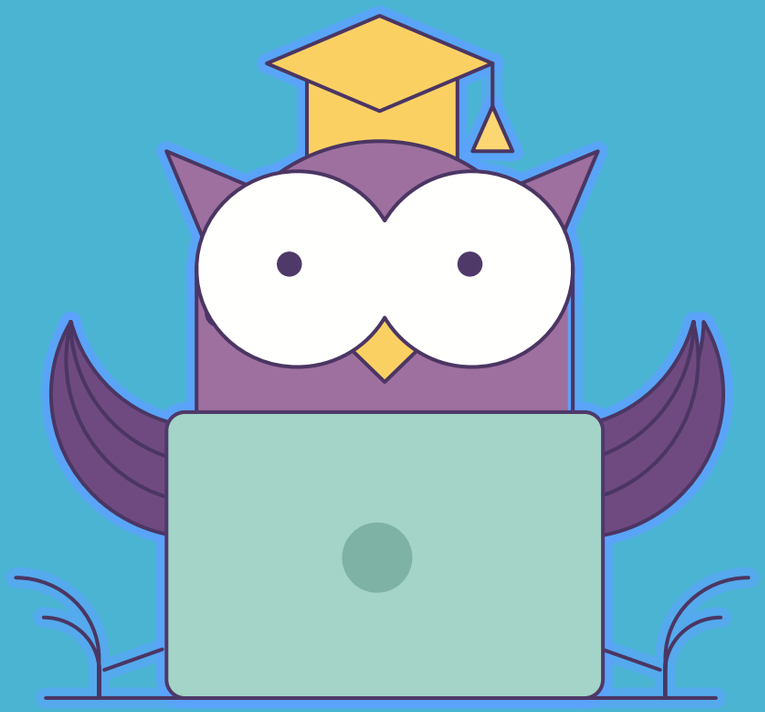

ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

# Как меня слышно и видно?



> Напишите в чат

**+** если все хорошо

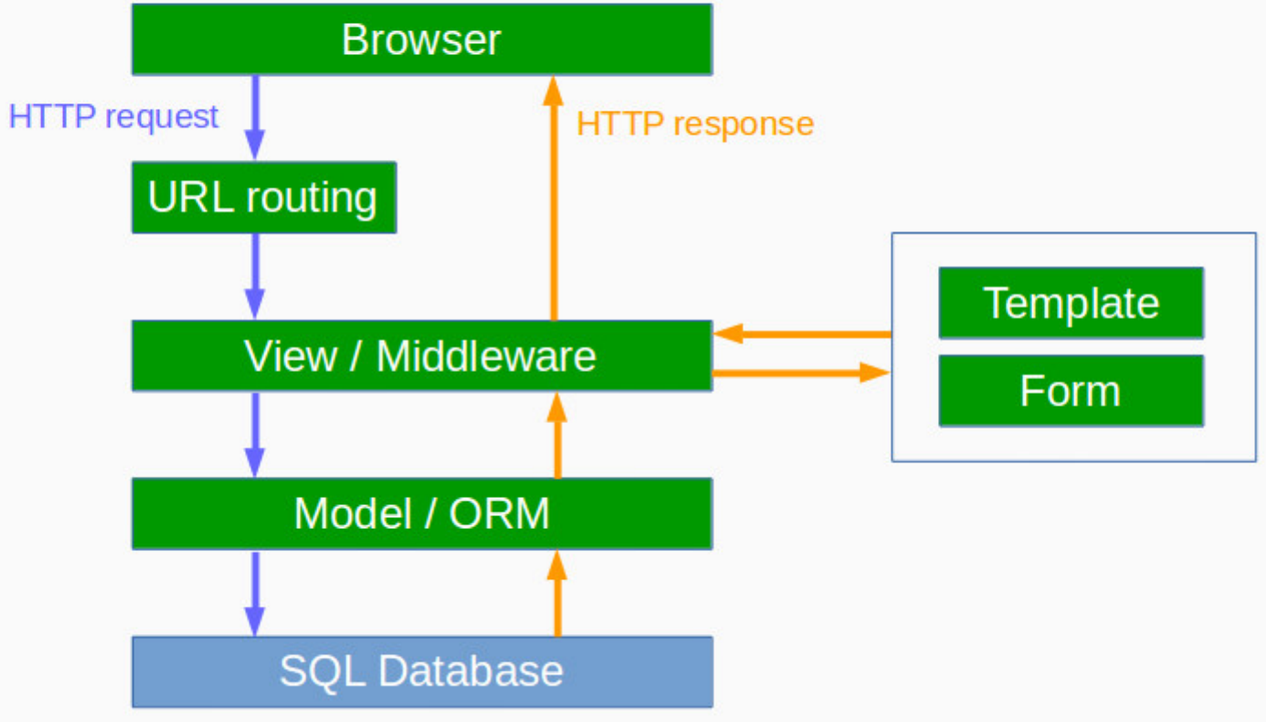
**-** если есть проблемы со звуком или с видео

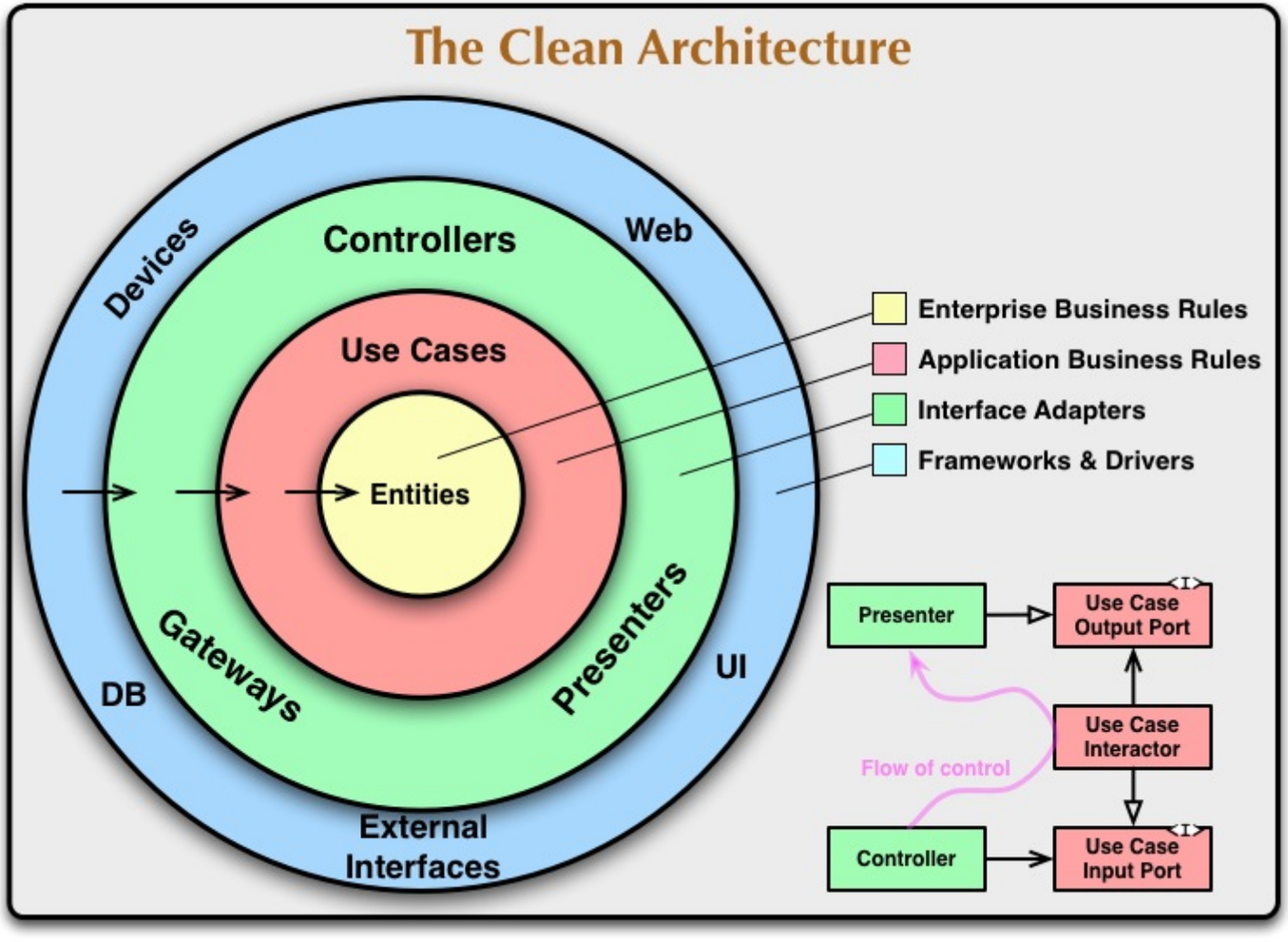
!проверить запись!

# Clean Architecture

Дмитрий Смаль

## Django's architecture




The Clean Architecture
Devices
Controllers
Web
Use Cases
Entities
Gateways
Presenters
DB
UI
External
Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Presenter
Use Case
Output Port
Use Case
Interactor
Flow of control
Controller
Use Case
Input Port

Ключевое правило: внутренние слои НЕ зависят от внешних

# Демо - проект

https://github.com/OtusTeam/Go/tree/master/cleancalendar

# Демо - проект

```
├── Makefile
├── api
│   └── api.proto
├── cmd
│   ├── grpc_client.go
│   ├── grpc_server.go
│   └── root.go
├── internal
│   ├── domain
│   │   ├── errors
│   │   │   └── error.go
│   │   ├── interfaces
│   │   │   └── event_storage.go
│   │   ├── models
│   │   │   └── event.go
│   │   └── services
│   │       └── event.go
│   ├── grpc
│   │   └── api
│   │       ├── api.pb.go
│   │       └── server.go
│   └── maindb
│       └── maindb.go
├── main.go
└── sql
```

# Слои проекта

## Devives, DB, UI..

- Postgres / драйвер pgx
- библиотека GRPC
- библиотека Cobra

## Gateways, Presenters, Controllers

- `cleancalendar/cmd/*.go`
- `internal/grpc/*.go`
- `internal/maindb/*.go`

## Entities, Usecases

- `internal/domain/models/*.go`
- `internal/domain/services/*.go`
- `internal/domain/interfaces/*.go`

# Go Standard Layout

https://github.com/golang-standards/project-layout

```
.
├── Makefile
├── api
├── assets
├── build
├── cmd
├── configs
├── deployments
├── docs
├── examples
├── githooks
├── init
├── internal
├── pkg
├── scripts
├── test
├── third_party
├── tools
├── vendor
├── web
└── website
```

- Обращение к `domain` слою через интерфейсы
- Dependcy Injection https://habr.com/ru/company/funcorp/blog/372199/

# Плюсы и минусы

Плюсы

- Чистая модель и бизнес-логика
- Возможность простого Unit-тестирования

Минусы

- Зачастую выглядит как overkill
- Есть шанс получить Anemic Model

Микросервисы ?

- Реализация бизнес-логики (валидация)
- Тестирование бизнес-логики
- Альтернативный интерфейс (REST/http)

Заполните пожалуйста опрос

https://otus.ru/polls/5138/

# Спасибо за внимание!