




# Golang Developer. Professional

[otus.ru](https://otus.ru)

 Проверить, идет ли запись

# Меня хорошо видно && слышно?

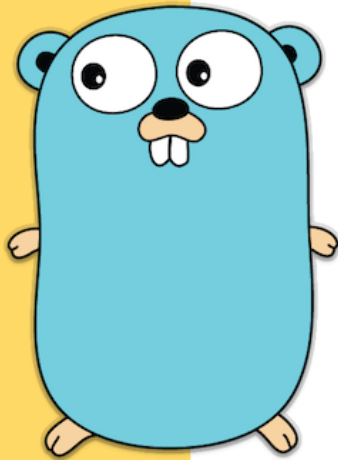
 Ставим “+”, если все хорошо  
“-”, если есть проблемы

Тема вебинара

# Работа с gRPC ч.2

**Дмитрий Копылов**

Backend Developer at OZON



# Правила вебинара



Активно  
участвуем



Off-topic обсуждаем  
в учебной группе



Задаем вопрос  
в чат или голосом



Вопросы вижу в чате,  
могу ответить не сразу

## Условные обозначения



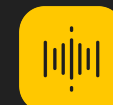
Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# О чем будем говорить

- Создание gRPC клиента и сервера
- Лучшие практики gRPC
- Интерсепторы

## Что такое интерсепторы (interceptors ~ middleware):

Это инструменты для перехвата и потенциального изменения запросов и ответов, проходящих между клиентами и серверами. Это особенно полезно для реализации поперечной (cross-cutting) функциональности, такой как аутентификация, логирование, мониторинг, распределение транзакций и обработка ошибок.

# Клиентские интерсепторы:

- Перехватывают входящие и исходящие вызовы на стороне клиента.
- Могут модифицировать запрос перед тем, как он будет отправлен на сервер.
- Могут обрабатывать или изменять ответ от сервера перед тем, как он будет возвращён вызывающей стороне.

# Серверные интерсепторы:

- Перехватывают входящие вызовы на стороне сервера до того, как они достигнут сервиса.
- Могут изменять поведение вызова, например, добавляя проверку аутентификации или проверку прав доступа, прежде чем запрос будет обработан.
- Могут вмешиваться в процесс ответа сервера, позволяя добавлять дополнительные заголовки или изменять ответ перед отправкой клиенту.



# Реализация интерсепторов:

В gRPC интерсепторы реализуются через определённые хуки или методы, которые вызываются в процессе обработки RPC вызовов. Например, в Go вы можете реализовать интерсептор на стороне сервера, предоставив функцию, которая будет принимать `context.Context`, информацию о RPC и обработчик, который непосредственно выполняет RPC вызов.

# Пример реализации интерсептора на Go

```
// UnaryServerInterceptor пример функции интерсептора
func UnaryServerInterceptor(ctx context.Context, req interface{}, info *grpc.UnaryServerInfo, handler grpc.Handler) (interface{}, error) {
    // До вызова обработчика
    log.Println("Before handler")
    if err := authorize(ctx); err != nil {
        return nil, err
    }

    // Вызов обработчика
    resp, err := handler(ctx, req)

    // После вызова обработчика
    log.Println("After handler")

    return resp, err
}

// Настройка сервера с интерсептором
server := grpc.NewServer(grpc.UnaryInterceptor(UnaryServerInterceptor))
```

# Прежде, чем начать

1) Устанавливаем protoc

<https://grpc.io/docs/protoc-installation/>

2) Обновляем `protoc-gen-go` и `protoc-gen-go-grpc`

```
go get -u google.golang.org/protobuf/cmd/protoc-gen-go
go get -u google.golang.org/grpc/cmd/protoc-gen-go-grpc
```

# Создание gRPC клиента и сервера

## От теории к практике

# Streaming

- Unary RPC - один запрос и один ответ
- Server streaming RPC - один запрос и много ответов
- Client streaming RPC - много запросов и один ответ
- Bidirectional streaming RPC - много запросов и много ответов

# Когда использовать streaming?

- Когда нужно передать большой объем данных (сервер может регулярно отправлять клиенту промежуточные результаты для информирования о прогрессе выполнения)
- Когда нужно передать данные в реальном времени (например, чаты, онлайн-игры)
- Подписка на события (Publish/Subscribe)
- Поточковая передача файлов (например, видео)
- Обработка данных на лету (например, обработка аудио/видео)

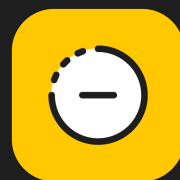
# Итоги

- Поговорили о лучших практиках gRPC
- Узнали, что такое Интерсепторы
- Разобрались с streaming
- Создали несколько gRPC клиентов и серверов

# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет





**Заполните, пожалуйста,  
опрос о занятии  
по ссылке в чате**

# Следующий вебинар

23 мая

Монолит и микросервисы



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой



Спасибо за внимание!

# Приходите на следующие вебинары

Дмитрий Копылов

Backend Developer at OZON

