1. Authentication : f(credentials) -> ~~UserId~~ -> Optional<UserId>
2. Authorization

login + password | token | ....

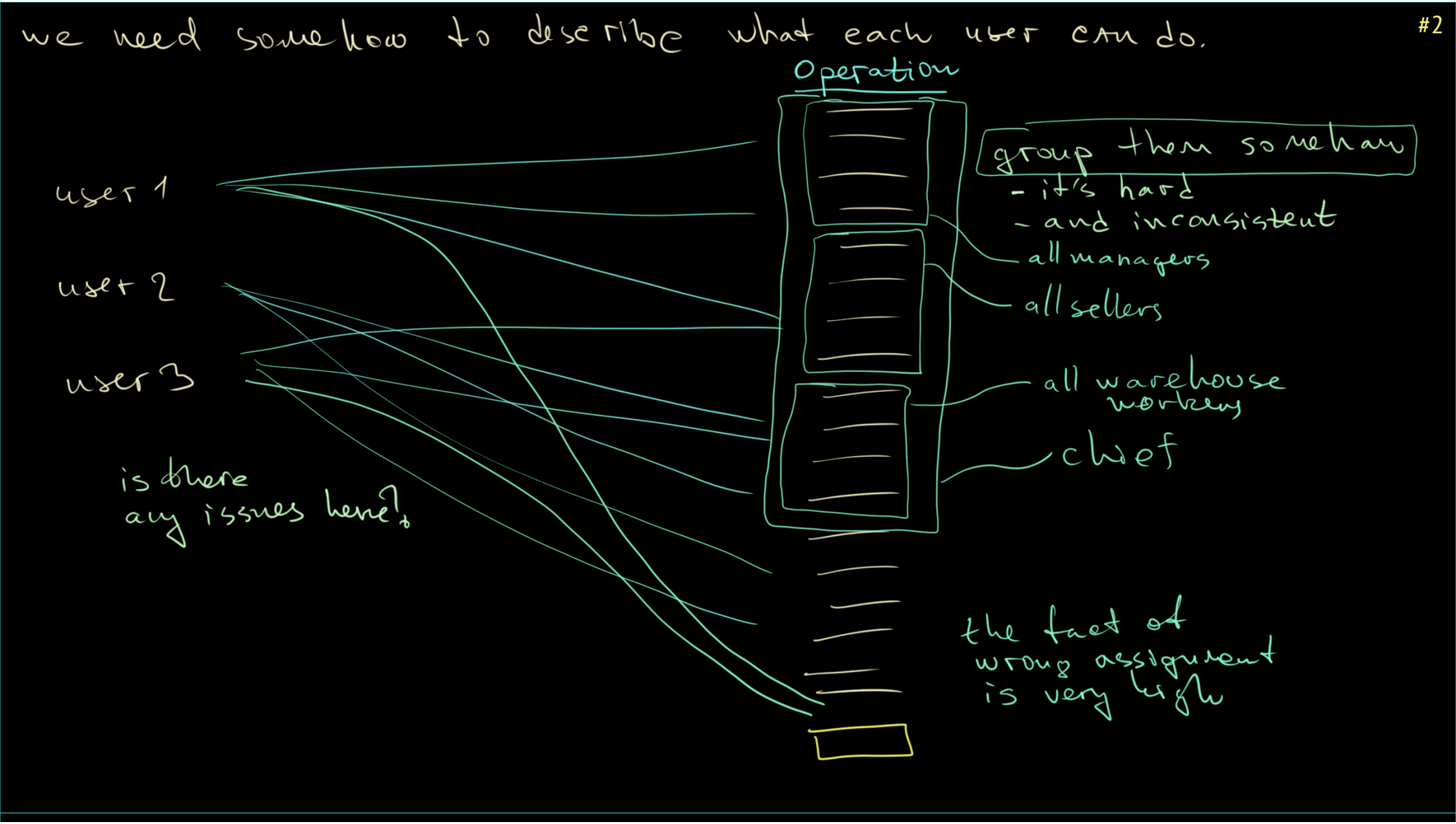
Authorization : f(userId) -> Set<Roles>

empty Set( )

Set<Permissions>

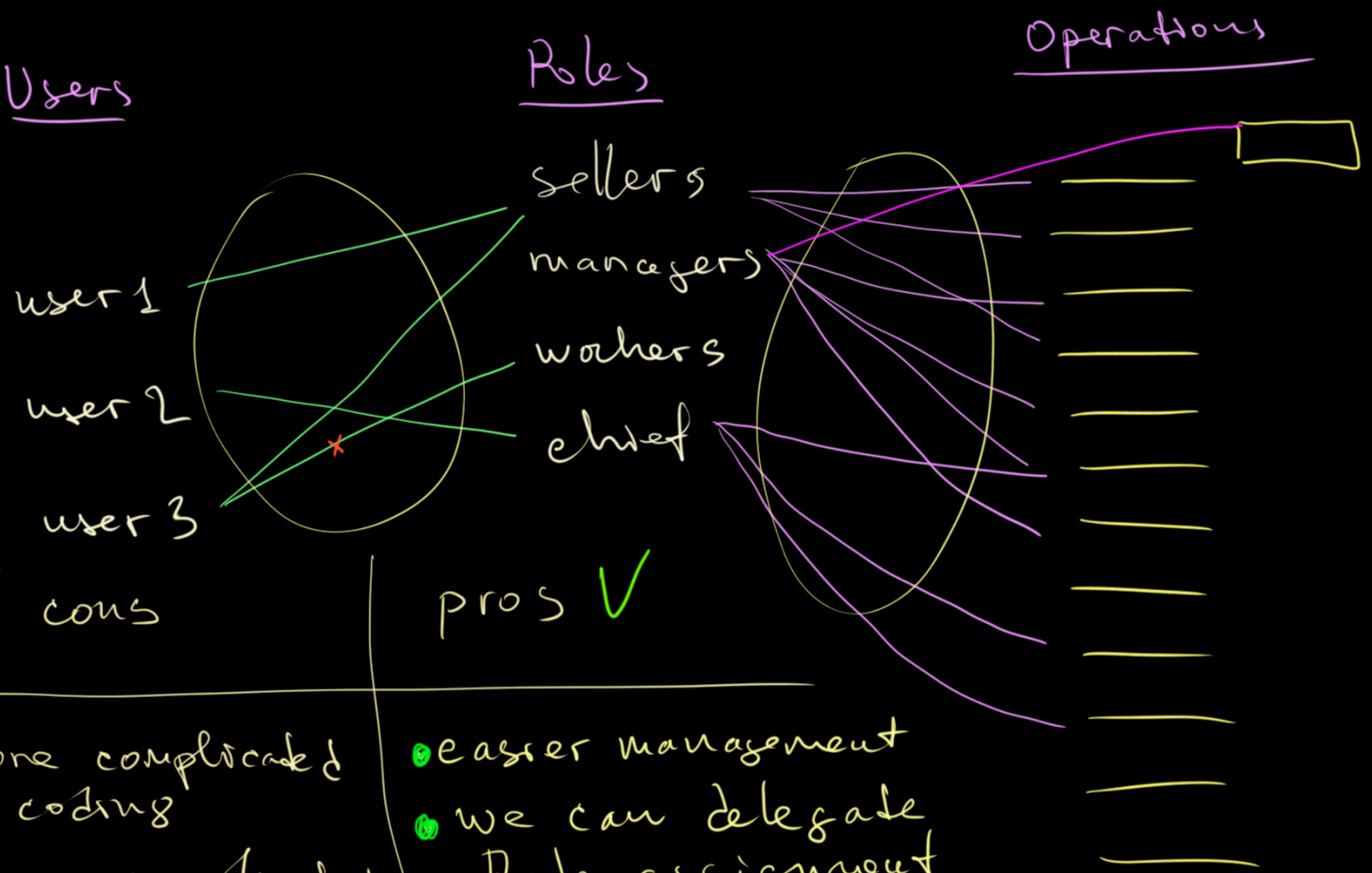
Set of Possible operations we are allowed.

we need somehow to describe what each user can do.



# Role-based Authorization

## RBAC




### cons

- more complicated coding
- more complicated db structure.

### pros

- easier management
- we can delegate Role assignment and Oper. assign to different person
- faster
- less error prone

1. table User
2. table Role
3. table Operation

} Dictionary tables

---

4. table w/ relation → User ⇄ Role
                       Role ⇄ Operation

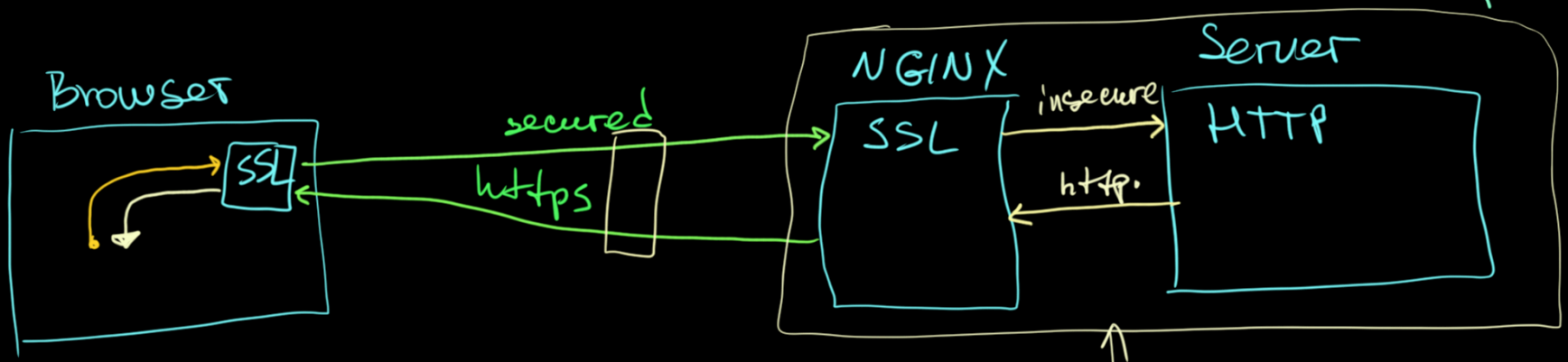
} Real Data w/ Permission/Role

5.

new user

new role

new operation




in the same cluster / same machine

# Web HTTP

how can we pass credentials

1. query parameter `?user=jim&passw=123`
2. form

```
url form encoded
user=jim
passw=123
```

3. Header Authorization `user=jim passw=123`

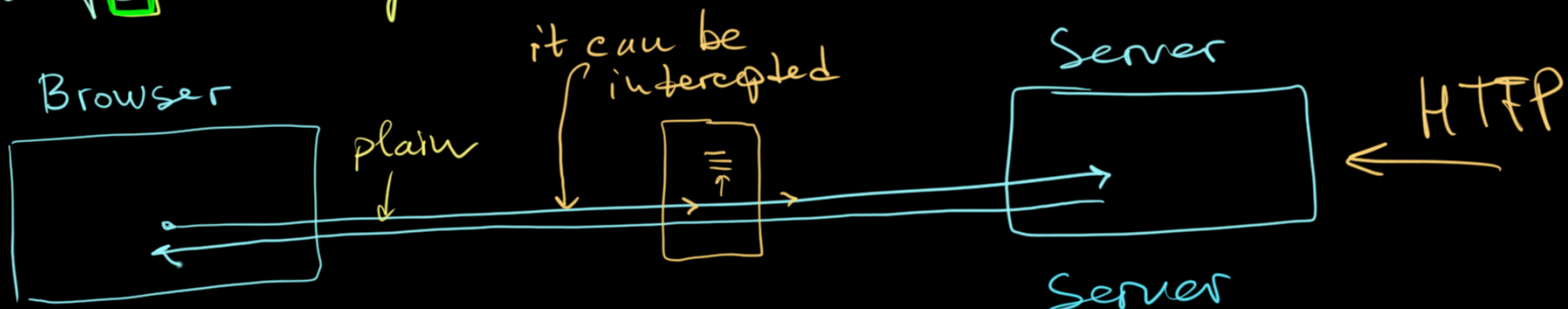
this info will appear in all logs — extremely unsafe

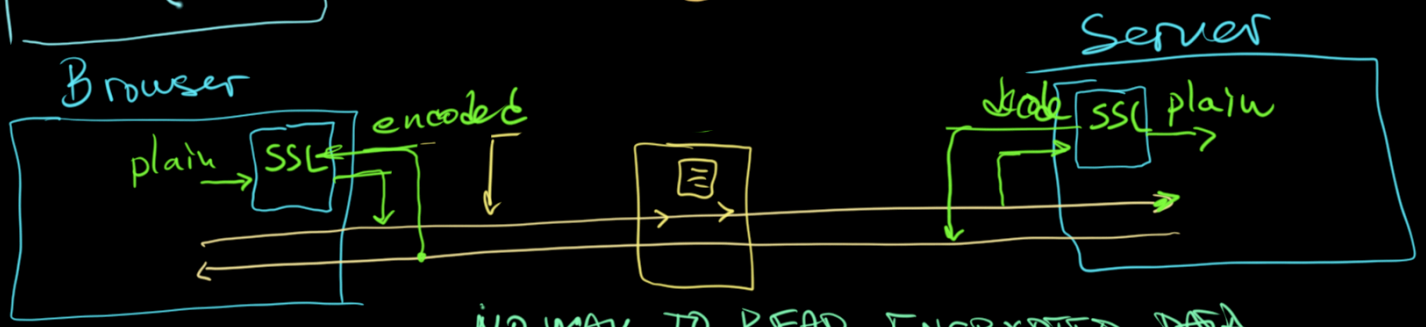
better, it will not show logs

can be intercepted

can be intercepted

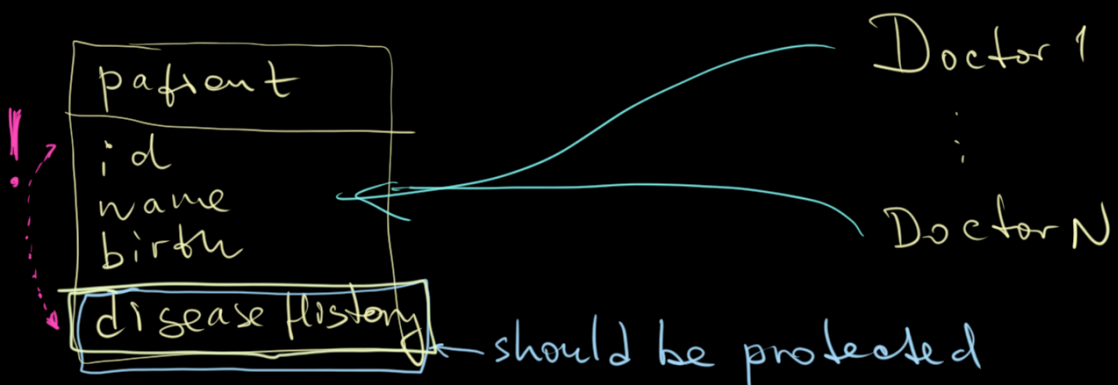
## https security




NO WAY TO READ ENCRYPTED DATA

# Medical Application




we need more tables

Doctor 1 — Doctor
Doctor 2
Doctor 3

DoctorId → Entity { Patient, Disease, Recipe }

DoctorId → Field of Entity

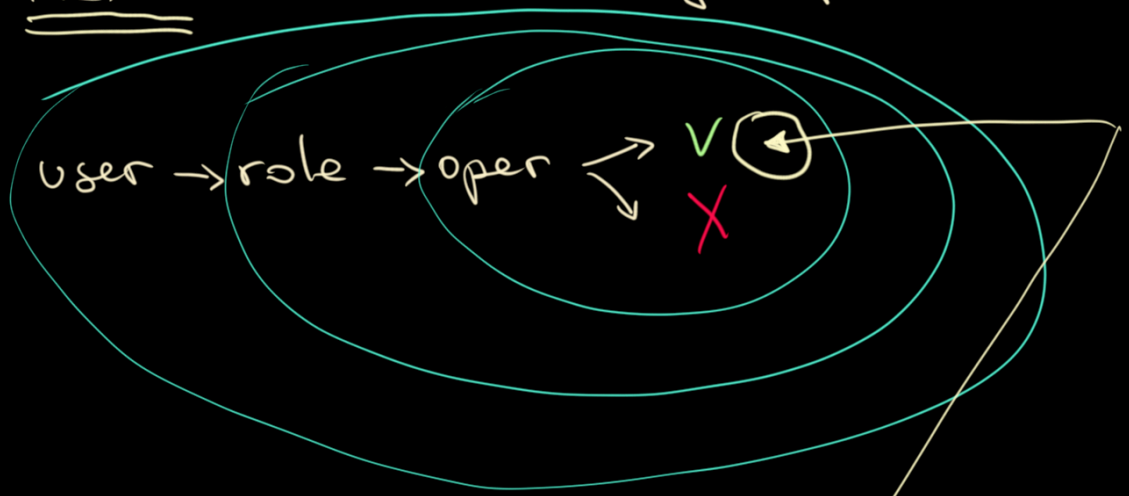
Is RBAC enough? NO

we need to have more grained access to the entities and to the fields/properties

DoctorId → Entity → Field

ABAC
Attribute-based

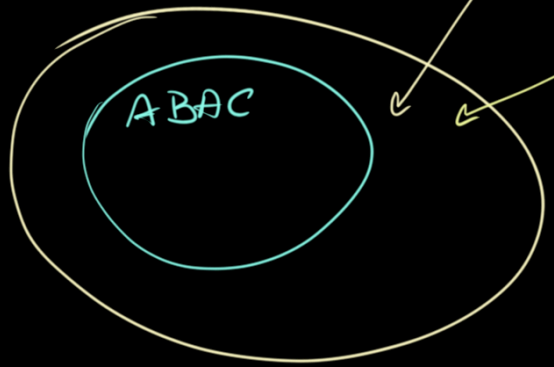
RBAC can be easily implemented as an Extra layer




business logic impl. has no idea about access control

ABAC - CAN NOT be implemented as an extra layer.

it's extremely coupled with implementation



## RBAC

User → Role → EntityType
Role → OperationType

more general (entity - relation based)

## ABAC

more precise (entity ID - based)

UserId → EntityId → FieldId
FieldName

## GDPR

general data protection regulation

← business responsibility

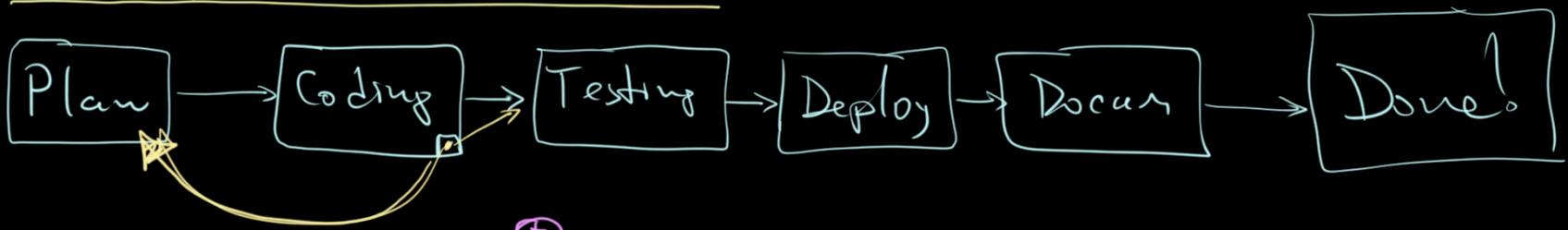
all the information stored in db needs to be encrypted

You need to encode ONLY data about Identity (First, Last Name, Birth, Id Code)

- more tables
- encode particular cols.

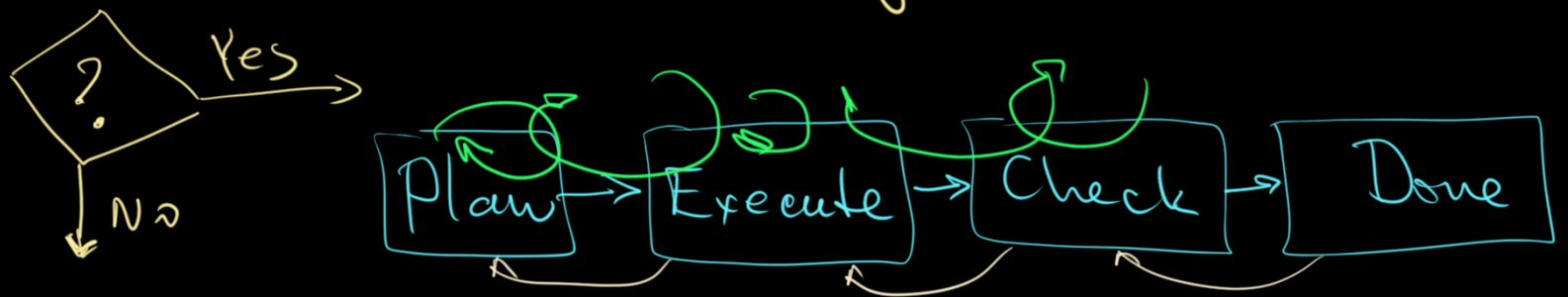
when we deal w/ GDPR we need to carefully design the data

# Development Methodologies:




Feedback ⊕ ⊖

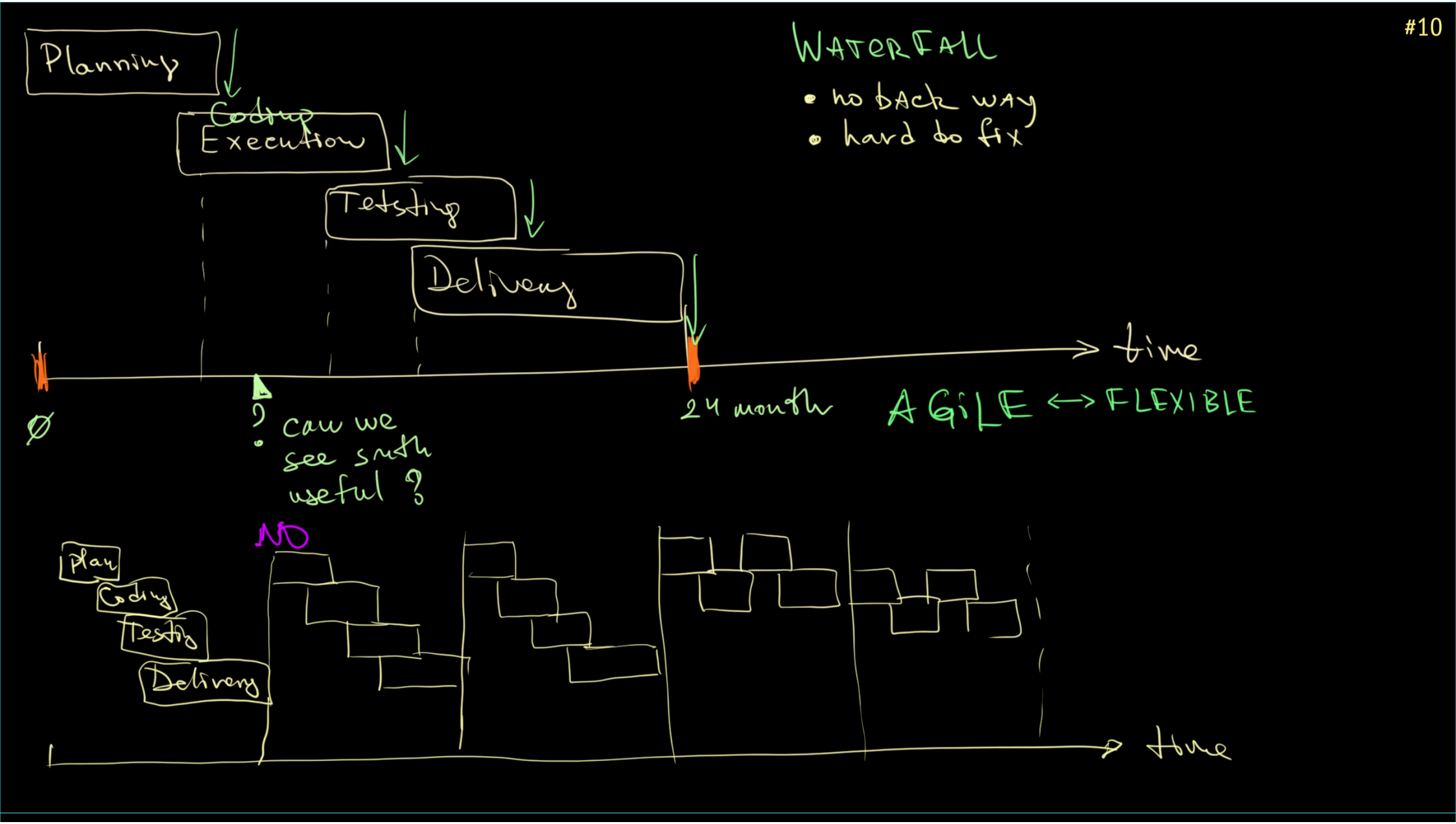
Any project fits that diagram




build house : almost no back links : ONE ITERATION

writing app : we have enormous back links : MANY ITERATIONS

but steps are the same


Planning
Coding
Execution
Testing
Delivery
time
∅
?
can we see smth useful ?
24 month
WATERFALL
• no BACK WAY
• hard to fix
AGILE ↔ FLEXIBLE
NO
Plan
Coding
Testing
Delivery
time

## Waterfall

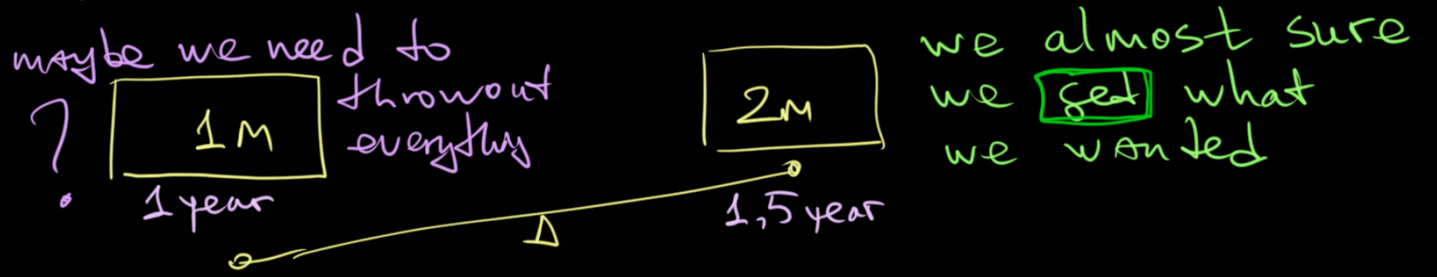
- ONE ITERATION
- WE CAN NOT SKIP ANY STAGE
- WE CAN NOT SEE INTERMEDIATE RESULTS
- FEEDBACK: IS TOO LONG TO WAIT
- ITERATION LENGTH — BIG

## AGILE

- MANY ITERATIONS
- WE CAN SKIP ANYTHING one realize smth went wrong
- WE CAN
- FEEDBACK is almost IMMEDIATE
- ITERATION LENGTH — SHORT

- TIME MATTERS
- we fix errors on early stages
- we understand faster if we are right

the cost is HIGHer



# DevOps

~~1 x PM~~
~~3 x Developers~~
~~1 x DevOps~~
~~1 x Test~~

→

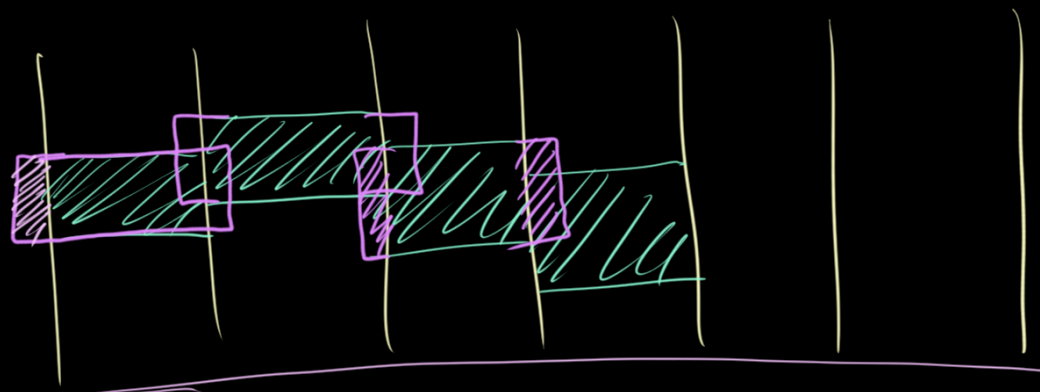
1 x PM
3 x Developers with DevOps skills
1 x Cloud Engineer / Automation Eng.
1 x Tester

DevOps - is not a position

DevOps Dev Admin

DevOps - extra SKILL

2017 (13) Kubernetes experience (8 years)
2014



DevOps - skill to understand and apply knowledge from ADJACENT AREAS