

Что я не люблю в  
ваших open source  
проектах на Go


@quasilyte



Я ♥ Go

Я ♥ Open Source

Обсудим то, что делает ваш  
проект более привлекательным.



# Стабильный master

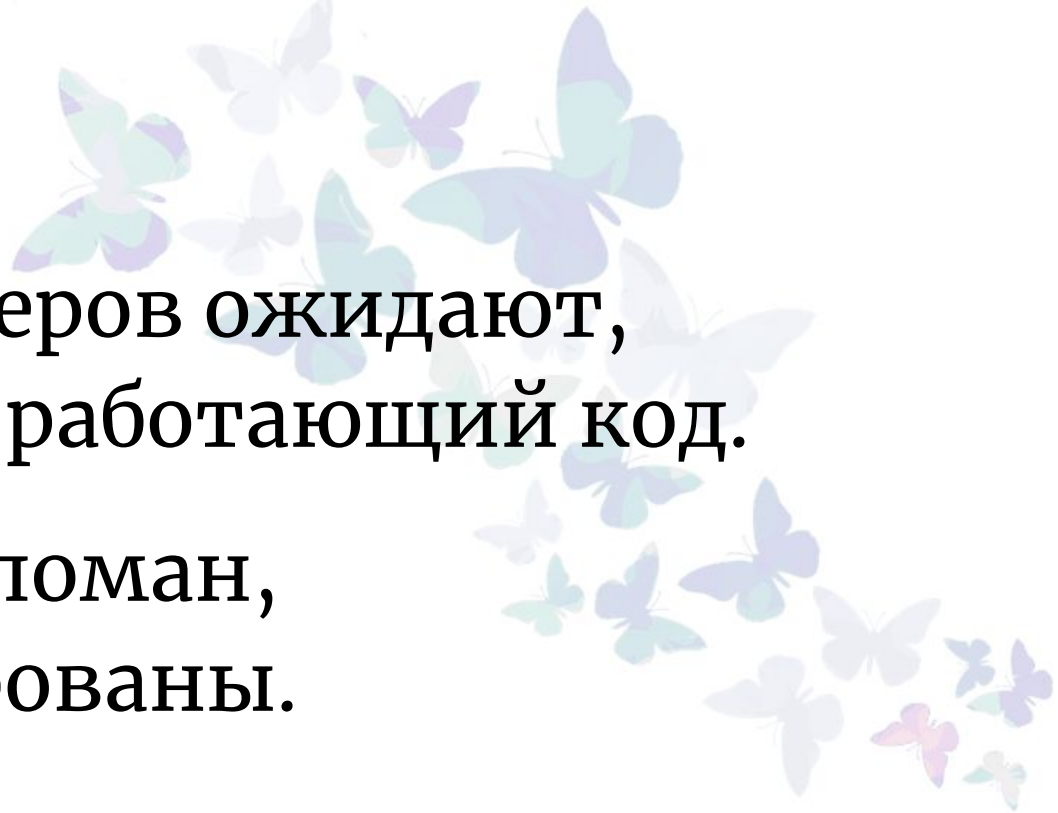
Ветка master содержит код,  
который проходит тесты и  
собирается хотя бы в базовой  
конфигурации.



# Go get

Большинство гоферов ожидают,  
что go get скачает работающий код.

Если ваш master сломан,  
они будут разочарованы.



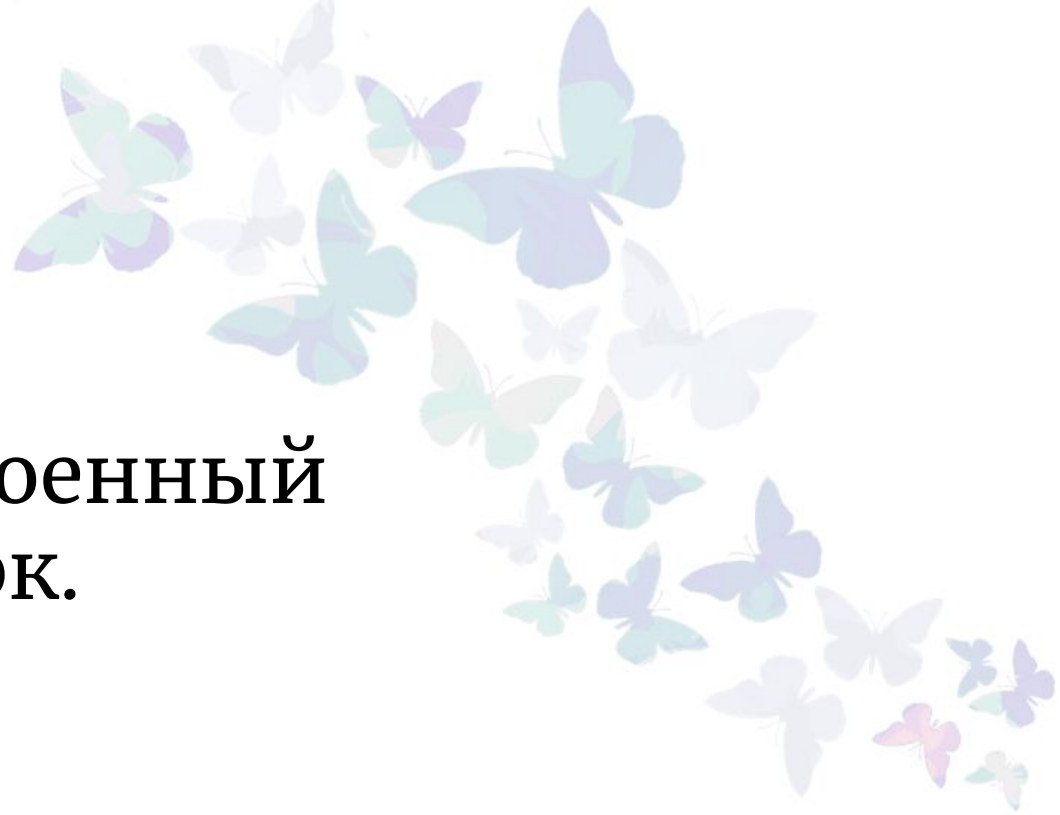
# Соблюдайте общие стандарты

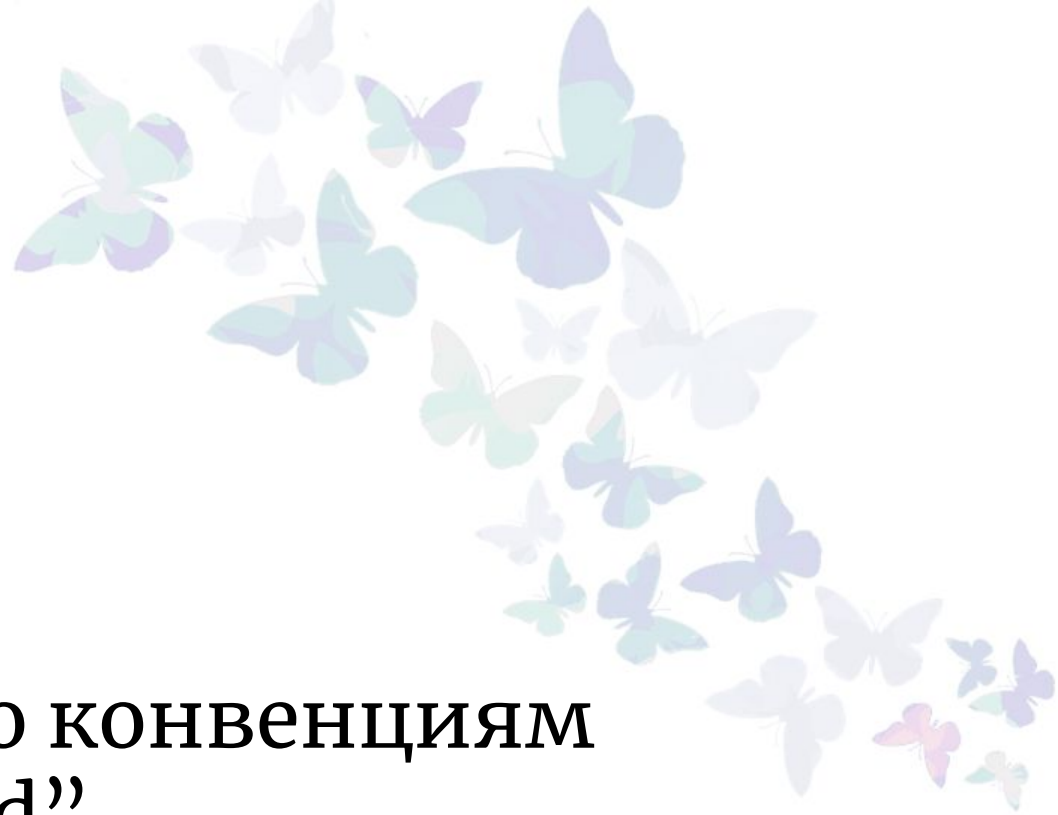
Как минимум ваш код должен  
удовлетворять `go lint+vet`  
и быть отформатирован с  
помощью `gofmt`.



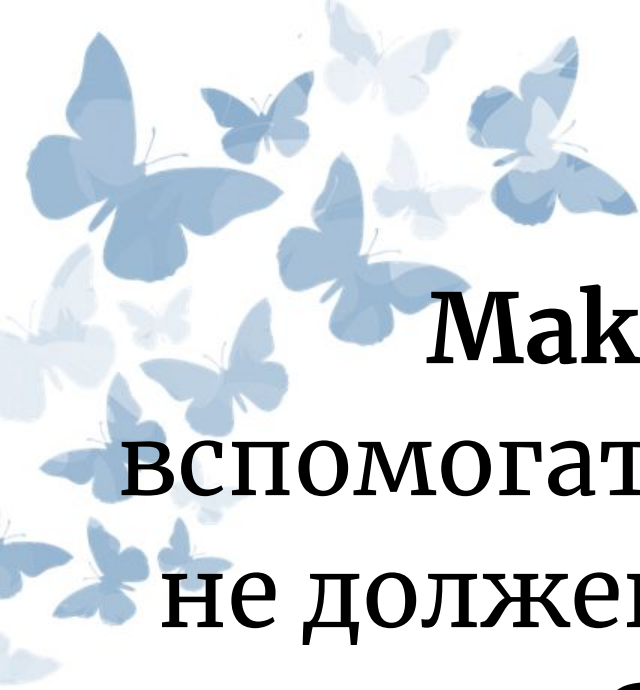
Тесты через встроенный  
testing фреймворк.

Запуск тестов по  
“go test” или “make test”.





Layout проекта по конвенциям  
утилиты “go build”.



**Makefile** может быть  
вспомогательным средством, но  
не должен быть необходим для  
сборки проекта.





# Идиоматичный стиль кода

В интернете очень много полезных статей об идиоматичном Go.

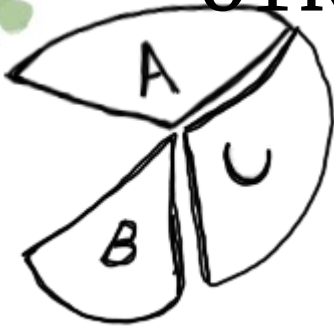
[golang.org/wiki/CodeReviewComments](https://golang.org/wiki/CodeReviewComments)

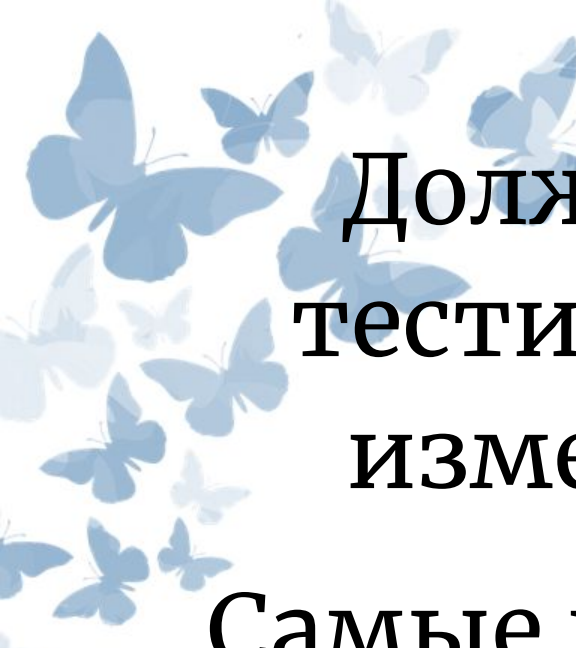
[golang.org/doc/effective\\_go.html](https://golang.org/doc/effective_go.html)

[dmitri.shuralyov.com/idiomatic-go](https://dmitri.shuralyov.com/idiomatic-go)

# Гранулярные тесты


Тесты, которые требуют множества зависимостей  
опциональны или  
отключаемы при запуске через  
`test.short=true`.





Должно быть возможно  
тестировать релевантные  
изменения в изоляции.

Самые нетривиальные тесты  
можно вынести в  
CI (Continuous Integration).



# Contributing guide

В проекте есть документ,  
описывающий как лучше  
всего начать своё путешествие  
по вашему проекту.



# Рекомендации

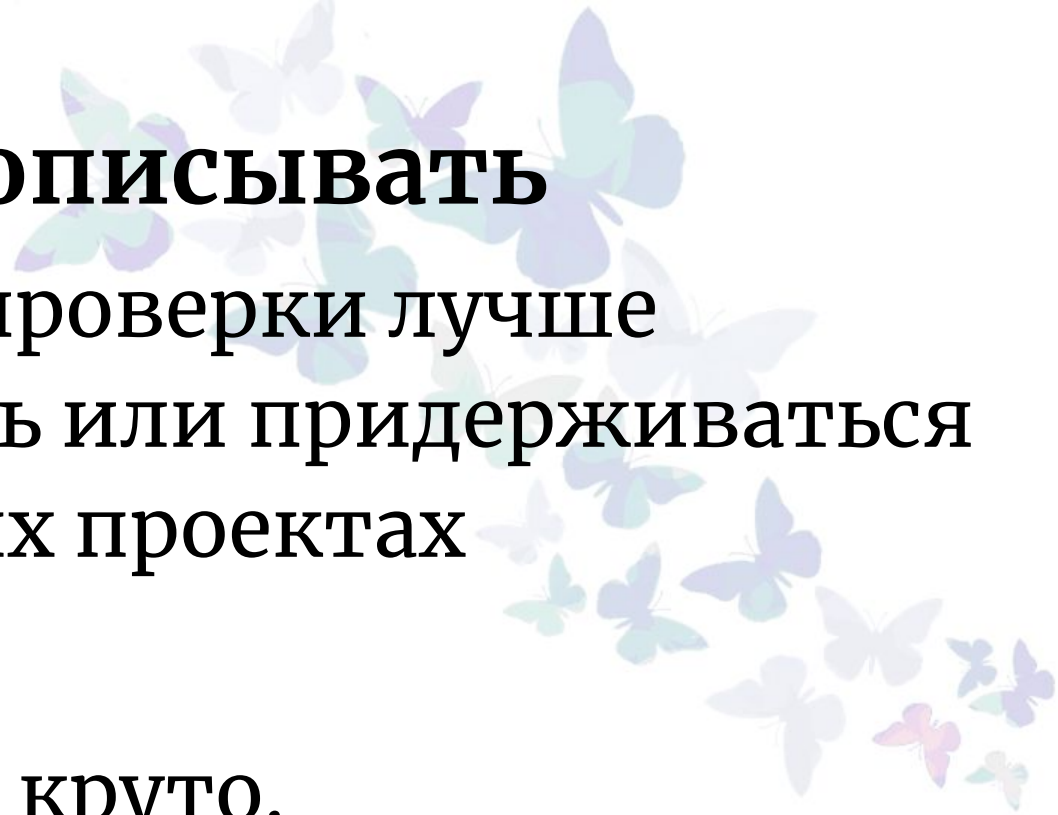
- Описание правил ревью
- Список разрешений и запретов
- Специфика работы с git (workflow)

Дополнительную информацию, вроде FAQ или подсказок, лучше вынести в отдельные документы.

**Что не нужно описывать**

Стилистические проверки лучше  
автоматизировать или придерживаться  
принятых в других проектах  
стандартов.


Быть как все – это круто.



# Открытость процесса


Отказ от принятия изменений должен быть обоснован либо на месте, либо в документе CONTRIBUTING.md.

В идеале, в open source проекте нет “скрытых знаний”, которыми не делятся со “внешними” участниками.



Но не ожидайте,  
что каждый человек будет читать  
ваш CONTRIBUTING.md.

Чаще всего на него просто удобно  
ссылаться на ревью.



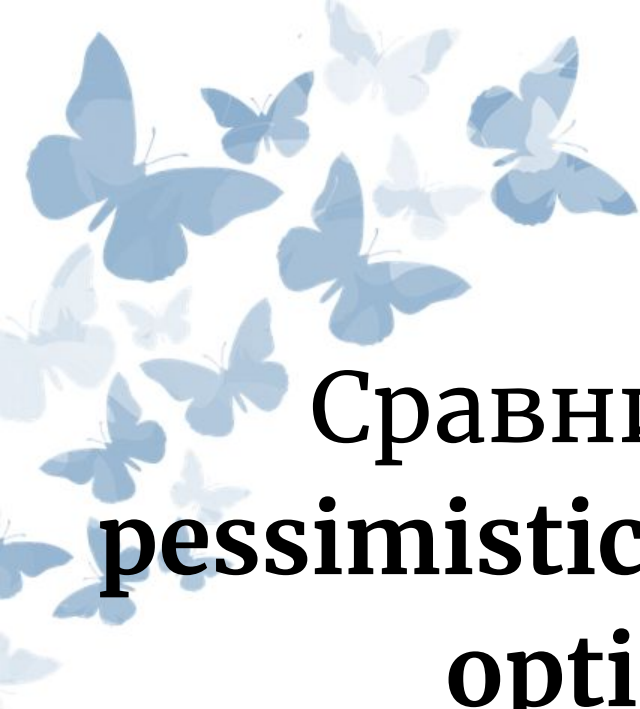


# Позитивное ревью

Процесс ревью PR быстр,  
корректные изменения  
принимаются без замедлений.

Определение корректного  
изменения нужно описать в  
CONTRIBUTING.md.






Сравним традиционную,  
**pessimistic** схему с предлагаемым  
**optimistic** аналогом.



# zeromq C4

Лучше всего выбрать готовую модель для проведения публичного ревью и адаптировать её под свой проект (либо использовать без изменений).

[rfc.zeromq.org/spec:42/C4/](https://rfc.zeromq.org/spec:42/C4/)



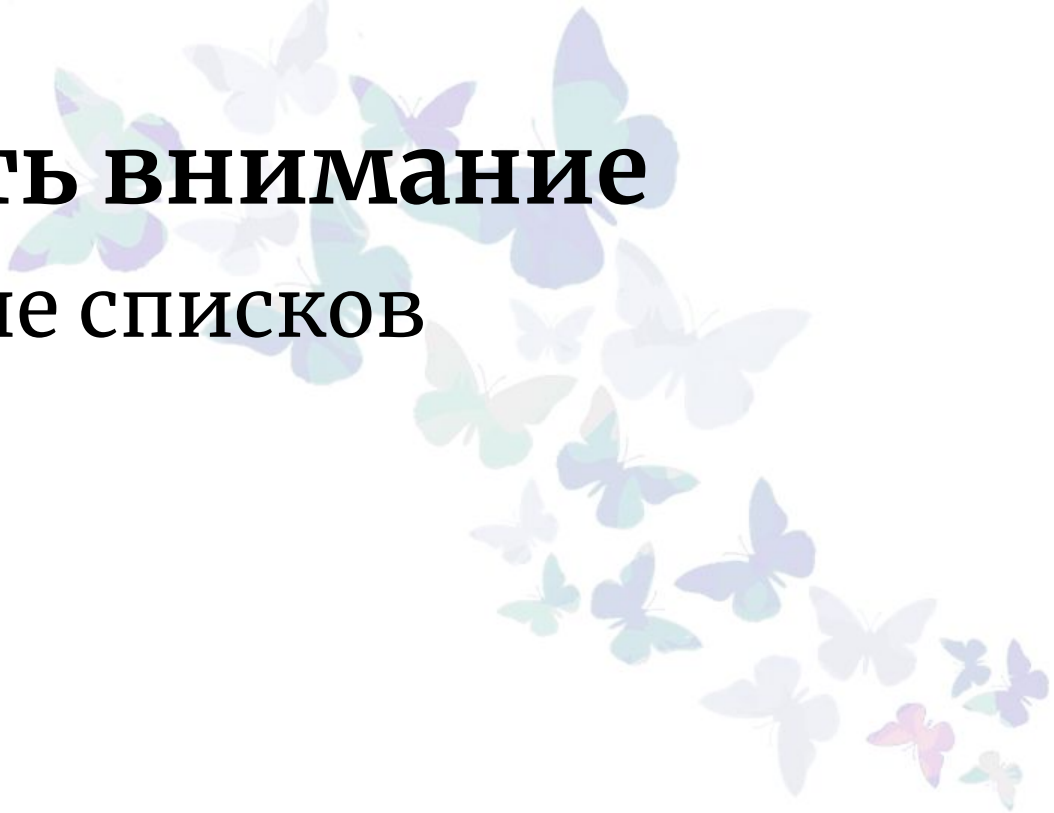
**Здоровый godoc**  
godoc документация  
библиотеки отображается  
ожидаемым образом.



godoc reference

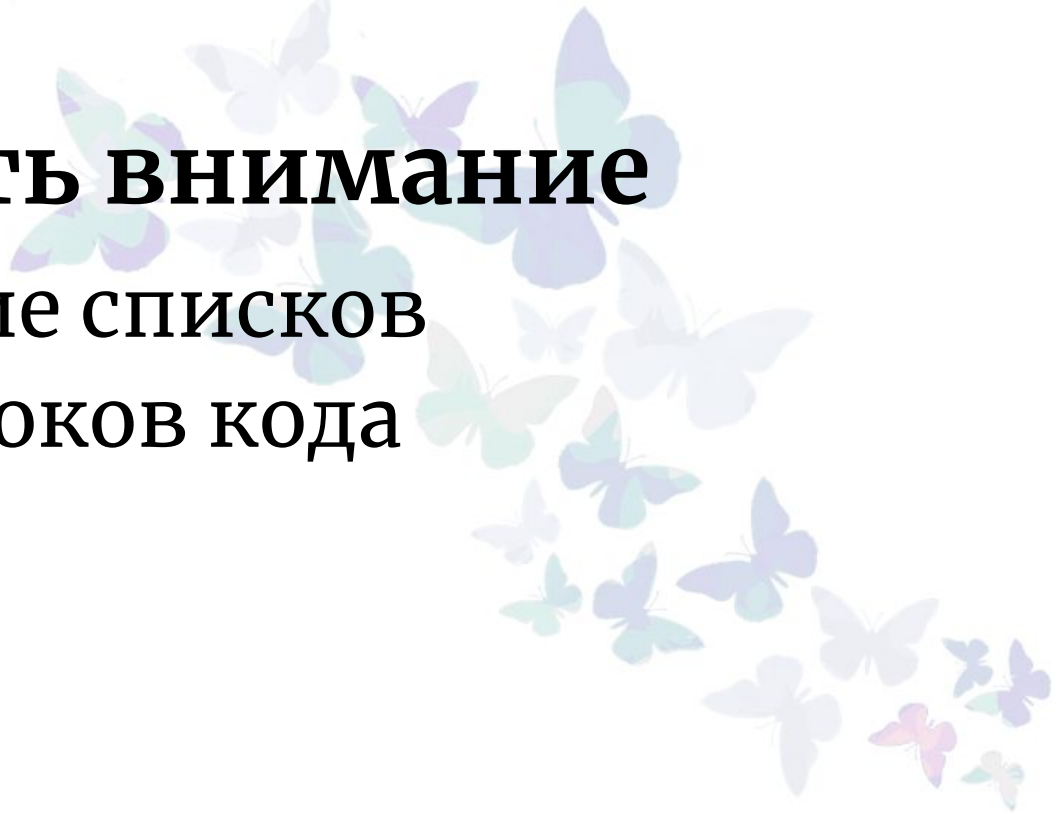
# На что обратить внимание

- Форматирование списков



# На что обратить внимание

- Форматирование списков
- Оформление блоков кода



# На что обратить внимание

- Форматирование списков
- Оформление блоков кода
- Попали ли примеры в документацию

# На что обратить внимание

- Форматирование списков
- Оформление блоков кода
- Попали ли примеры в документацию
- Особые метки типа “Deprecated:”



# Всем множества



# на гитхабе

(Fin.)

