

チーム遠距離開発

# 目次

- 概要
- でも
- アプリ実装
- 反省
- FetureWorks

# 1. 概要

# 1. 概要

## ・テーマ

# 社会人に役立つアプリ



# 1. 概要

社会人に役立つって何や・・・



# 1. 概要

ということで、

まずは、

既存のアプリ調査

# 1. 概要



ということで、

まずは、

## 既存のアプリ調査

## 1. 概要



ということで、

まずは、



# 既存のアプリ調査



# 1. 概要



ということで、

まずは、



## 既存のアプリ調査

# 1. 概要



ということで、

まずは、



## 既存のアプリ調査



# 1. 概要

- 調査に対する考察

**既存のアプリは「特化型が多い」**

- スケジュール特化
- タスク管理特化
- 婚活特化

# 1. 概要

- 調査に対する考察

既存のアプリは「特化型が多い」

- スケジュール特化
- タスク管理特化
- 婚活特化

→ いちいちアプリ切り替えるのたるくね?? (笑)

# 1. 概要

- 仮定

社会人に役立つ機能をたくさん詰め込んだ

## 多機能スケジュールアプリ

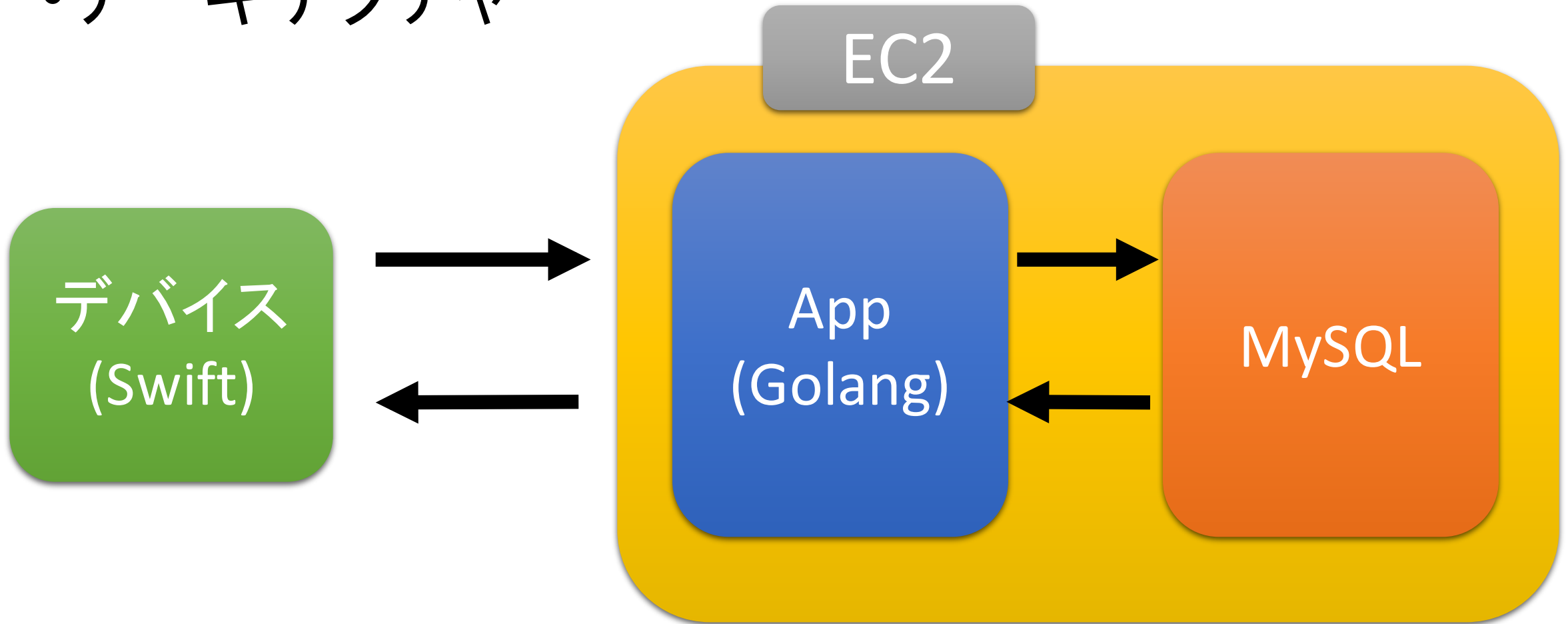
があればイヌケンさんの社会人生活の効率化できる！！

## 2. でもするお

### 3. アプリ実装

## 2. アプリ実装

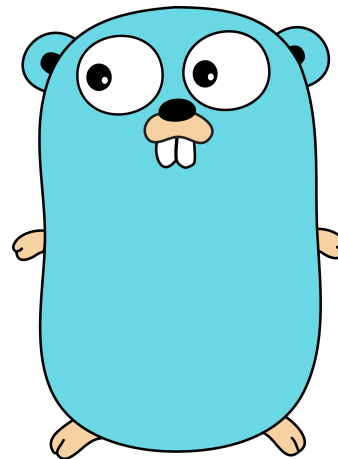
- アーキテクチャ





## 2. アプリ実装

- 取り入れた新しい技術
  - Swift3(フロント)
  - Xcode
  - Golang1.7(サーバーサイド)
  - echo
  - AWS
  - Webhook(Github)



Amazon EC2

## 2. アプリ実装

立ちはだかった大きな壁

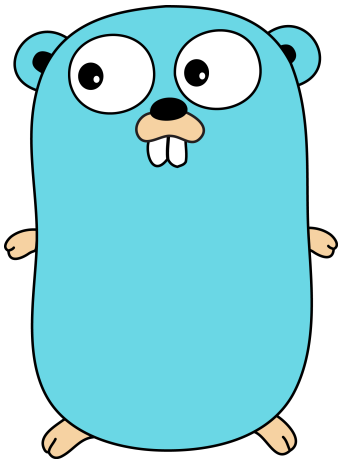
-json-

{JSON}

## 2. アプリ実装

Jsonの扱いが非常に難しかった  
その①

Swift側でJsonが受け取れない



{JSON}



## 2. アプリ実装

### 原因

SwiftがJson受け取りで待っているのに  
対してStringで返し続ける(笑)

## 2. アプリ実装

### 原因

SwiftがJson受け取りで待っているのに  
対してStringで返し続ける(笑)

```
57 if err != nil {
58     fmt.Println(err)
59     fal := json_event{0,"0","0","0","0"}
60     res := json_all{}
61     res.Status = false
62     res.Data = append(res.Data, fal)
63     return res
64 }
65 fal := json_event{0,"0","0","0","0"}
66 res := json_all{}
67 res.Status = true
68 res.Data = append(res.Data, fal)
69 return res

1
2 func Echo_event_regist(db *sql.DB) echo.HandlerFunc {
3     return func(c echo.Context) error {
4         event := initiation(c)
5         status := event.regist_event(db)
6         return c.String(http.StatusOK, status)
7     }
}

model/event_regist.go [+]
[5] 0:vim*
```



```
57 if err != nil {
58     fmt.Println(err)
59     fal := json_event{0,"0","0","0","0"}
60     res := json_all{}
61     res.Status = false
62     res.Data = append(res.Data, fal)
63     return res
64 }
65 fal := json_event{0,"0","0","0","0"}
66 res := json_all{}
67 res.Status = true
68 res.Data = append(res.Data, fal)
69 return res

1
2 func Echo_event_regist(db *sql.DB) echo.HandlerFunc {
3     return func(c echo.Context) error {
4         event := initiation(c)
5         status := event.regist_event(db)
6         return c.JSON(http.StatusOK, status)
7     }
}

model/event_regist.go
[5] 0:vim*
```

## 2. アプリ実装

Jsonの扱いが非常に難しかった  
その②

Swift側でJsonがパースできない



{JSON}



JSON

{ }

## 2. アプリ実装

### 原因

Go側のレスポンスのJson構造体の  
定義の仕方が違う(笑)

## 2. アプリ実装

### 原因

Go側のレスポンスのJson構造体の  
定義の仕方が違う(笑)

```
+ m/json_struct.go
1 package model
2
3 //イベント情報を返すjson構造
4 type json_all struct {
5     Status bool `json:"status"`
6     Data []json_event
```

```
//イベントの詳細な情報用のjson構造
type json_event struct {
    ^Id int
    ^ISummary string
    ^IDtstart string
    ^IDtend string
    ^IDescription string
}
```



```
m/json_struct.go
1 package model
2
3 //イベント情報を返すjson構造
4 type json_all struct {
5     Status bool `json:"status"`
6     Data []json_event
```

```
//イベントの詳細な情報用のjson構造
type json_event struct {
    ^Id int `json:"id"`
    ^ISummary string `json:"summary"`
    ^IDtstart string `json:"dtstart"`
    ^IDtend string `json:"dtend"`
    ^IDescription string `json:"description"`
}
```



## 2. アプリ実装

結果

俺の実装がクソだった(笑)

## 2. アプリ実装

結果

俺の実装がクソだった(笑)

本当に

## 2. アプリ実装

結果

俺の実装がクソだった(笑)

本当に

本当にすまないと  
思っている

すまないと思っている



## 2. アプリ実装

感動した技術  
-自動デプロイ-

## 2. アプリ実装

# Githubのwebhookを利用

Pushきたお(^ω^)



リポジトリにpushすると  
指定したURLを叩いてくれる

## 2. アプリ実装



# 今回の利用方法

外部コマンドの実行の点でpythonを使用  
osライブラリが使いやすい



## 2. アプリ実装

簡単3ステップで自動デプロイ！

- ①Githubの最新状態に更新
- ②実行中のアプリプロセスをkill
- ③アプリを実行

## 2. アプリ実装

### Point

- ① 指定したコマンドのプロセスの検索
- ② 複雑なコマンドはシェルスクリプトで実行



## 4. 反省

- スケジューリングミスったw
- お互い新しい言語という点を考慮すべきだった

## 5. Future Works

- 基本的な機能しか実装できていない  
→ タスク管理, 議事録機能etcの実装
- もっと新しい技術を使いたい  
→ 音声認識で文章化etc

以上

チーム遠距離開発でした  
ご静聴ありがとうございました