Uma Black Friday sem catástrofes

Jéssica Bonson
Principal Engineer no Olist

TDC Floripa 2020, Trilha Microservices

olist

Jéssica Pauli de C Bonson

- +-8 anos de exp em pesquisa/desenvolvimento
- graduação/mestrado em Ciências da Computação
- foco em dev backend, machine learning e big data

Hobbies:

Jogos
RPG
Canto

olist

Maior loja nos principais marketplaces do Brasil.

Arquitetura em microsserviços e serverless.
Python. Go. PostgreSQL. AWS. Heroku.

20+ APIs
120+ serviços

3m+ produtos
30k+ logistas
10m+ anúncios

200k+ pedidos
em maio/2020

olist
empowering commerce

Arquitetura do Olist

olist

início

primeiros passos com o olist 🏆

conclua o cadastro para ativar a conta e começar a vender

olist início pedidos produtos perguntas financeiro concluir cadastro Jéssica Teste

estoque & preço

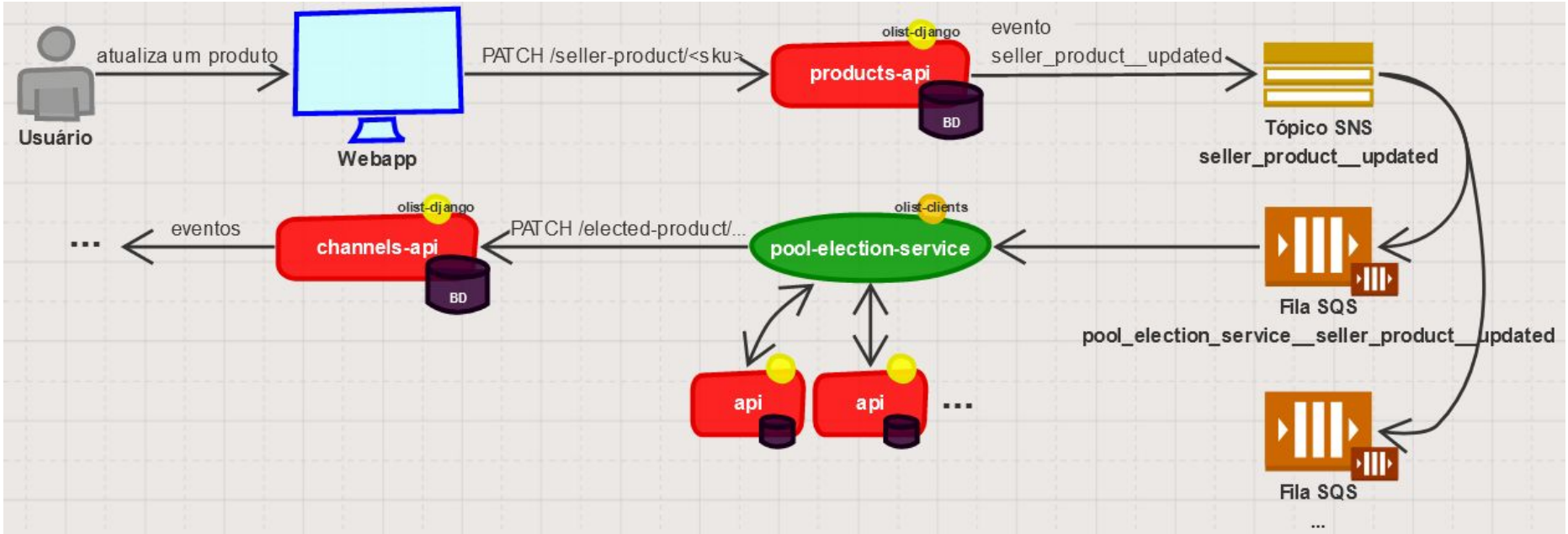
produtos sem estoque

buscar por título ou código

Mantenha as informações dos produtos sempre atualizadas para continuar vendendo.

maior oferta

produto	estoque	preço de	preço por
 Aerogerador - Potência máxima 500W - Turbina eólica - gerador d EAN 7909389166204 SKU PRDOR1EX3AO2I1KE	9	R$ 999,99	R$ 999,99
 Almofada Infantil Belchior Pets 421001 43x29 cm EAN 7899873463103 SKU PRDK00CLD6ZI8XSD	10	R$ 50,00	R$ 50,00
 Almofada Infantil Belchior Pets 421004 34x43 cm EAN 7899873463394 SKU PRDBP17HGI5MNAVK	5	R$ 50,00	R$ 50,00


atualiza um produto
Usuário
Webapp
PATCH /seller-product/<sku>
olist-django
products-api
BD
evento
seller_product__updated
Tópico SNS
seller_product__updated
Fila SQS
pool_election_service__seller_product__updated
Fila SQS
...
olist-clients
pool-election-service
PATCH /elected-product/...
olist-django
channels-api
BD
eventos
...
api
api
...

Techstack

- Python 3
- Django Rest Framework (DRF)
- Heroku
- Postgres
- AWS (API Gateway, SQS, SNS, S3, IAM, Lambda, RDS, Athena, ...)

- Outros: Redis, Javascript, Go, Lambda, ElasticSearch, ...

Onde melhorar?

Como medir?

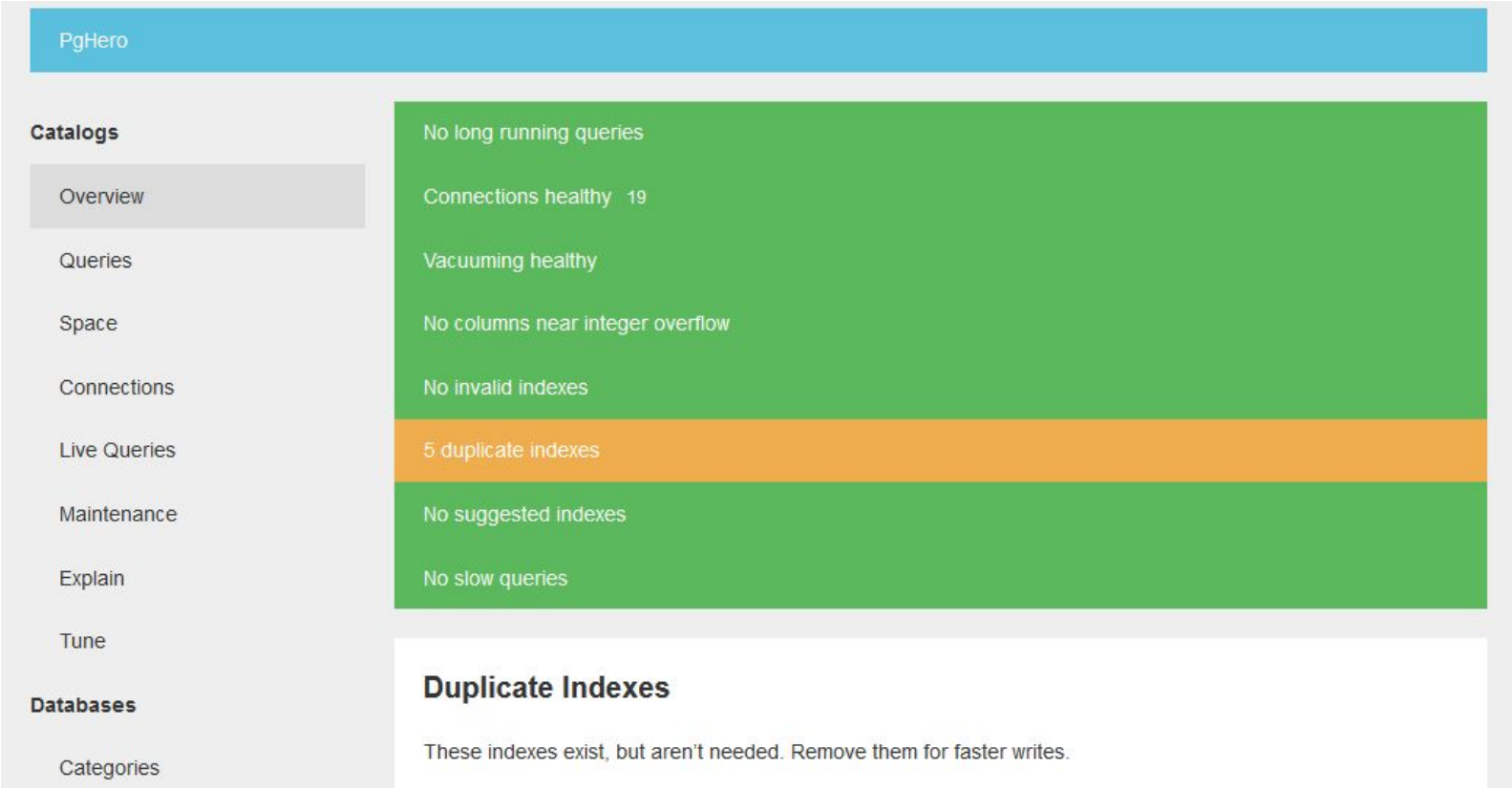
Como melhorar?

Onde melhorar?


APM / opbeans-go / Transactions
opbeans-go
Transactions
Transaction duration
Requests per minute
200 rpm
100 rpm
0 rpm
03 PM
06 PM
09 PM
Thu 30
03 AM
Transaction sample
Timeline
HTTP
Service
Process
Agent
URL
Container
User
Services
0ms
10ms
20ms
30ms
40ms
50ms
Elastic APM

olist

pgHero



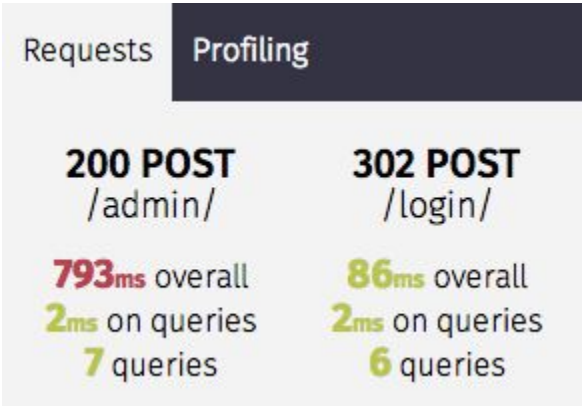
olist

Como medir?

Django-silk

https://github.com/jazzband/django-silk

Profiling e inspeção



Siege

https://github.com/JoeDog/siege

Teste de carga e benchmark

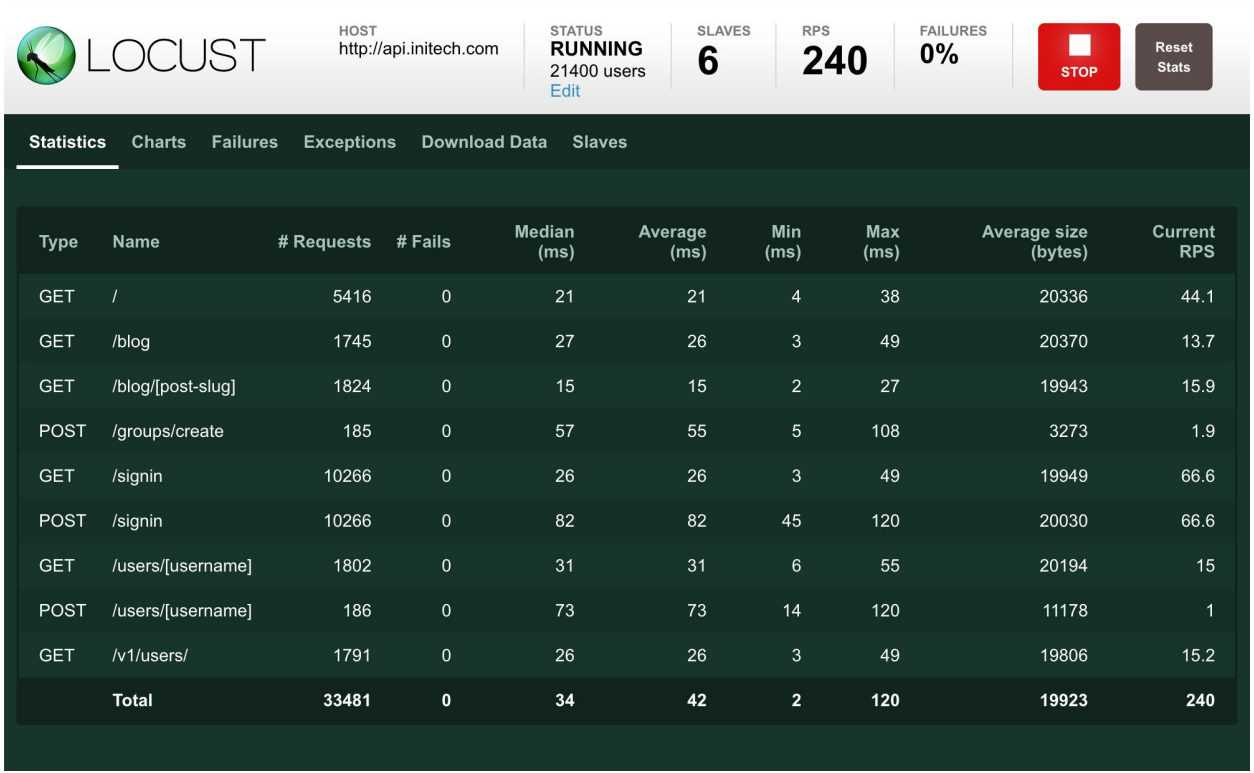
```
siege -r 100 http://www.acme.com
```

```
Transactions:                   1500 hits
Availability:                 100.00 %
Elapsed time:                 264.56 secs
Data transferred:              23.08 MB
Response time:                  1.97 secs
Transaction rate:               5.67 trans/sec
Throughput:                     0.09 MB/sec
Concurrency:                   11.14
Successful transactions:        1500
Failed transactions:               0
Longest transaction:           12.60
Shortest transaction:           0.92
```

Locust

https://locust.io/

Teste de carga com scripting em Python



Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	5416	0	21	21	4	38	20336	44.1
GET	/blog	1745	0	27	26	3	49	20370	13.7
GET	/blog/[post-slug]	1824	0	15	15	2	27	19943	15.9
POST	/groups/create	185	0	57	55	5	108	3273	1.9
GET	/signin	10266	0	26	26	3	49	19949	66.6
POST	/signin	10266	0	82	82	45	120	20030	66.6
GET	/users/[username]	1802	0	31	31	6	55	20194	15
POST	/users/[username]	186	0	73	73	14	120	11178	1
GET	/v1/users/	1791	0	26	26	3	49	19806	15.2
	Total	**33481**	**0**	**34**	**42**	**2**	**120**	**19923**	**240**

olist

Como melhorar?

Acesso ao Banco de Dados

olist

Exemplo

Para 100 Books, onde cada um tem uma FK para um Publisher...

```
queryset = Book.objects.all()
for book in queryset:
    print(str({'name': book.name, 'publisher': book.publisher.name}))
```

Exemplo

Para 100 Books, onde cada um tem uma FK para um Publisher...

```
queryset = Book.objects.all()
for book in queryset:
    print(str({'name': book.name, 'publisher': book.publisher.name})) # 101 queries
```

Exemplo: select_related

Para 100 Books, onde cada um tem uma FK para um Publisher...

```
queryset = Book.objects.all()
for book in queryset:
    print(str({'name': book.name, 'publisher': book.publisher.name})) # 101 queries

queryset = Book.objects.select_related('publisher').all()
for book in queryset:
    print(str({'name': book.name, 'publisher': book.publisher.name})) # 1 query
```

Exemplo

Para 10 Stores, onde cada uma tem 10 Books, em uma relação many-to-many...

```
queryset = Store.objects.all()
for store in queryset :
    books = [book.name for book in store.books.all()]
    print(str({'name': store.name, 'books': books}))
```

Exemplo

Para 10 Stores, onde cada uma tem 10 Books, em uma relação many-to-many...

```
queryset = Store.objects.all()
for store in queryset :
    books = [book.name for book in store.books.all()]
    print(str({'name': store.name, 'books': books})) # 11 queries
```

Exemplo: prefetch_related

Para 10 Stores, onde cada uma tem 10 Books, em uma relação many-to-many...

```
queryset = Store.objects.all()
for store in queryset:
    books = [book.name for book in store.books.all()]
    print(str({'name': store.name, 'books': books})) # 11 queries

queryset = Store.objects.prefetch_related('books')
for store in queryset:
    books = [book.name for book in store.books.all()]
    print(str({'name': store.name, 'books': books})) # 2 queries
```

Exemplo: prefetch_related com filtro

Para 10 Stores, onde cada uma tem 10 Books, em uma relação many-to-many...

```
queryset = Store.objects.prefetch_related('books')
for store in queryset:
    books = [book.name for book in store.books.filter(price__range=(250, 300))]
    print(str({'name': store.name, 'books': books}))
```

Exemplo: prefetch_related com filtro

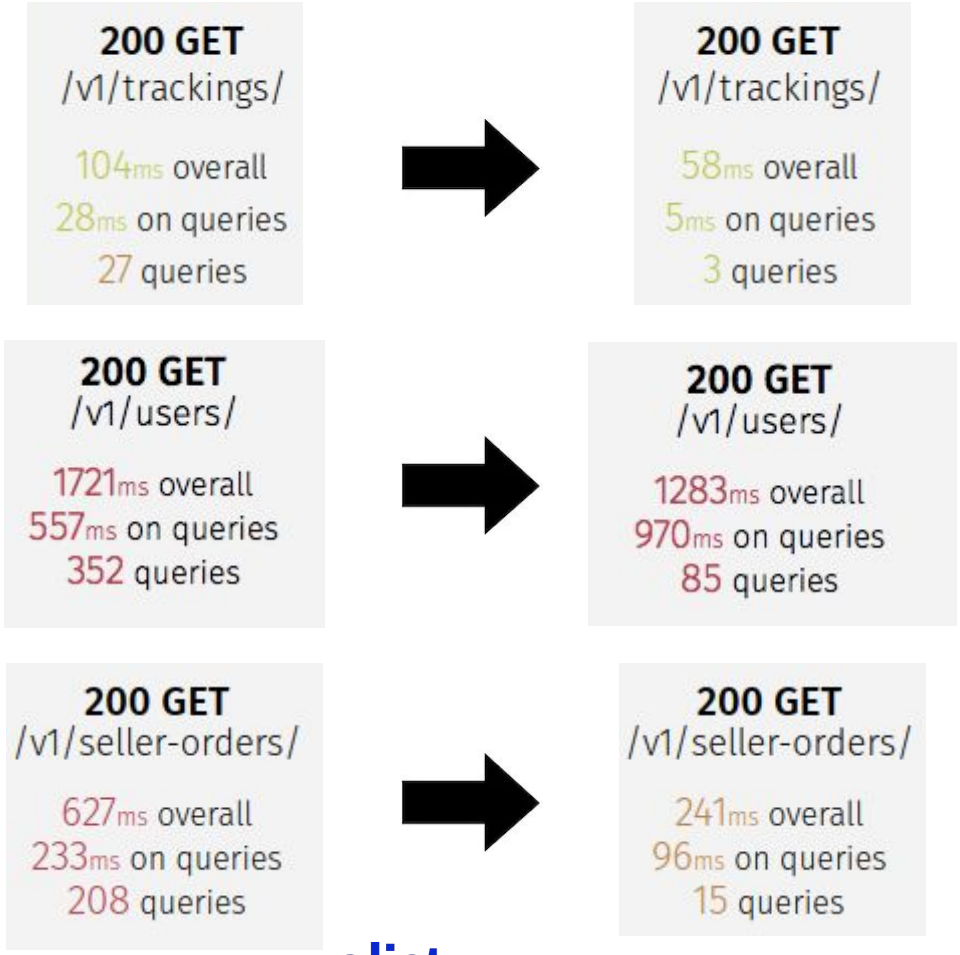
Para 10 Stores, onde cada uma tem 10 Books, em uma relação many-to-many...

```
queryset = Store.objects.prefetch_related('books')
for store in queryset:
    books = [book.name for book in store.books.filter(price__range=(250, 300))]
    print(str({'name': store.name, 'books': books})) # 12 queries
```

Exemplo: prefetch_related com filtro

```
queryset = Store.objects.prefetch_related(
Prefetch('books', queryset=Book.objects.filter(price__range=(250, 300))))

for store in queryset:
    books = [book.name for book in store.books.all()]
    print(str({'name': store.name, 'books': books})) # 2 queries
```

Exemplo



olist

Suporte a Partial Response

olist

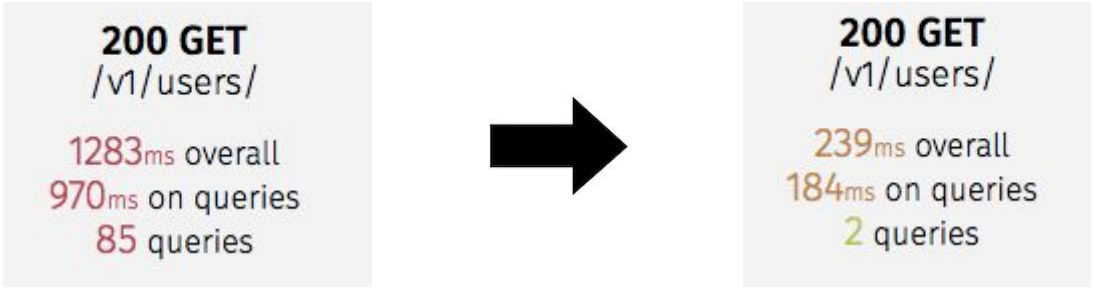
[django-rest-framework-jsonmask]

```
GET /api/tickets/

200 OK
[
    {
        "id": 1,
        "title": "This is a ticket",
        "body": "This is its text",
        "author": {
            "id": 5,
            "username": "HomerSimpson",
        }
    }
]
```

```
GET /api/tickets/?fields=id,title,body

200 OK
[
    {
        "id": 1,
        "title": "This is a ticket",
        "body": "This is its text"
    }
]
```

olist

Exemplo: Endpoint de autenticação



Como usar?

- Herdar rest_framework_jsonmask.views.OptimizedQuerySetMixin
- Herdar rest_framework_jsonmask.serializers.FieldsListSerializerMixin

```
def get_queryset(self):
    queryset = super().get_queryset()
    return queryset.select_related("owner").prefetch_related("addresses")
```

```
@data_predicate("owner")
def load_owner(self, queryset):
    return queryset.select_related("owner")
```

```
@data_predicate("addresses")
def load_addresses(self, queryset):
    return queryset.prefetch_related("addresses")
```

Renderização

[django-rest-framework-rapidjson] ou [drf-ujson-renderer]

```
REST_FRAMEWORK = {
    'DEFAULT_RENDERER_CLASSES': (
        'rest_framework_rapidjson.renderers.RapidJSONRenderer',
    ),
    'DEFAULT_PARSER_CLASSES': (
        'rest_framework_rapidjson.parsers.RapidJSONParser',
    )
}
```

/seller-orders/
Antes: Time per request: 1030.060 [ms] (mean)
Depois: Time per request: 973.138 [ms] (mean)

/orders/
Antes: Time per request: 12512.722 [ms] (mean)
Depois: Time per request: 1837.474 [ms] (mean)

Melhorias de Banco de Dados

- Adicionar/Remover índices no BD
- Reestruturar relações no BD
- Ativar full text search do Postgres
- Réplica de leitura do BD
- Dumps de dados antigos de histórico para NoSQL
- Particionamento

Melhorias de Arquitetura

- Usar cache
- Usar código assíncrono
- Evitar ponto único de falha
- Dividir APIs e serviços que ficarem muito grandes
- Evitar contadores em paginação
- Implementar padrão circuit-breaker
- Conteinerização

Valeu!

@jessica.bonson

@jpbonson

olist

Referências

https://medium.com/better-programming/django-select-related-and-prefetch-related-f23043fd635d

https://medium.com/@lucasmagnum/djangotip-select-prefetch-related-e76b683aa457

https://dizballanze.com/django-project-optimization-part-1/

https://k6.io/blog/comparing-best-open-source-load-testing-tools

https://docs.djangoproject.com/en/1.8/topics/db/optimization/#understand-querysets

https://docs.djangoproject.com/en/1.11/ref/models/querysets/#django.db.models.query.QuerySet.select_related

https://medium.com/@hansonkd/performance-problems-in-the-django-orm-1f62b3d04785

http://pythonclub.com.br/django-introducao-queries.html