Продуктовая веб-разработка на Go

Максим Рындин | 17 марта 2018

Gett


GETT

GETT

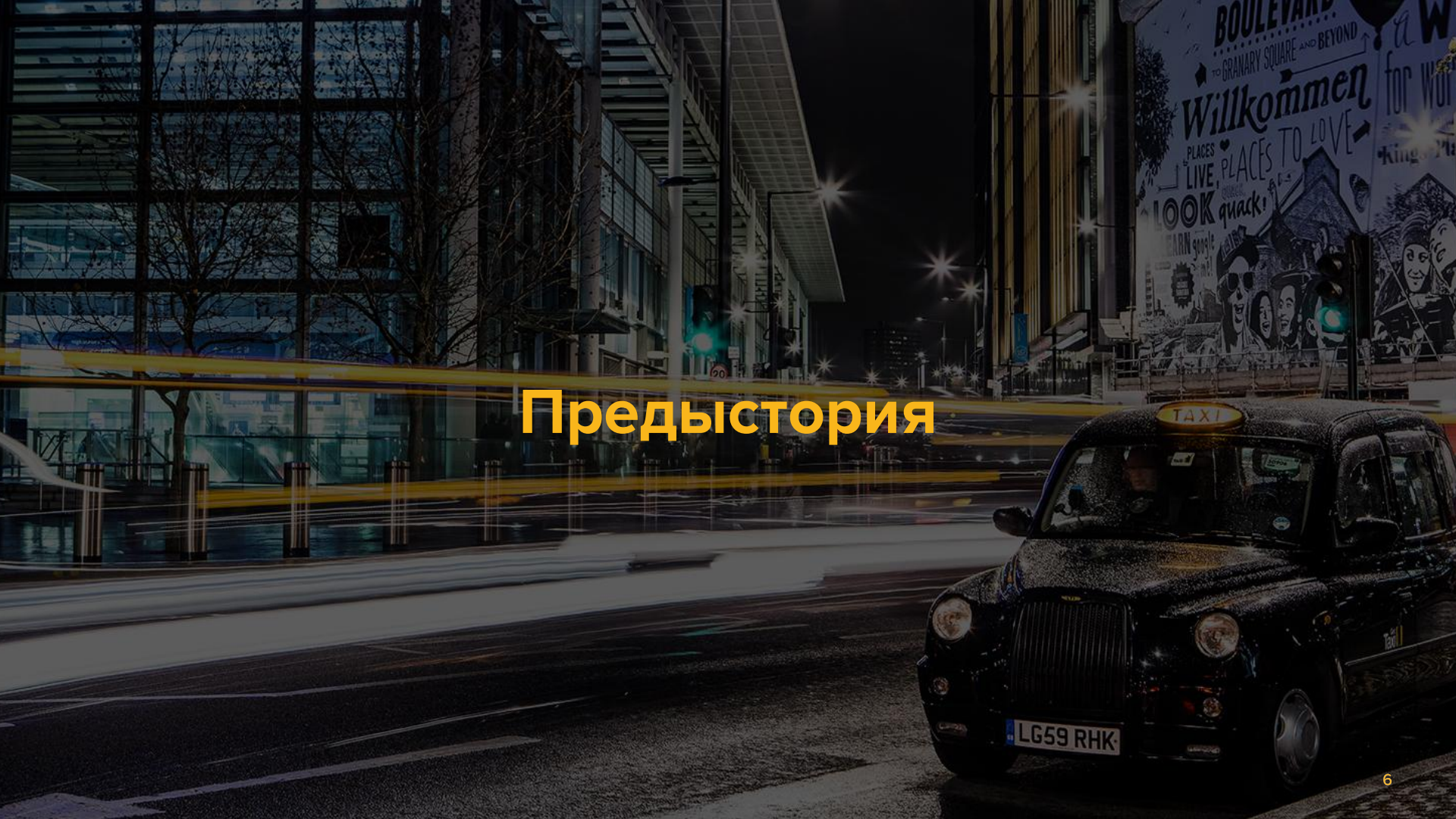
1. международный сервис заказа такси, основанный в 2011

GETT

1. международный сервис заказа такси, основанный в 2011
2. основные продукты: мобильные приложения, веб-портал B2B

GETT

1. международный сервис заказа такси, основанный в 2011
2. основные продукты: мобильные приложения, веб-портал B2B
3. в начале 2017 года открылся офис RnD в Москве



Предыстория

Предыстория

1. монолит на Ruby on Rails

Предыстория

1. монолит на Ruby on Rails
2. проблемы при росте нагрузок

Предыстория

1. монолит на Ruby on Rails
2. проблемы при росте нагрузок
3. вынос функциональности сбора координат в отдельный сервис, node.js

Предыстория

1. монолит на Ruby on Rails
2. проблемы при росте нагрузок
3. вынос функциональности сбора координат в отдельный сервис, node.js
4. новый рост нагрузок и новые проблемы

	Node.js	Go
Transactions, hits	74522	237597
Availability, %	100	100
Elapsed time, sec	599.56	599.34
Data transferred, MB	71.14	312.01
Response time, sec	0.8	0.25
Transaction rate, trans/sec	124.29	396.43
Throughput, MB/sec	0.12	0.52
Concurrency	99.44	99.22
Successful transactions	74522	237597
Failed transactions	0	0
Longest transaction	1.14	0.75
Shortest transaction	0.38	0.2

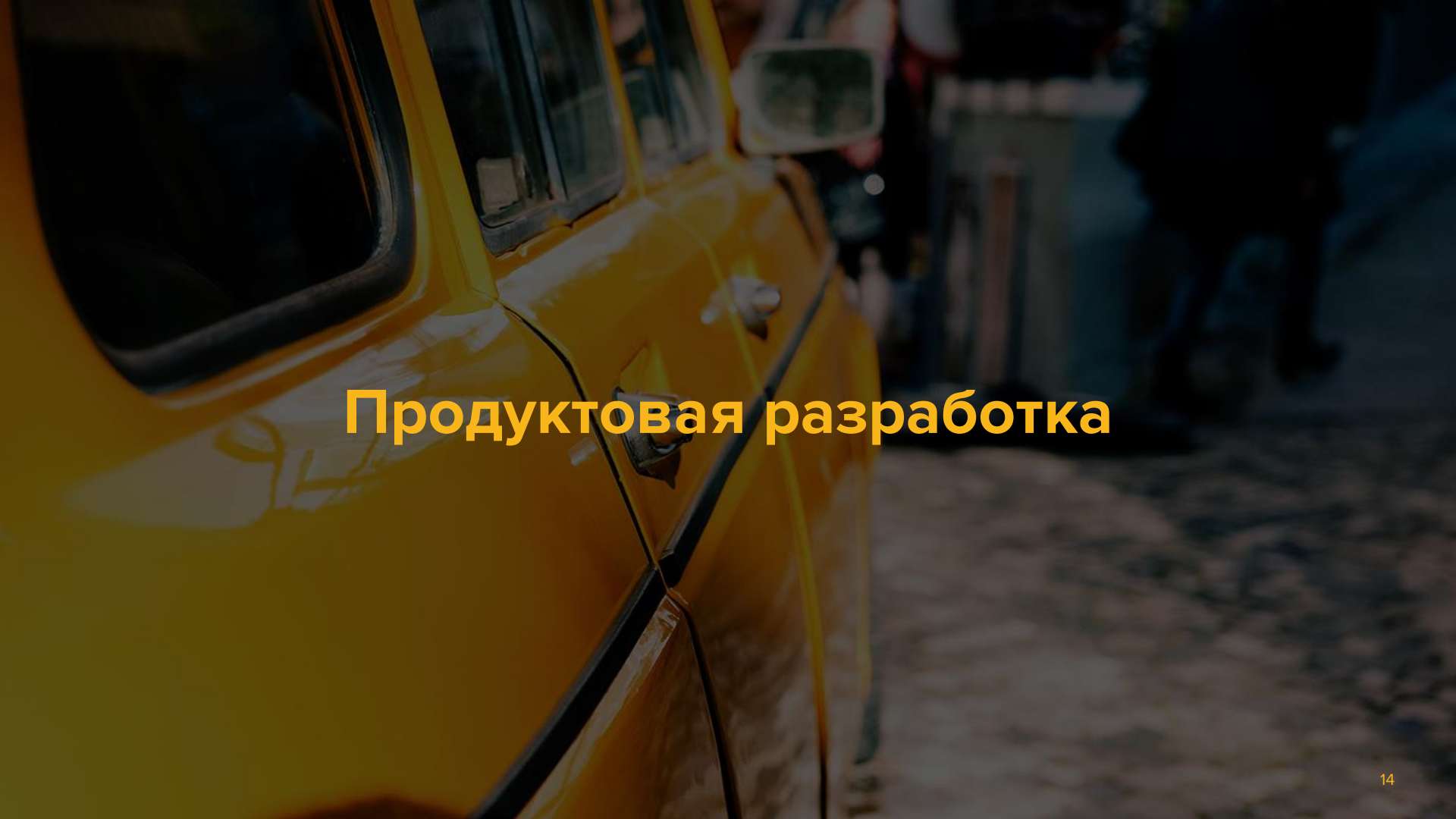
Gett

Предыстория

1. монолит на Ruby on Rails
2. проблемы при росте нагрузок
3. вынос функциональности сбора координат в отдельный сервис, node.js
4. новый рост нагрузок и новые проблемы
5. SOA, много ввода/вывода

Предыстория

1. монолит на Ruby on Rails
2. проблемы при росте нагрузок
3. вынос функциональности сбора координат в отдельный сервис, node.js
4. новый рост нагрузок и новые проблемы
5. SOA, много ввода/вывода
6. GO!



Продуктовая разработка

Продуктовая разработка

1. разные требования

Продуктовая разработка

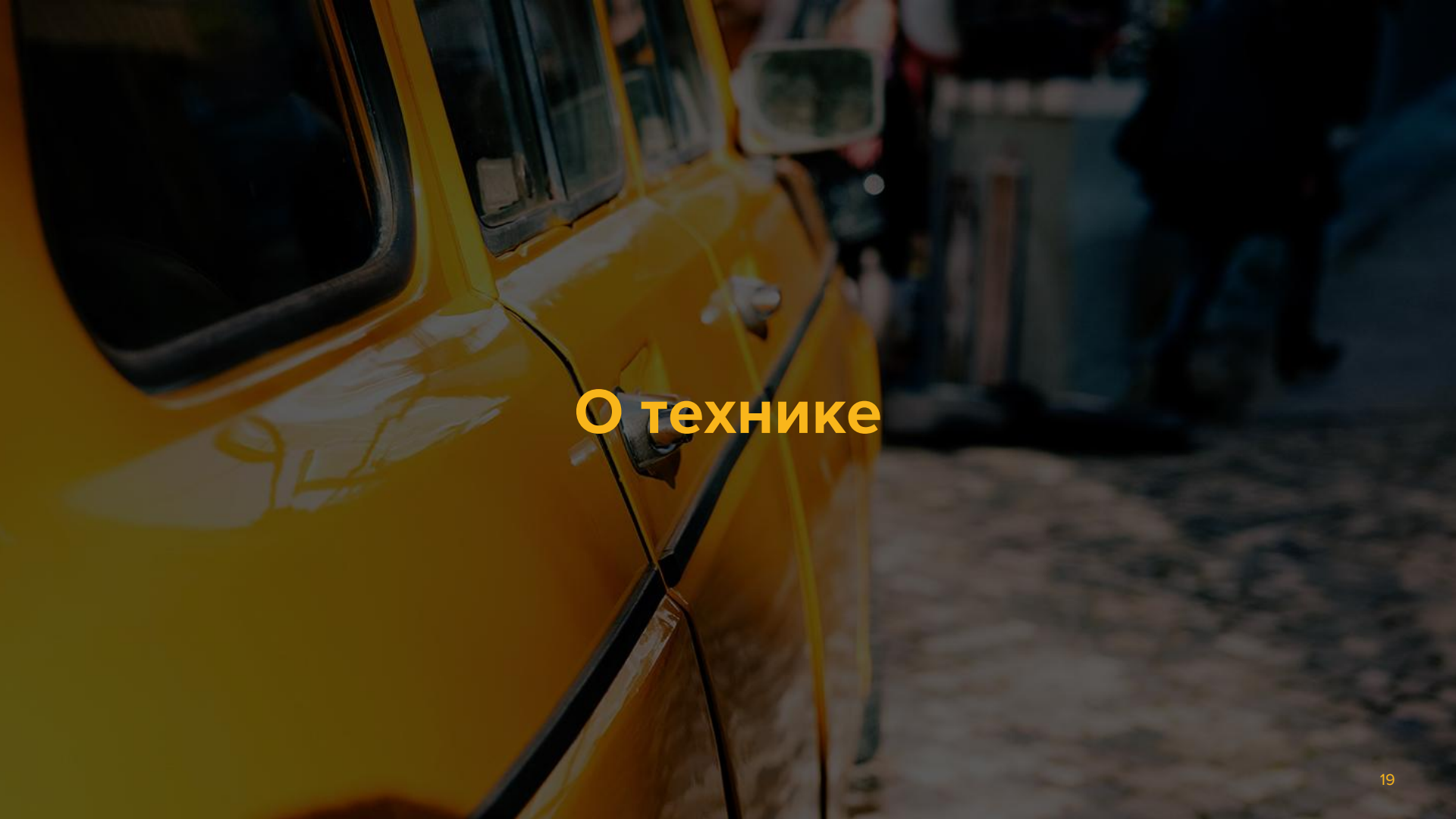
1. разные требования
2. постановка требований вокруг пользовательских историй

Продуктовая разработка

1. разные требования
2. постановка требований вокруг пользовательских историй
3. тестирование

Продуктовая разработка

1. разные требования
2. постановка требований вокруг пользовательских историй
3. тестирование
4. желание не дублировать работу



О технике

Веб фреймворк

1. MVC

Веб фреймворк

1. MVC
2. Beego, https://github.com/astaxie/beego/

Веб фреймворк

1. MVC
2. Beego, https://github.com/astaxie/beego/
3. MVC => MC

Веб фреймворк

1. MVC
2. Beego, https://github.com/astaxie/beego/
3. MVC => MC
4. автосборка

Веб фреймворк

1. MVC
2. Beego, https://github.com/astaxie/beego/
3. MVC => MC
4. автосборка
5. маршрутизация по комментариям

Контроль базы данных

1. контроль схемы базы данных на уровне кода

Контроль базы данных

1. контроль схемы базы данных на уровне кода
2. Swan, https://bitbucket.org/liamstask/goose/

Контроль базы данных

1. контроль схемы базы данных на уровне кода
2. Swan, https://bitbucket.org/liamstask/goose/
3. предупреждения об опасных миграциях

Контроль базы данных

1. контроль схемы базы данных на уровне кода
2. Swan, https://bitbucket.org/liamstask/goose/
3. предупреждения об опасных миграциях
4. построение индексов с параметром CONCURRENTLY

Всегда можно сделать так

```
COMMIT;
CREATE INDEX CONCURRENTLY huge_index ON huge_table (column_one, column_two);
BEGIN;
```

Тестирование

1. Ginkgo, https://github.com/onsi/ginkgo/

Тестирование

1. Ginkgo, https://github.com/onsi/ginkgo/
2. начальные условия

NewRelic

1. анализ времени обработки запроса

NewRelic

1. анализ времени обработки запроса
2. выявление медленных запросов в БД и внешних вызовов

NewRelic

1. анализ времени обработки запроса
2. выявление медленных запросов в БД и внешних вызовов
3. отслеживание потоков данных

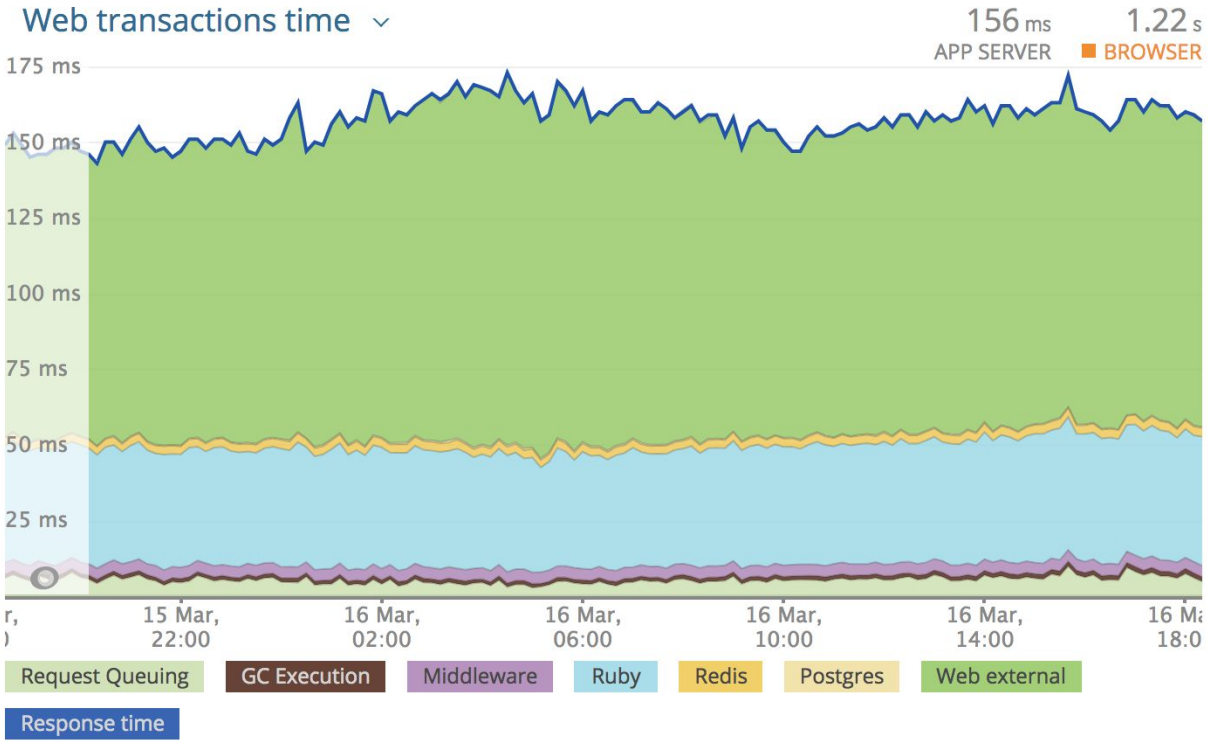
NewRelic

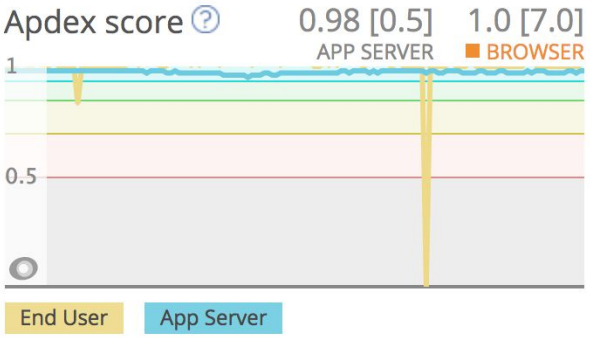
1. анализ времени обработки запроса
2. выявление медленных запросов в БД и внешних вызовов
3. отслеживание потоков данных
4. анализ ошибок и предупреждения

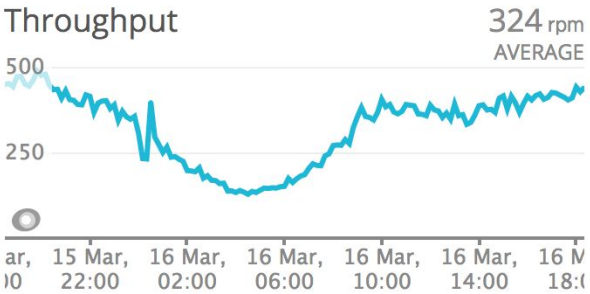
NewRelic

1. анализ времени обработки запроса
2. выявление медленных запросов в БД и внешних вызовов
3. отслеживание потоков данных
4. анализ ошибок и предупреждения
5. https://github.com/newrelic/go-agent/

NewRelic

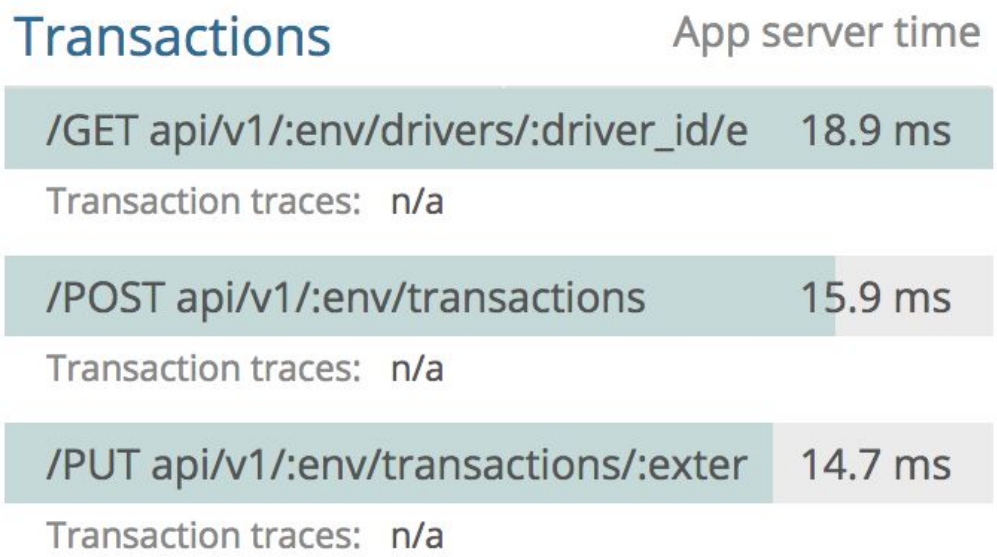

Web transactions time
156 ms
APP SERVER
1.22 s
BROWSER
175 ms
150 ms
125 ms
100 ms
75 ms
50 ms
25 ms
15 Mar, 22:00
16 Mar, 02:00
16 Mar, 06:00
16 Mar, 10:00
16 Mar, 14:00
Request Queuing
GC Execution
Middleware
Ruby
Redis
Postgres
Web external
Response time


Apdex score
0.98 [0.5]
APP SERVER
1.0 [7.0]
BROWSER
1
0.5
End User
App Server


Throughput
324 rpm
AVERAGE
500
250
15 Mar, 22:00
16 Mar, 02:00
16 Mar, 06:00
16 Mar, 10:00
16 Mar, 14:00

Фильтры Beego для отслеживания запросов:

```
beego.InsertFilter("*", beego.BeforeRouter, StartTransaction, false)
beego.InsertFilter("*", beego.AfterExec, NameTransaction, false)
beego.InsertFilter("*", beego.FinishRouter, EndTransaction, false)
```



Обратные вызовы запросов в БД:

```
db.Callback().Create().Before("gorm:begin_transaction").
    Register("newrelicStart", startSegment)
db.Callback().Create().After("gorm:commit_or_rollback_transaction").
    Register("newrelicStop", endSegment)
```

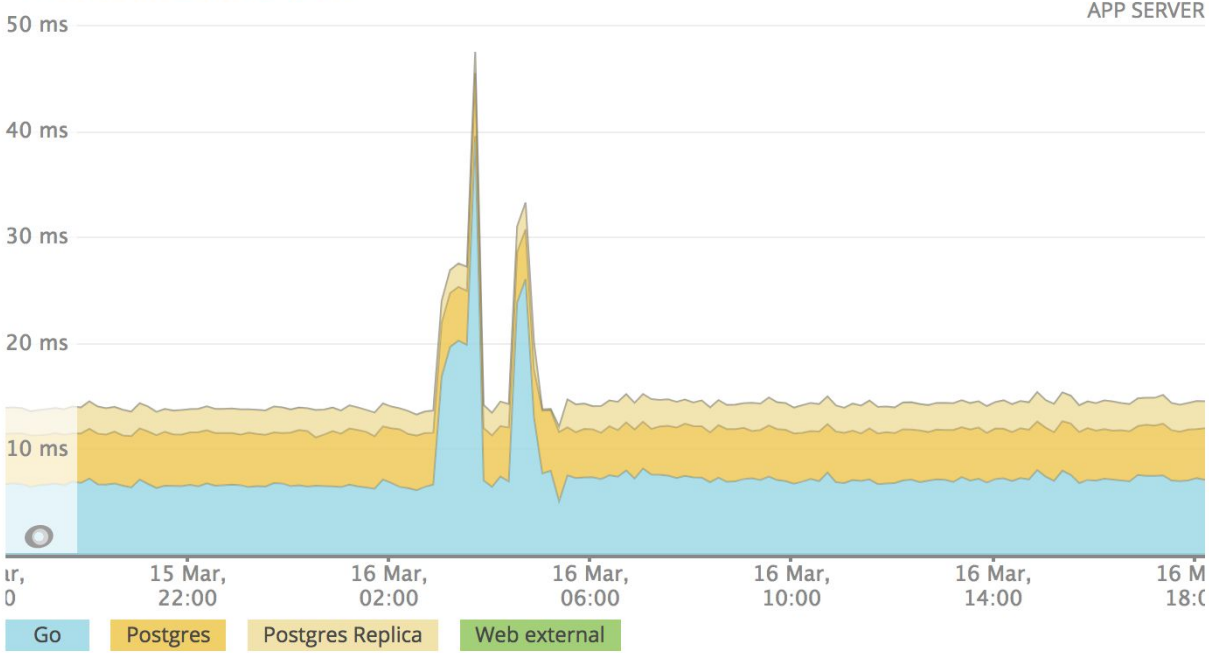
NewRelic

1. обертка над redis.Client
2. обертка над http.Client

NewRelic

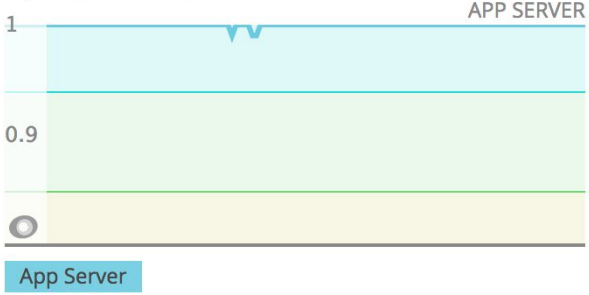
Web transactions time

14.4 ms
APP SERVER



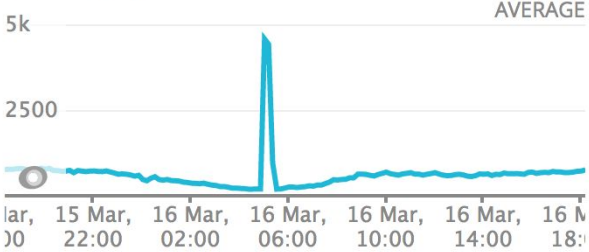
Apdex score

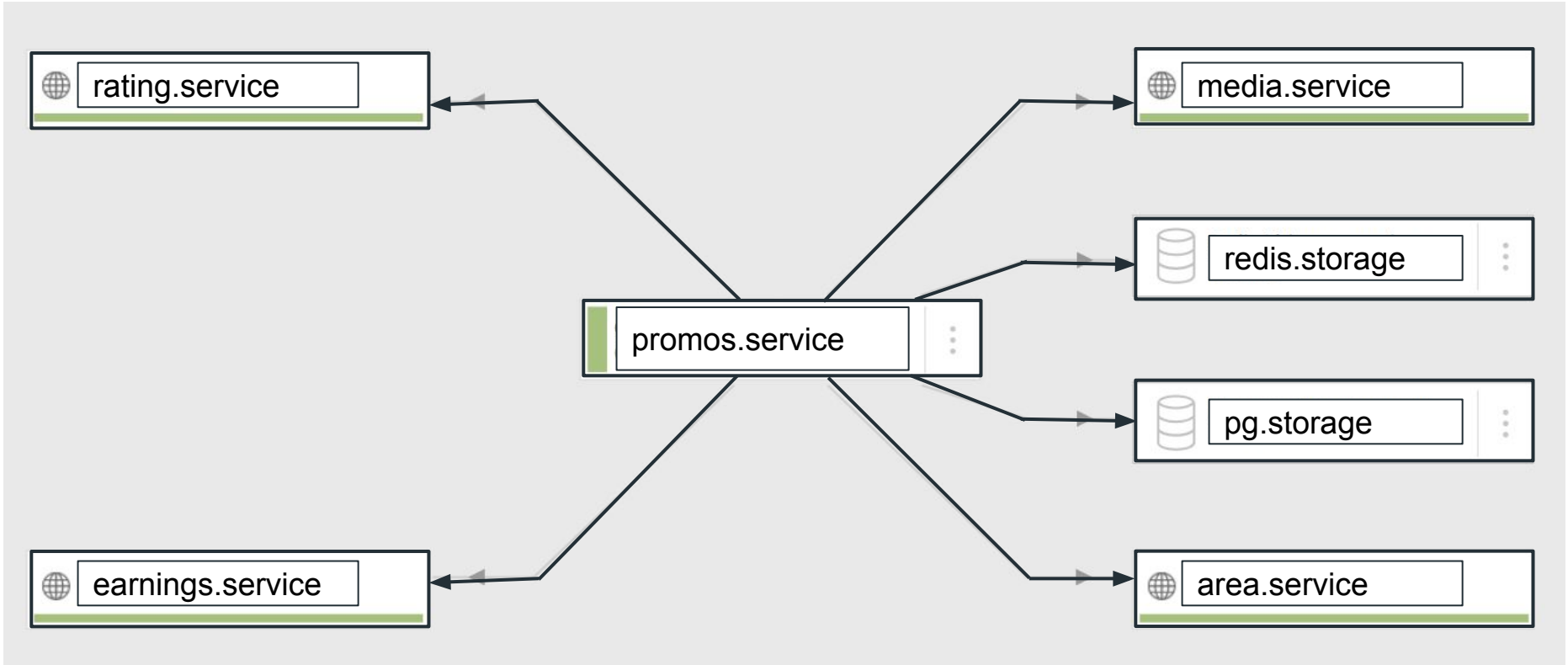
1.0 [0.5]
APP SERVER



Throughput

612 rpm
AVERAGE




rating.service
media.service
redis.storage
promos.service
pg.storage
earnings.service
area.service

Спасибо