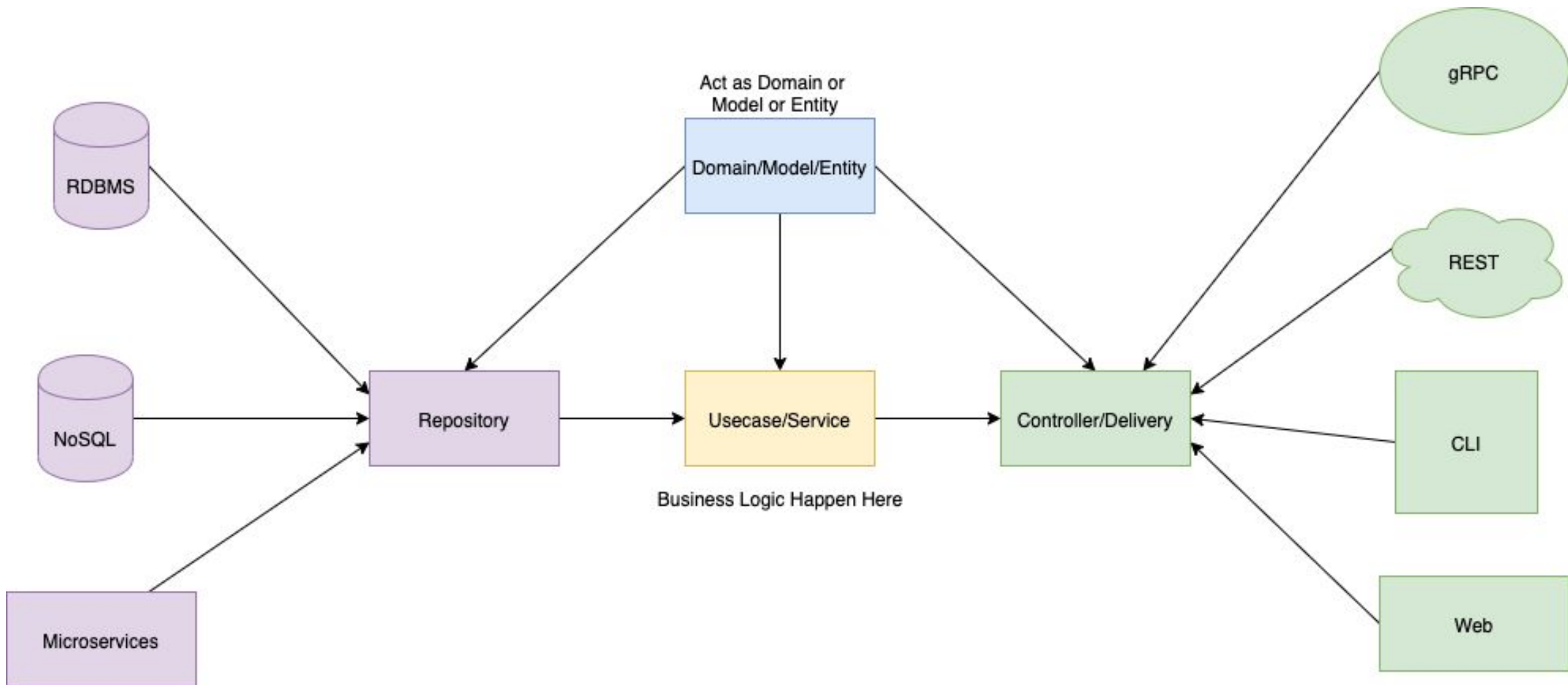
씩씩거리는 무지

복잡한 api들이 생기면서 성능이슈가 발생하기 시작했는데 (람다 기반이라 더 이슈 많았음)  
파이썬 리팩토링을 할까 다른 언어로 넘어갈까  
중에서 고가 그때 한창 화두였어가지고 선택했어요

10:26



## 고민2. 클린 아키텍처 사용





## 고민3. 의존성 주입



Compile-time Dependency Injection for Go

go

golang

dependency-injection

codegen

initialization

● Go · ☆ 12.9k · Updated on 7월 24일



A dependency injection based application framework for Go.

go

golang

framework

service

dependency-injection

● Go · ☆ 5.7k · Updated 16 days ago





## 고민4. HTTP 프레임워크



**Gin** is a HTTP web framework written in Go (Golang). It features a Martini-like API with much better performance -- up to 40 times faster....

go

middleware

performance

framework

router

● Go · ☆ 78.2k · Updated 9 days ago



lightweight, idiomatic and composable router for building Go HTTP services

go

api

golang

http

middleware

● Go · ☆ 18.3k · Updated 6 days ago



## 고민5. ORM

	GORM	SQLBoiler	Jet
타입 안정성	제한적	우수	우수
쿼리 작성 방식	ORM 방식 자동화 쿼리 작성	코드 생성 방식	빌더 타입
성능	일반적	우수	우수
유연성	ORM 기반 자동화된 기능	자동 코드 생성	우수
사용 난이도	쉬움 (ORM 추상화 제공)	중간 (초기 설정 및 코드 생성 필요)	중간 (SQL 직접 작성, 복잡한 쿼리 가능)
레퍼런스	매우 활발	활발	적음



# 그 외

- 문서화: [github.com/swaggo](https://github.com/swaggo)
- 설정 및 환경변수: [github.com/spf13/viper](https://github.com/spf13/viper)
- 로그: [github.com/uber-go/zap](https://github.com/uber-go/zap)
- 그 외 에러 처리, 로그 처리, 테스트 등 많은 숙제가 남아있어요



# Don't Panic

```
func main() {  
    _, err := fmt.Println("Hello, World!")  
    if err != nil {  
        panic(err)  
    }  
}
```



# Panic & Recovery

```
// RecoveryMiddleware is a middleware that recovers from panics and writes a 500 error to the client
func RecoveryMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        defer func() {
            if err := recover(); err != nil {
                // Log the error and the stack trace
                log.Printf("Panic: %v\n%s", err, debug.Stack())

                // Respond with a 500 internal server error
                http.Error(w, http.StatusText(http.StatusInternalServerError), http.StatusInternalServerError)
            }
        }()
        // Call the next handler
        next.ServeHTTP(w, r)
    })
}
```

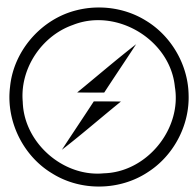


# Named result parameter

```
func ReadFull(r io.Reader, buf []byte) (n int, err error) {  
    for len(buf) > 0 && err == nil {  
        var nr int  
        nr, err = r.Read(buf)  
        n += nr  
        buf = buf[nr:]  
    }  
}
```



# 03. 기술 부채에 대한 이야기, 앞으로의 방향성



서버-클라이언트 간  
양방향 통신



elasticsearch

검색 기능 고도화



RabbitMQ

메시지 큐를 통한 이벤트 소싱



GopherCon Korea 2024



# Q&A

감사합니다 :)

