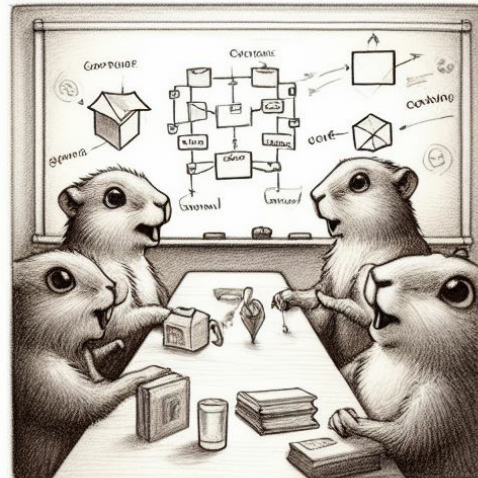
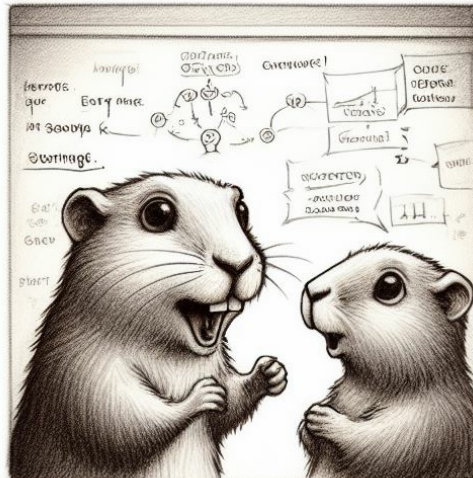
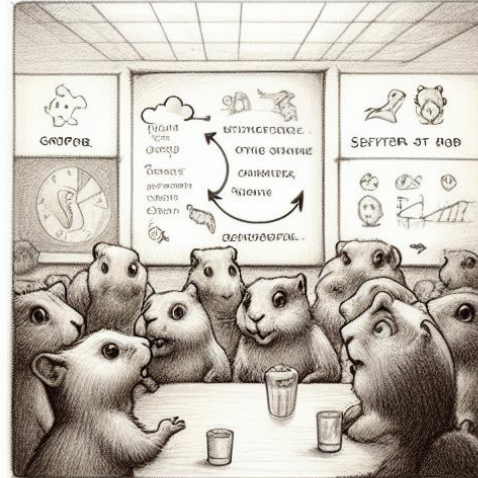
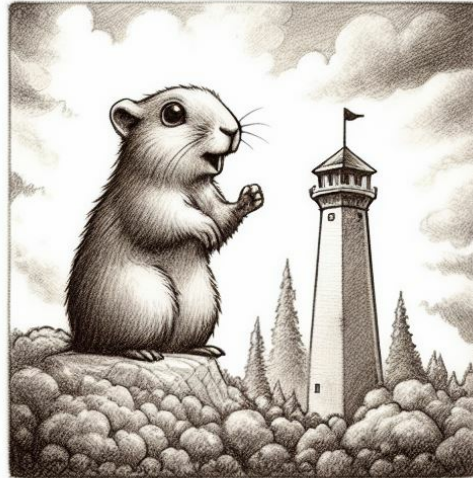


Collaborating on architecture



What are we talking about today?

- Why does collaboration matter?
- When good intentions go south
- Approaches that have helped me
- How to continue conversation in Golang

Why does collaboration matter?

- The best ideas come from clashing opinions
- If everyone is rowing in the same direction,
we will get there faster



When good intentions go south

I believe that we all have good intentions.

And even the flawed approach is better than inactivity.

Large pull requests

I was working on a feature. There was this architectural issue. One thing led to the other, and...

feat(sick-leaves): add database model

Merged

mergify merged 19 commits into `main` from `oct-1294` on May 18, 2023

Conversation 118

Commits 19

Checks 33

Files changed 20

Being too excited for specific solution



Dictating from the ivory tower



Scaling the Practice of Architecture, Conversationally



Approaches that have helped me

Decision records

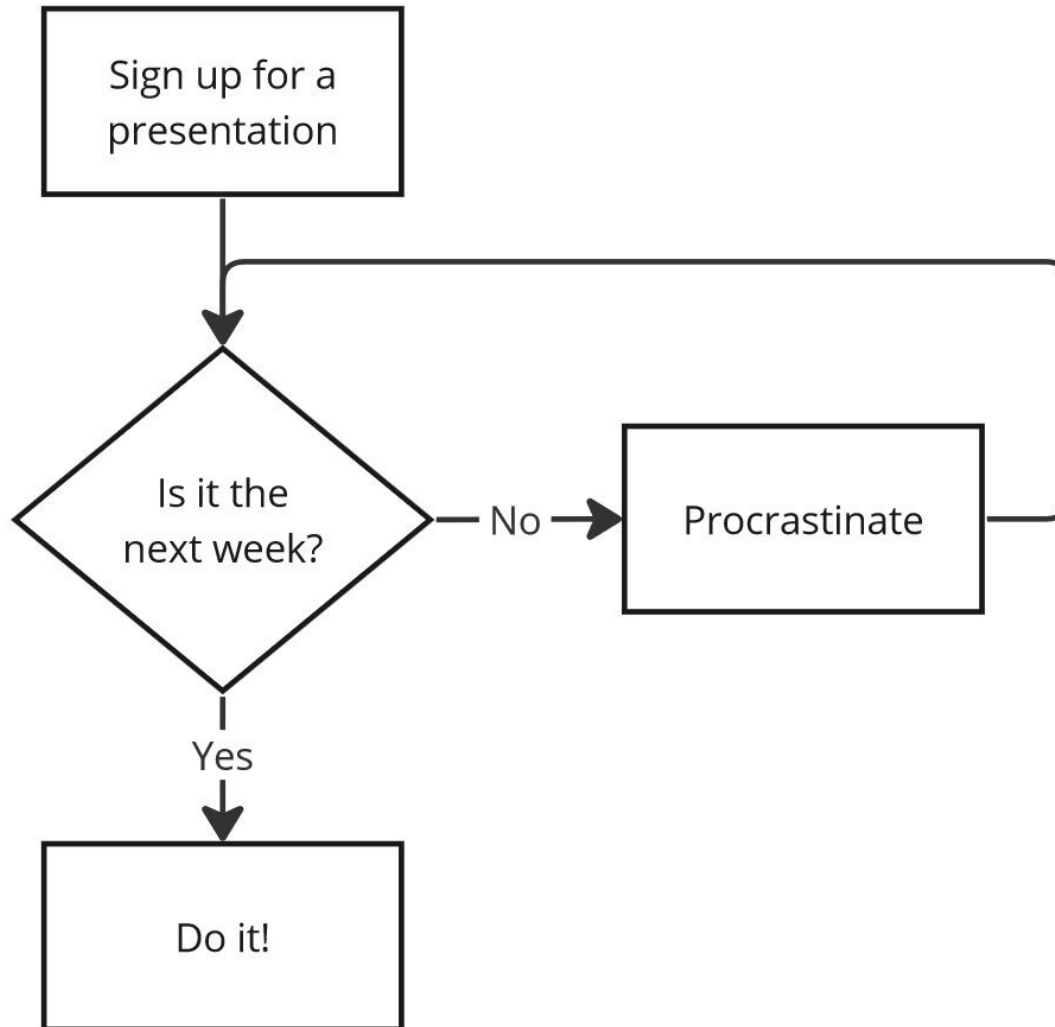
Decision Records

Table +

Aa Name	Status	Authors	Proposed On	Due Date
Implementation of DORA metrics	Draft	D Daniel Antos		
Handling distributed transactions	Proposed	D Daniel Antos	April 3, 2024	April 23, 2024
Isolating tenants data	Accepted	D Daniel Antos	August 15, 2023	August 23, 2023
Handling external API rate limits	Implemented	D Daniel Antos	January 7, 2024	January 17, 2024
Tool for managing translations	Needs revising	D Daniel Antos	April 5, 2023	April 27, 2023

[Read more on my blog](#)

Diagrams



Proof of Concept (PoC)



temporal-poc

Public



Pin



Unwatch

1



main



Go to file



<> Code



antosdaniel feat: add payment processing

ca7a6ca · last week



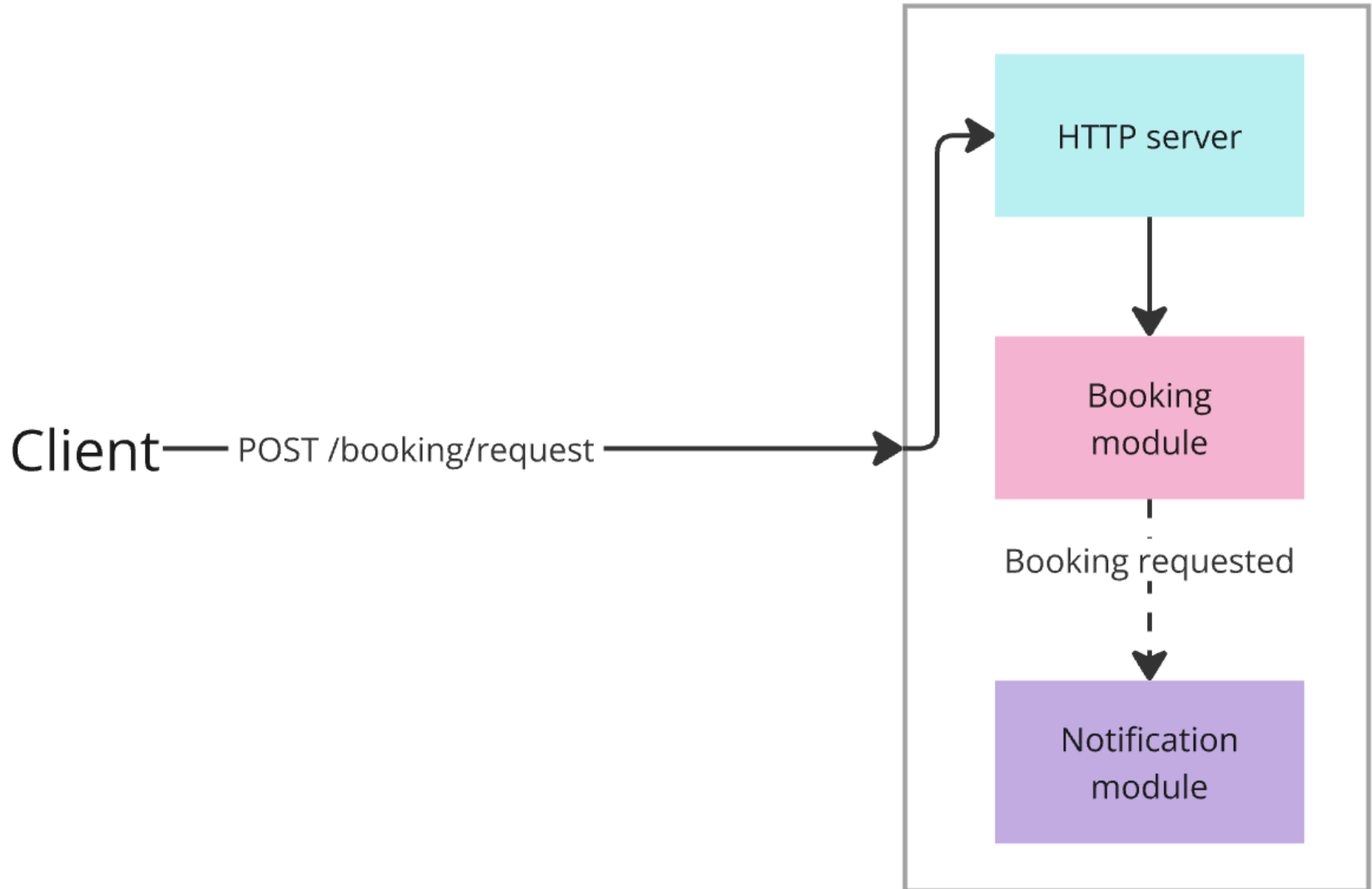
10 Commits



Themes for these approaches

- Show the journey.
- What are the worst ways to solve the problem?
- At which point the other opinion is better?

How to continue the conversation in Go




main.go can tell a story about your service

```
1 func main() {
2     s := setup.New()
3
4     notification, err := notification_setup.NewSetup()
5     if err != nil {
6         log.Fatalf("Could not set up no module: %v", err)
7     }
8
9     booking, err := booking_setup.New(s.Enqueueer, notification.API)
10    if err != nil {
11        log.Fatalf("Could not set up booking module: %v", err)
12    }
13
14    go booking.ListenForEvents(s.Dequeueer)
15    go notification.PeriodicRetry()
16
17    metrics.RegisterRoutes(s.Router)
18    booking.RegisterRoutes(s.Router.Group("/booking"))
19    s.StartServer()
20 }
```

Restrict access

```
type API interface {  
    SendNotification(templateID string, userID string) error  
}
```



```
type APIImplementation struct {  
    TemplateRepo TemplateRepo  
    UserRepo      UserRepo  
    SmsSender     SmsSender  
}
```

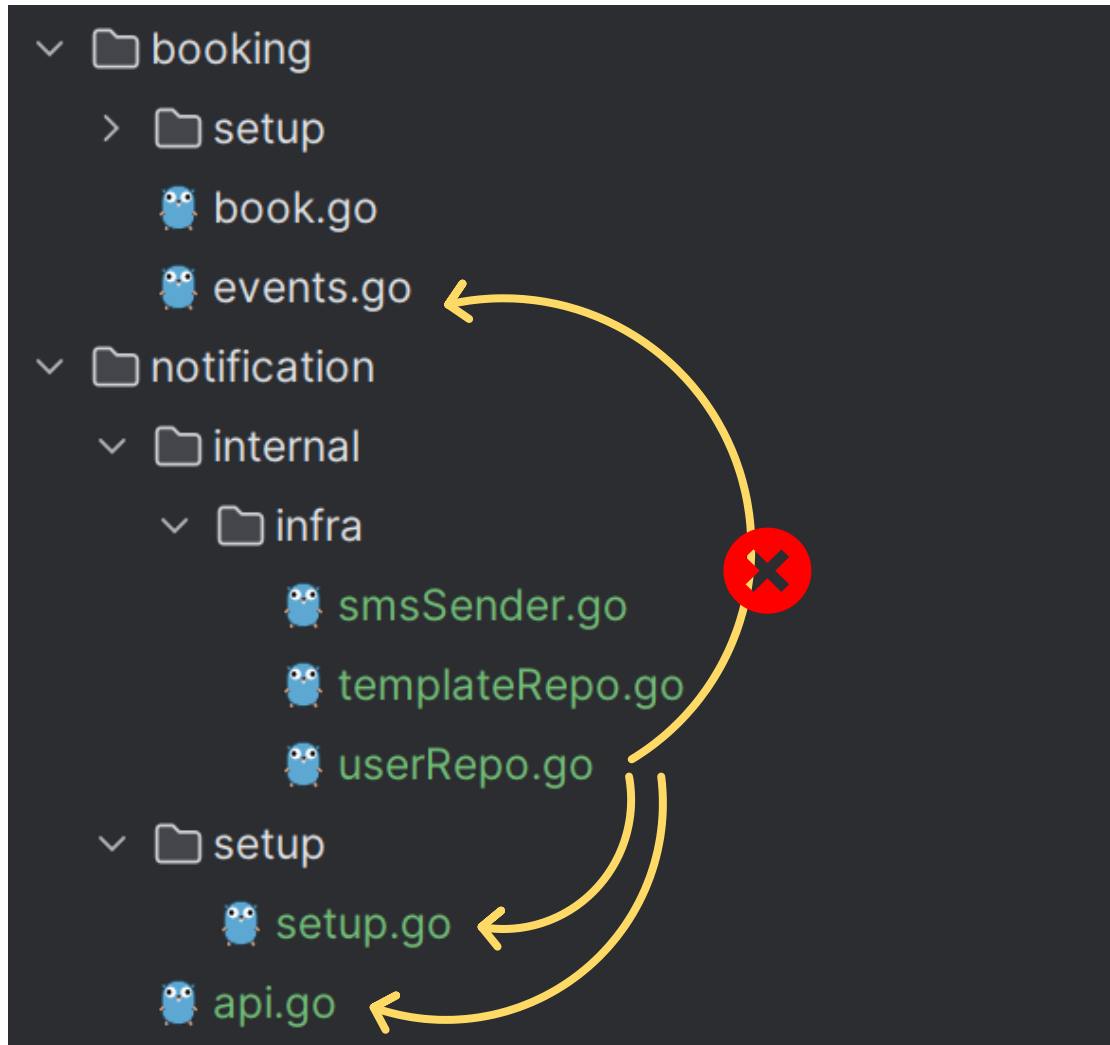
```
notifications := api.APIImplementation{  
    TemplateRepo: infra.TemplateRepo{},  
    UserRepo:     infra.UserRepo{},  
    SmsSender:    infra.SmsSender{},  
}
```

```
type api struct {  
    TemplateRepo TemplateRepo  
    UserRepo      UserRepo  
    SmsSender     SmsSender  
}
```

```
func New(templateRepo TemplateRepo, userRepo UserRepo, sender SmsSender) (API, error) {  
    return withLogs{  
        API: api{  
            TemplateRepo: templateRepo,  
            UserRepo:      userRepo,  
            SmsSender:     sender,  
        },  
    }, nil  
}
```

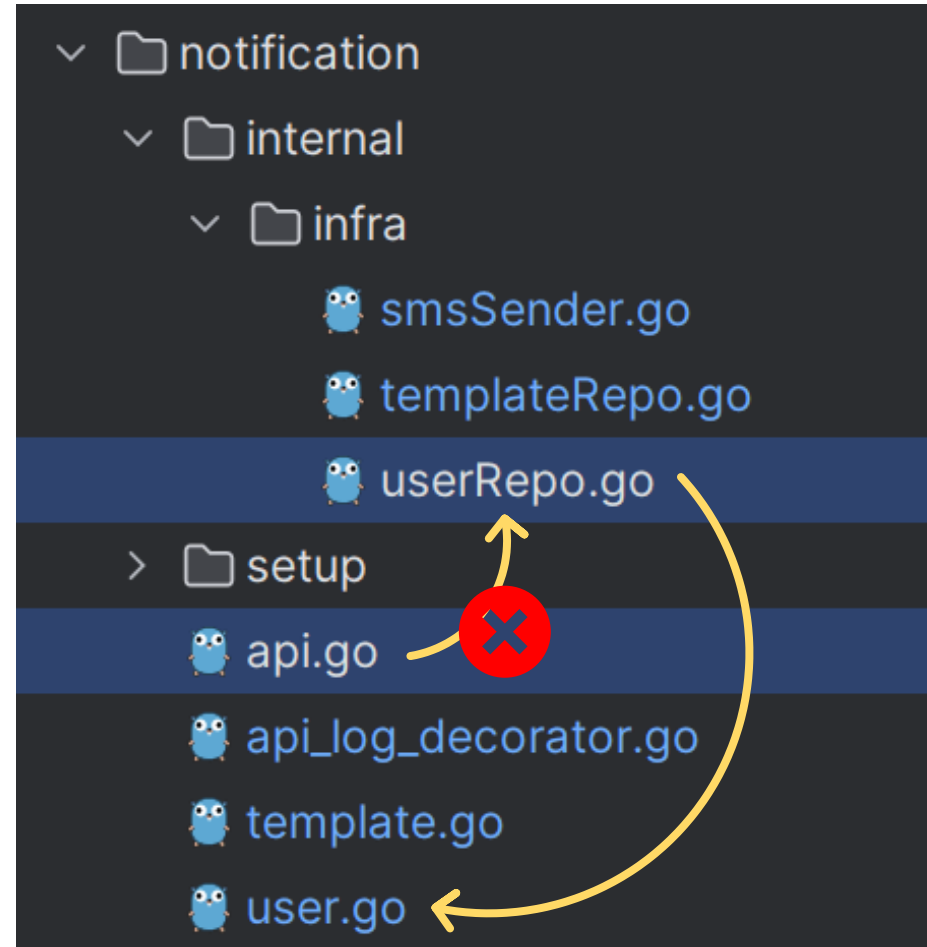
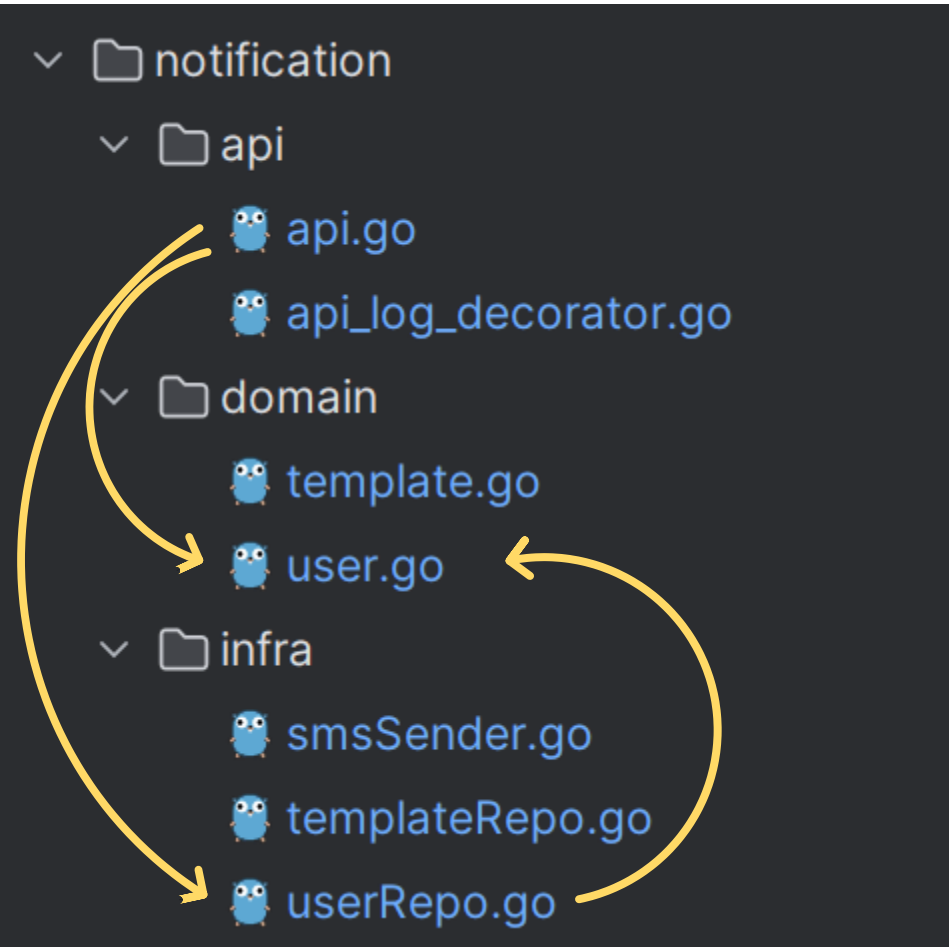
We control creation,
user knows only about the interface

Restrict access



"internal" folder restricts who can use it

Use cyclic dependency to express (limit) what developer can do



Summary

- Don't do it alone
- Find a way to collaborate that fits **you and your team**
- The best architecture doesn't matter, if you don't have buy-in
- Expressing ideas in the code explicitly lengthen their lives
- Don't do it alone

Thanks!

Let's have some questions!



My blog

danielantos.com



Slides