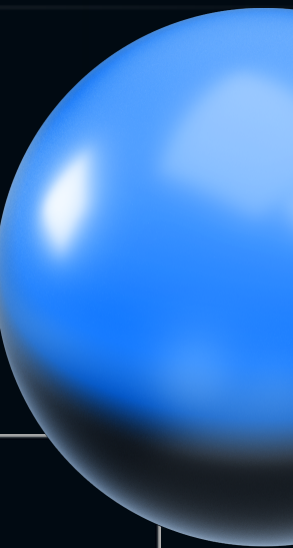
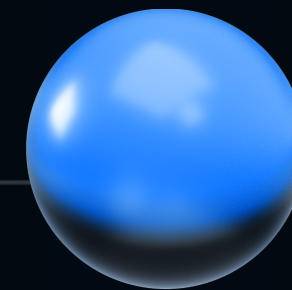


Реактивщина с apache thrift + project armeria



**Виктор
Сильнов**

Альфа банк



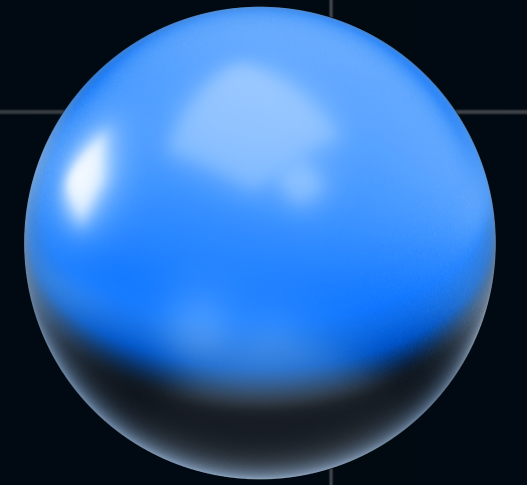
✉ t.me/kto_viktor



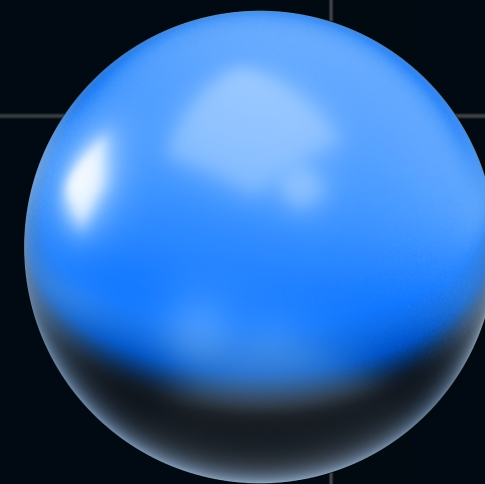
Альфа Банк

Про что доклад

- Проблема, с которой мы столкнулись в синхронных микросервисах
- Решение, которое подошло нам в нашем случае
- Подводные камни, полученный опыт и выводы

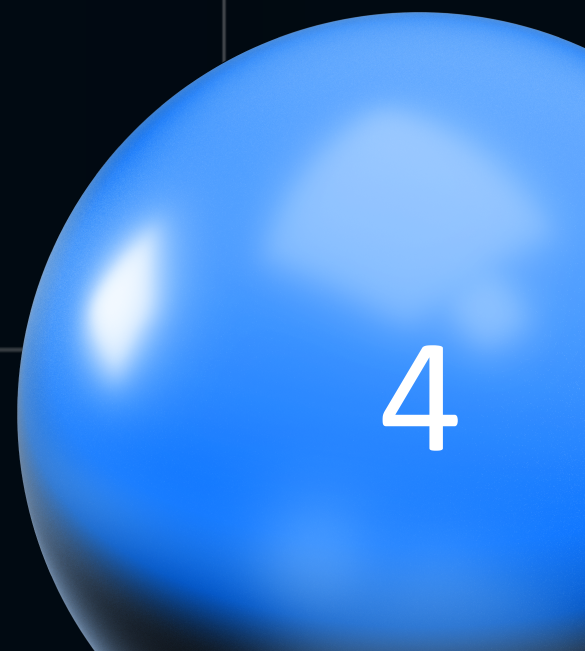
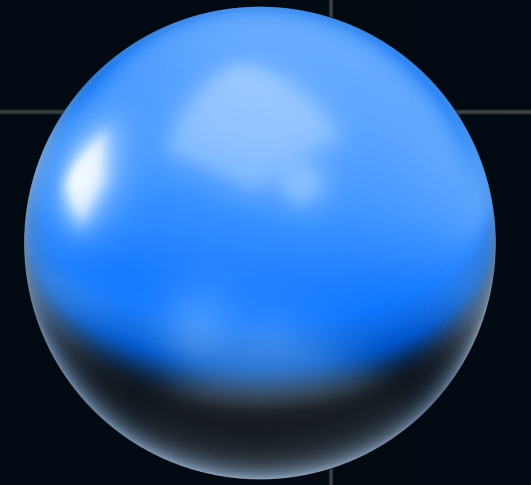


Сценарий



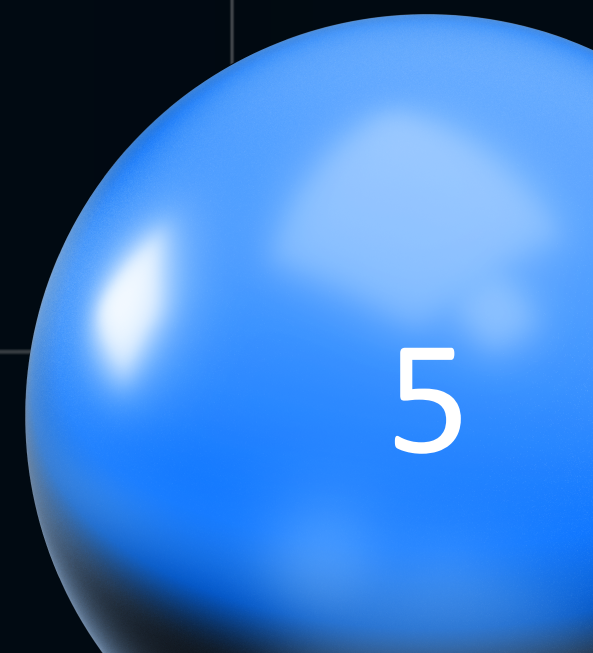
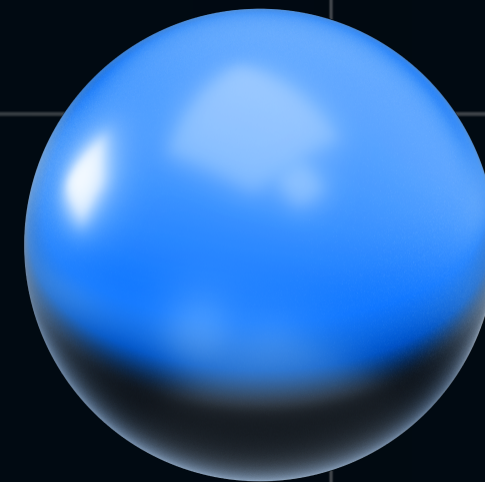
Сценарий

- Банк для бизнеса



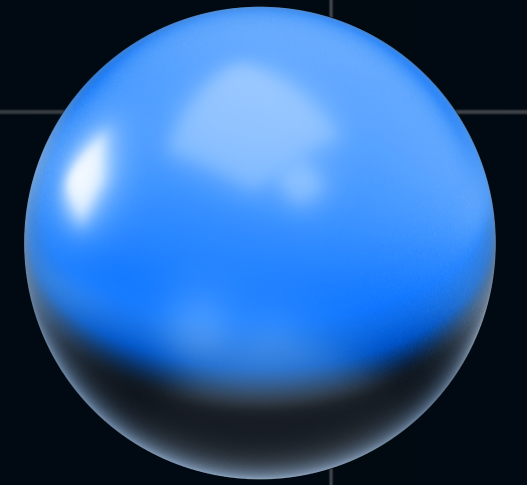
Сценарий

- Банк для бизнеса
- Слой под мобильные платформы



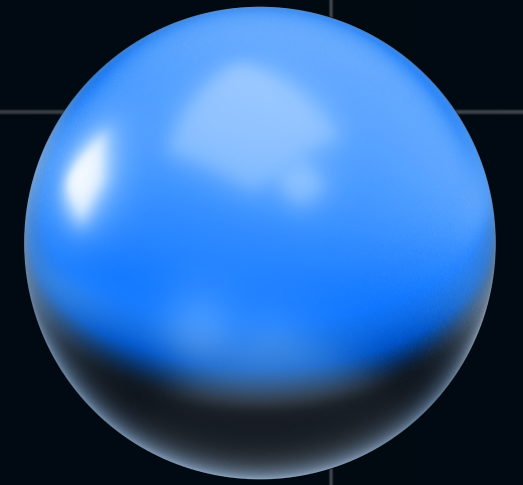
Сценарий

- Банк для бизнеса
- Слой под мобильные платформы
- Кластер микросервисов в mesos/marathon

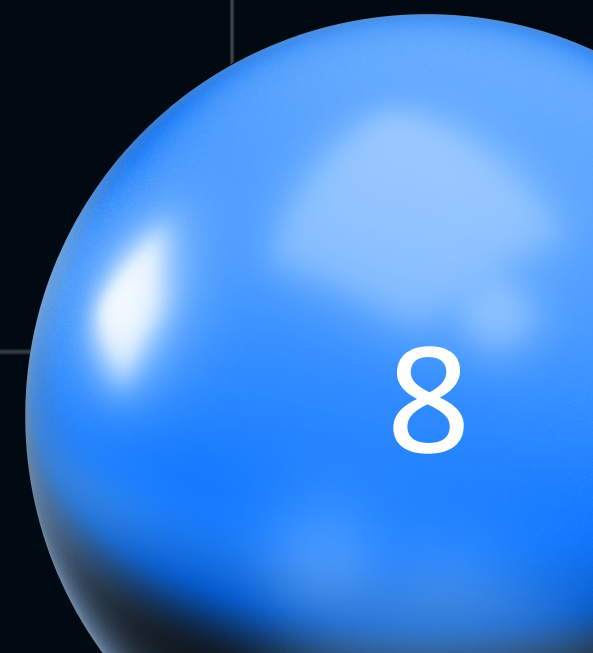
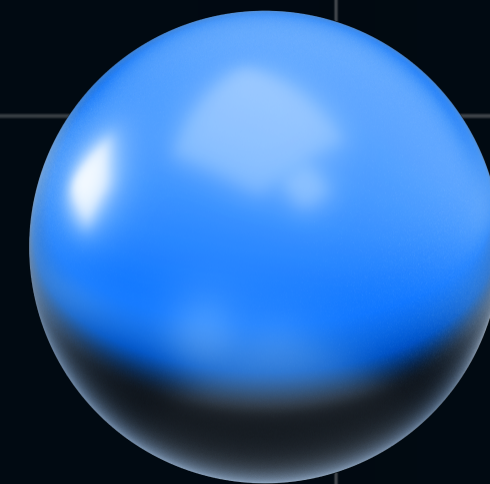


Сценарий

- Банк для бизнеса
- Слой под мобильные платформы
- Кластер микросервисов в mesos/marathon
- Apache thrift

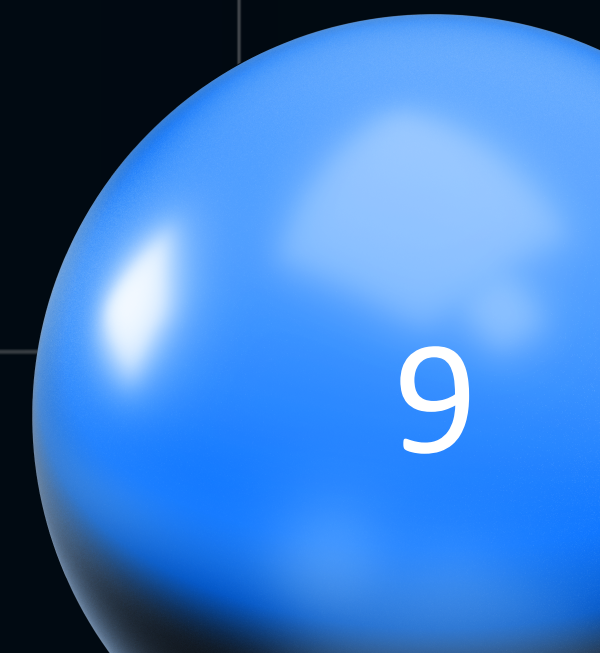
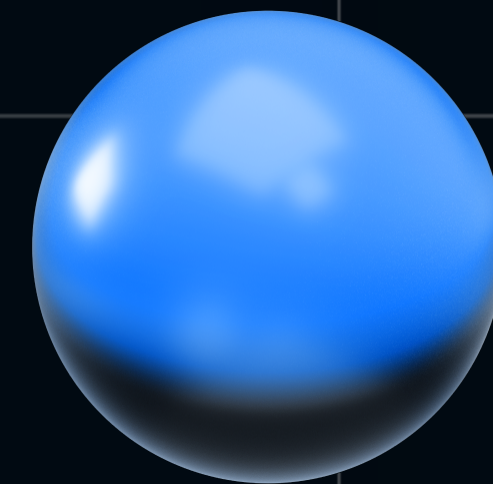


Пара слов про apache thrift



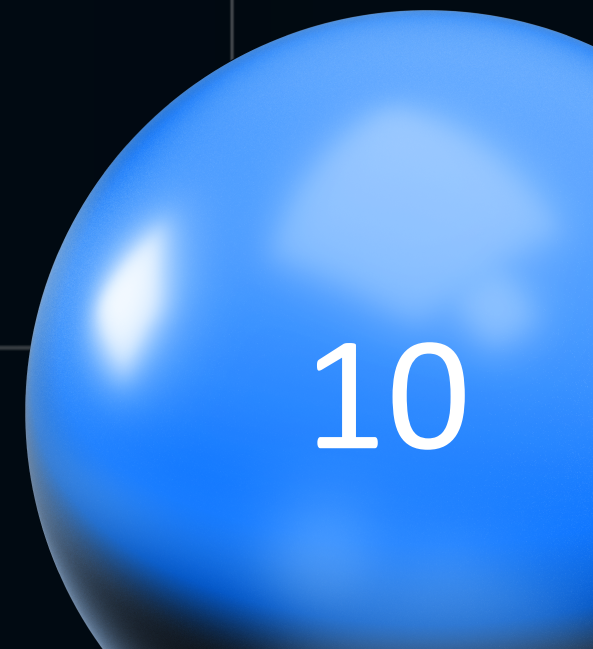
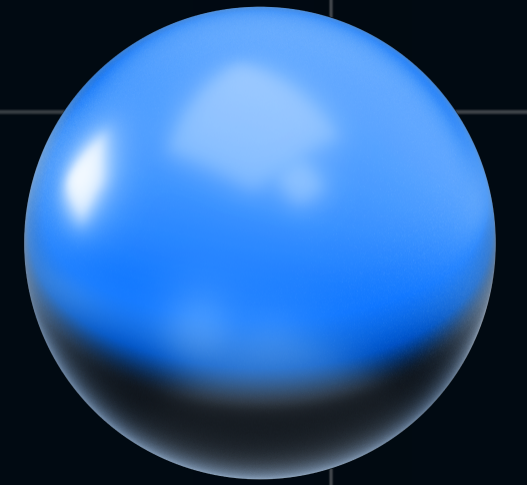
Пара слов про apache thrift

- contract-first



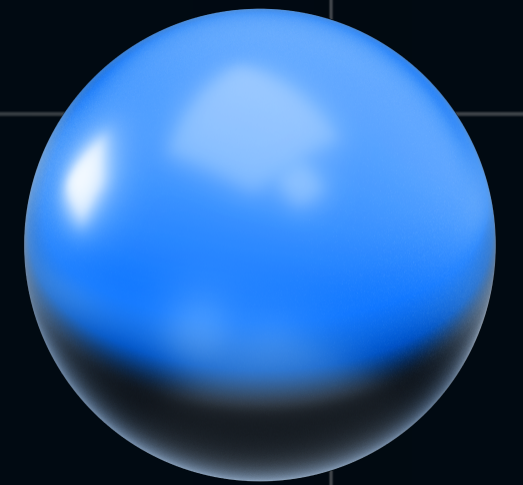
Пара слов про apache thrift

- contract-first
- Кроссплатформенный
(используем как на фронте, так и на бэке)

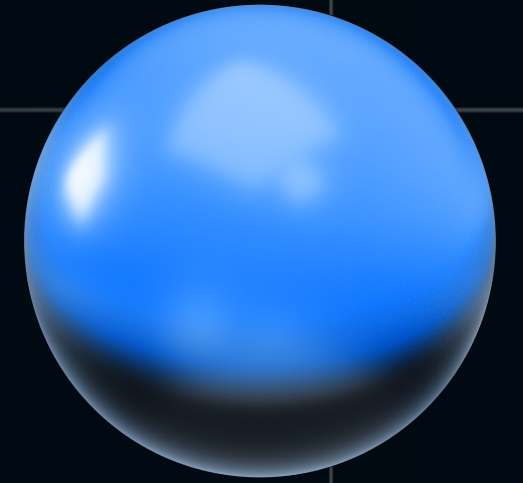


Пара слов про apache thrift

- contract-first
- Кроссплатформенный
(используем как на фронте, так и на бэке)
- обратная совместимость



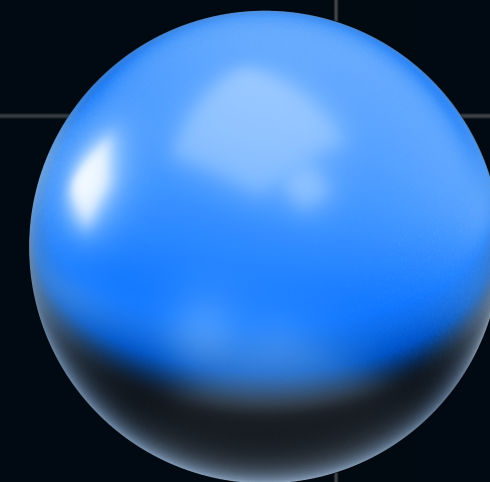
Пара слов про apache thrift



```
struct TSalaryProjectRequest {  
    1: i32          employeesNumber,      # количество сотрудников в компании  
    2: i32          salaryFund,          # фонд оплаты труда в месяц  
    3: string       city,                # город  
    4: string       phoneNumber          # номер телефона  
    5: optional string cityFiasId        # Идентификатор города в Федеральной  
}
```

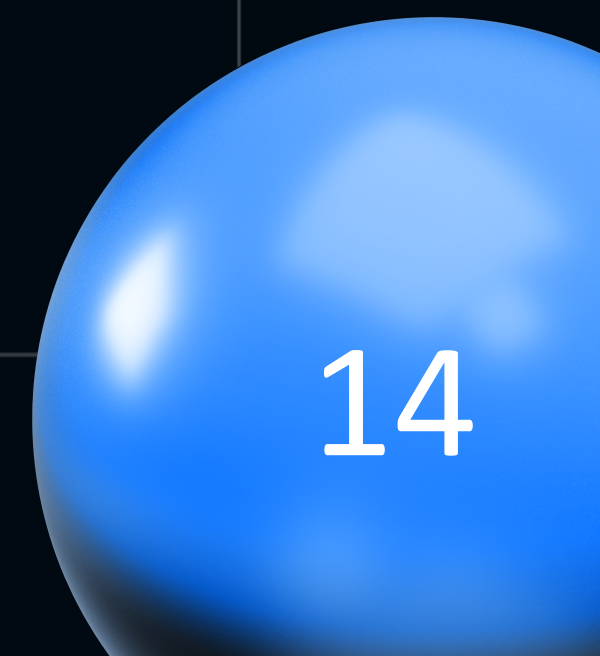
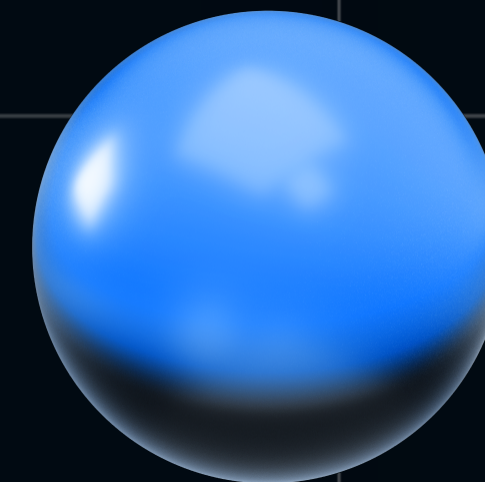


Как работает

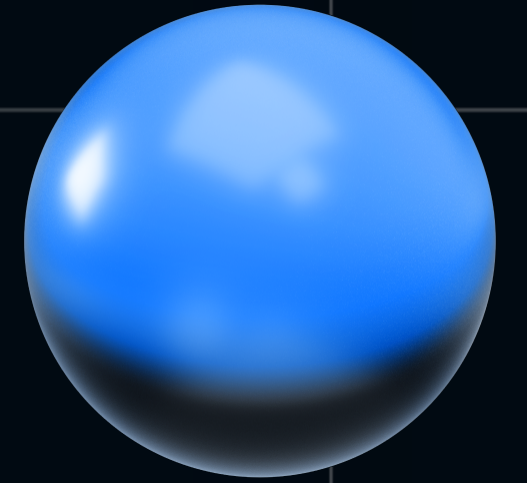


Как работает

- На сервлетах (tomcat)



Как работает



← → ↻ github.com/aatarasoff/spring-thrift-starter

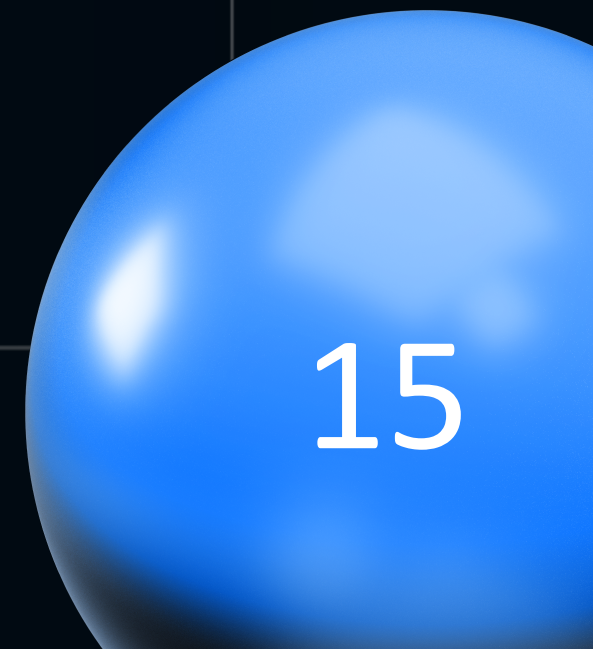
Product ▾ Solutions ▾ Open Source ▾ Pricing

[aatarasoff / spring-thrift-starter](#) Public

[Code](#) [Issues 2](#) [Pull requests 1](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

[master ▾](#) [2 branches](#) [27 tags](#) [Go to file](#) [Code ▾](#)

[aatarasoff](#) add gradle build for each commit or pr ✓ fda9ba1 on May 26, 2022 [🕒 149 commits](#)



Как работает

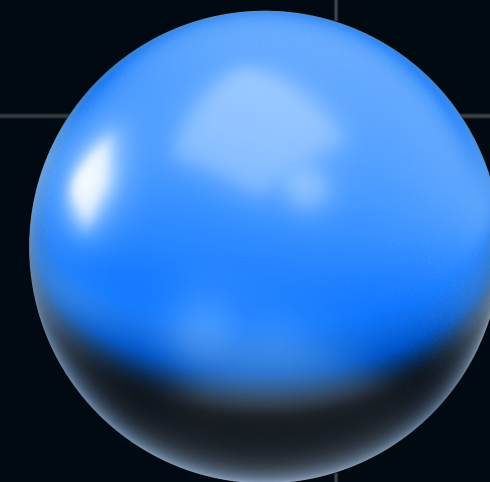
```
@ThriftController("/superapi/superentity")
@RequiredArgsConstructor
public class ActThriftController implements ActApiService.Iface {
    private final ActService actService;

    @Override
    public TActSettings getActSettings(UserData userData, String organizationId) {
        return actService.getActSettings(userData, organizationId);
    }

    @Override
    public TItemsInfo calcItemsInfo(UserData userData, String organizationId, TItemsInfo itemsInfo) {
        return actService.calcItemsInfo(userData, organizationId, itemsInfo);
    }

    @Override
    public File generateActPDF(UserData userData, String organizationId, TActDocument act) {
        return actService.generateActPDF(userData, organizationId, act);
    }
}
```

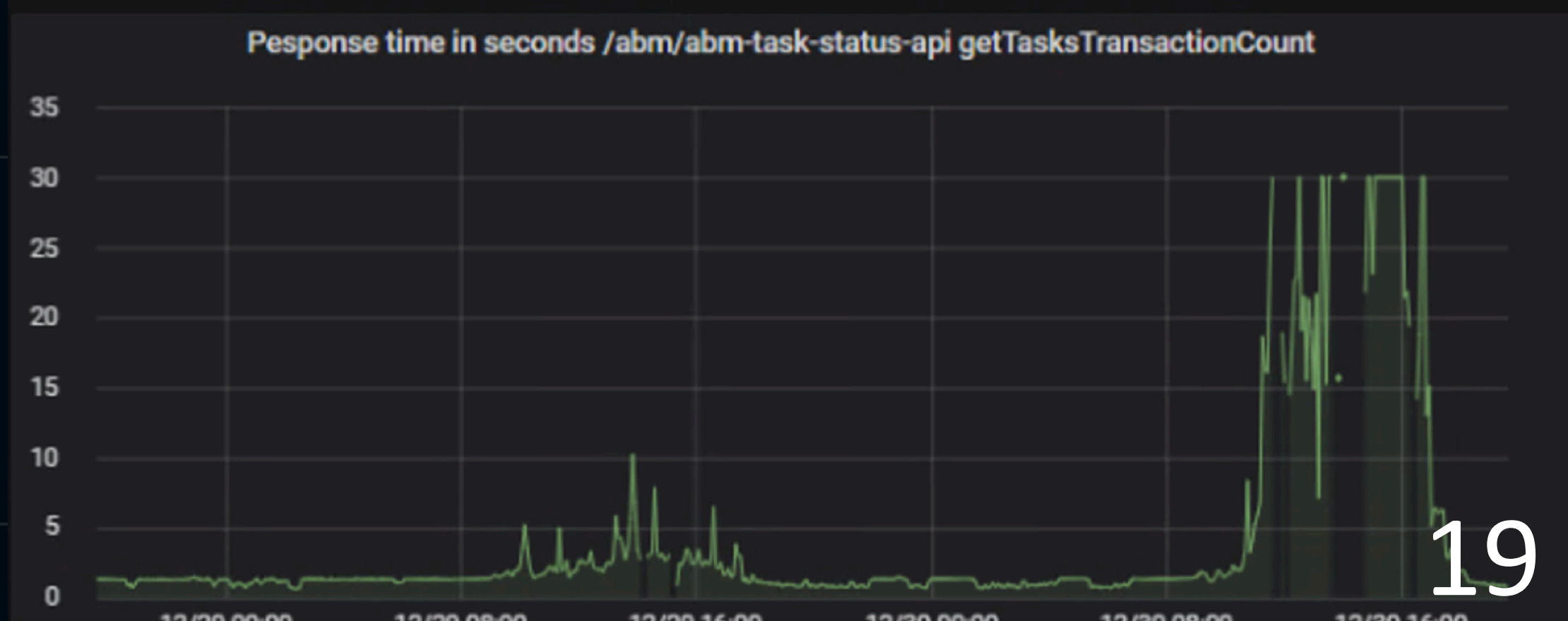
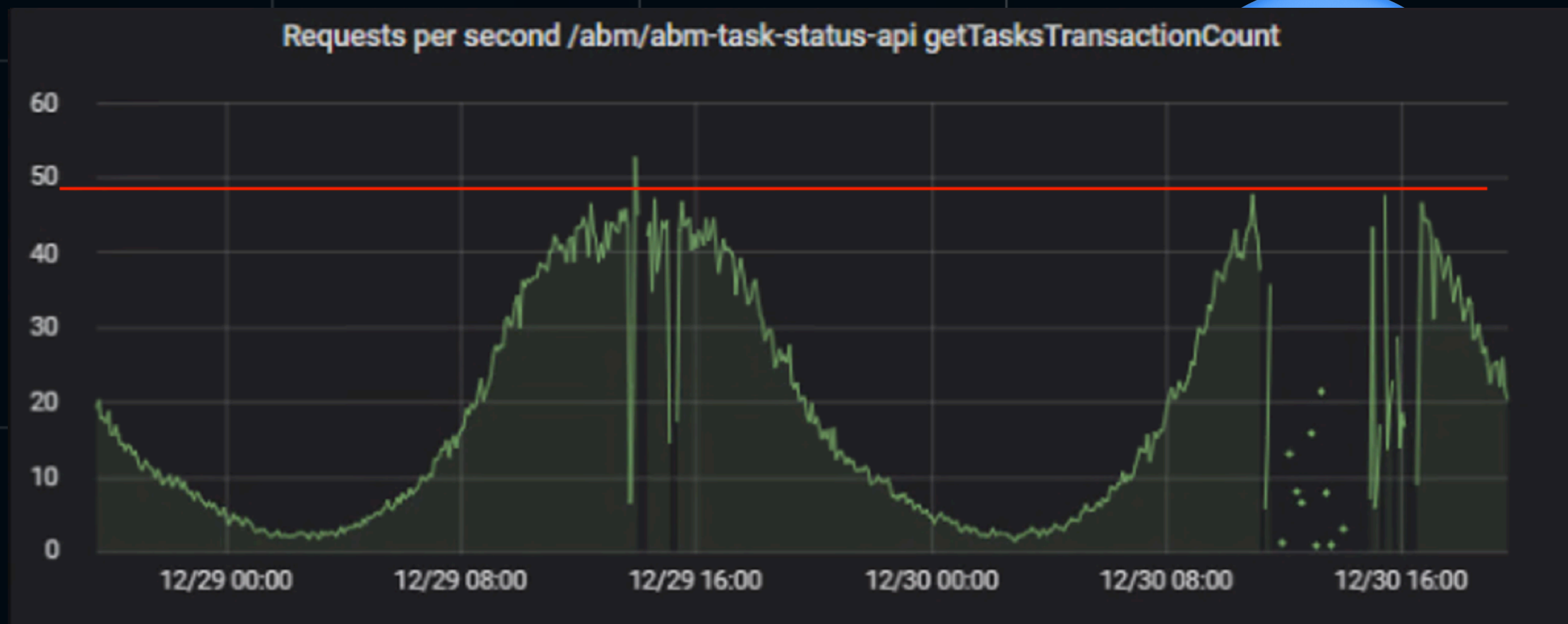

Проблема




Проблема














Проблема



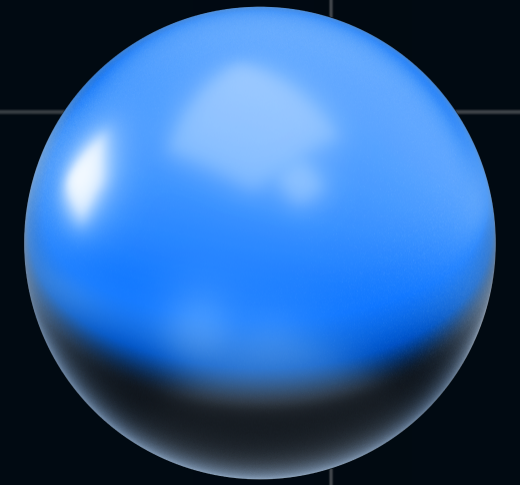
Проблема

 kibana

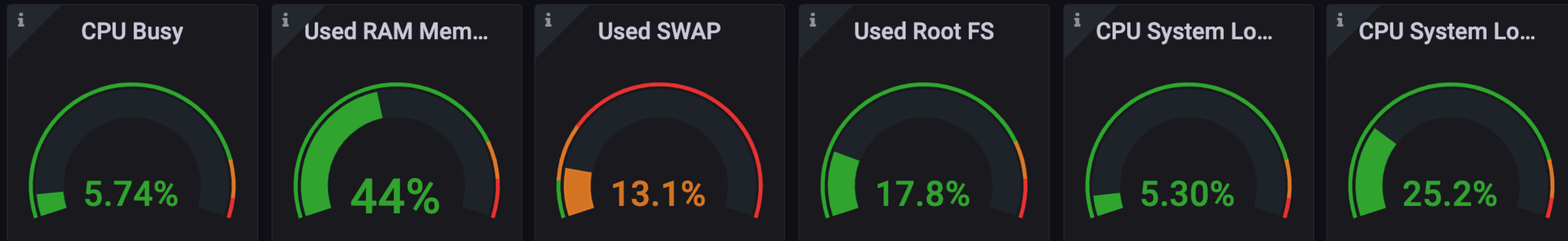
-  Discover
-  Visualize
-  Dashboard
-  Timelion
-  Canvas
-  Maps
-  Infrastructure
-  Logs
-  APM
-  Uptime
-  Graph

- ▶ April 7th 2023, 15:50:00.233 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause
- ▶ April 7th 2023, 15:45:44.735 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause
- ▶ April 7th 2023, 15:45:42.410 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause
- ▶ April 7th 2023, 15:44:49.350 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause
- ▶ April 7th 2023, 15:43:34.334 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [⊕ ⊖] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause
- ▶ April 7th 2023, 15:43:07.118 Servlet.service() for servlet [\$Proxy273Servlet] in context with path [] threw exception [org.apache.thrift.transport.TTransportException: org.apache.catalina.connector.ClientAbortException: java.io.IOException: **Connection reset by peer**] with root cause

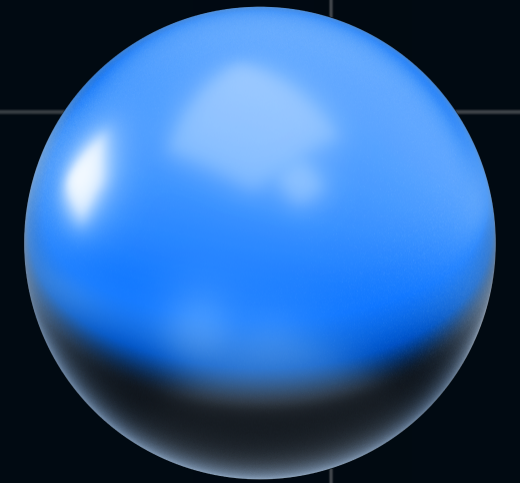
Метрики железа



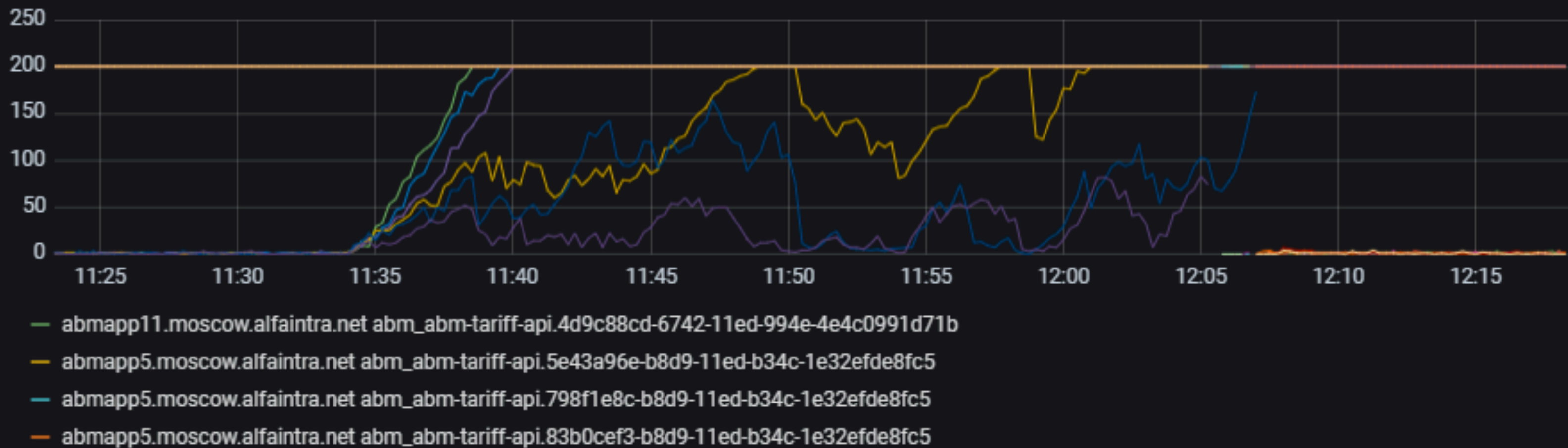
Basic CPU / Mem / Disk Gauge



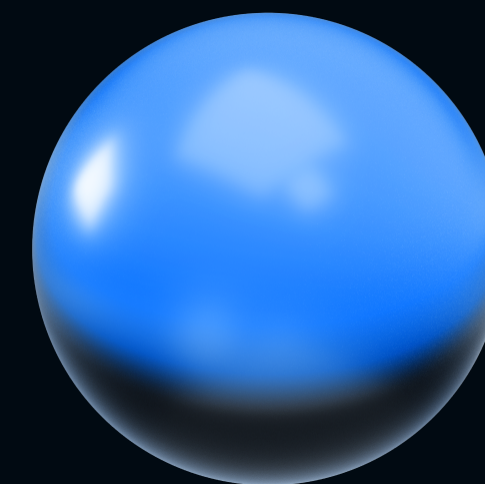
Метрики тредов



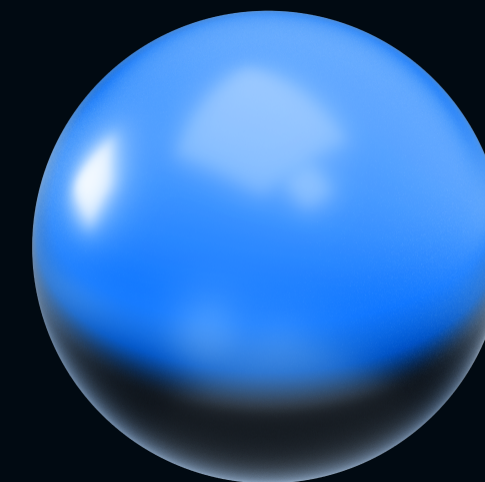
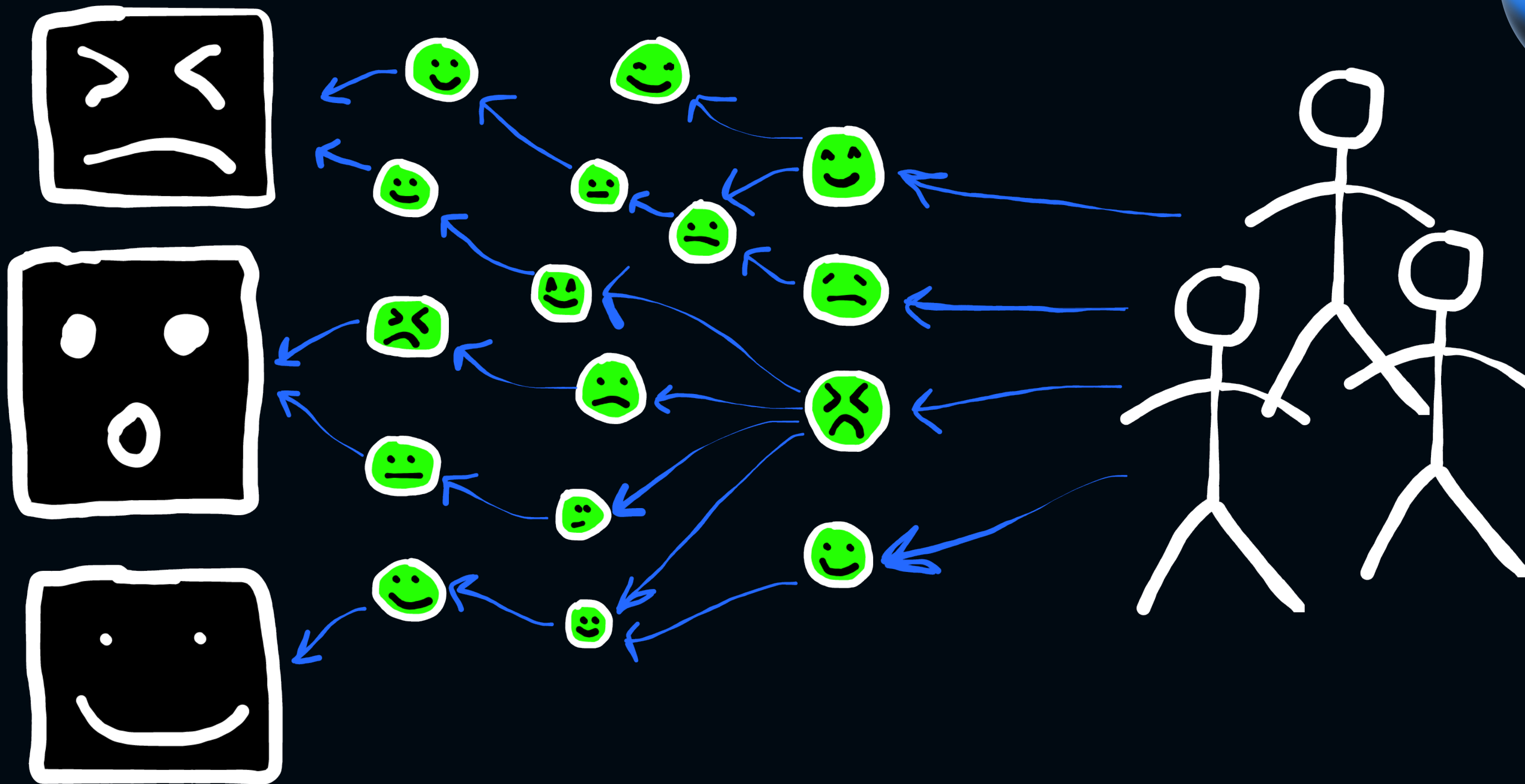
Thread metrics



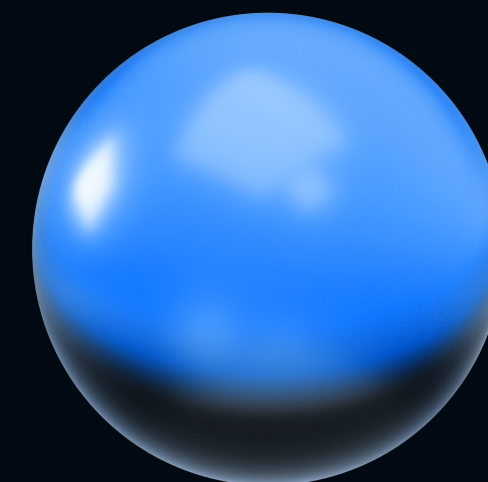
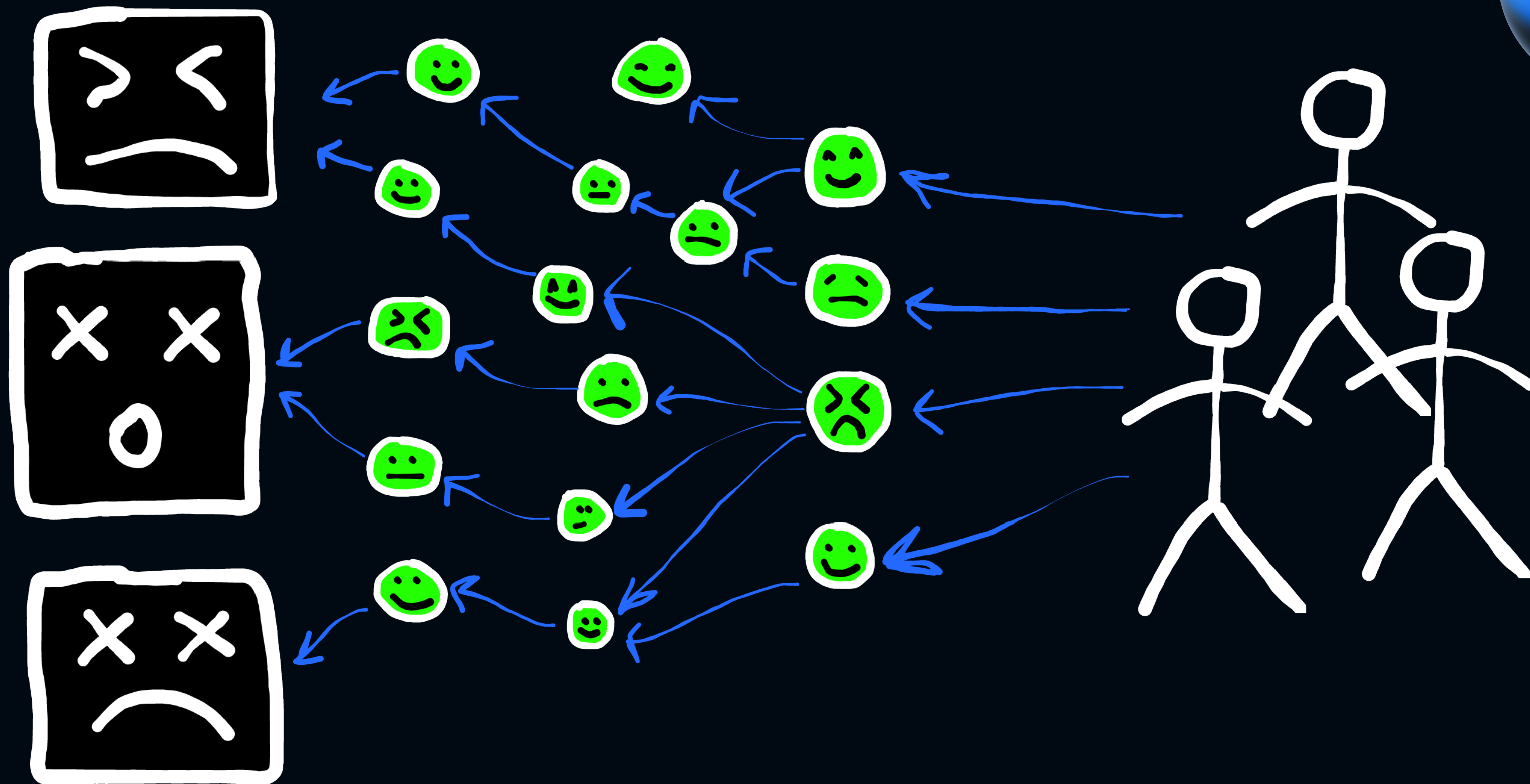
Что случилось



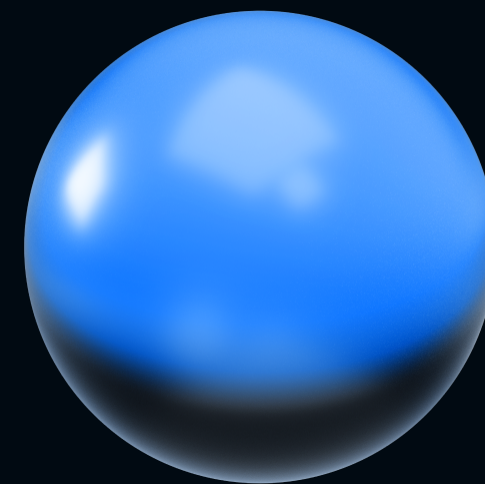
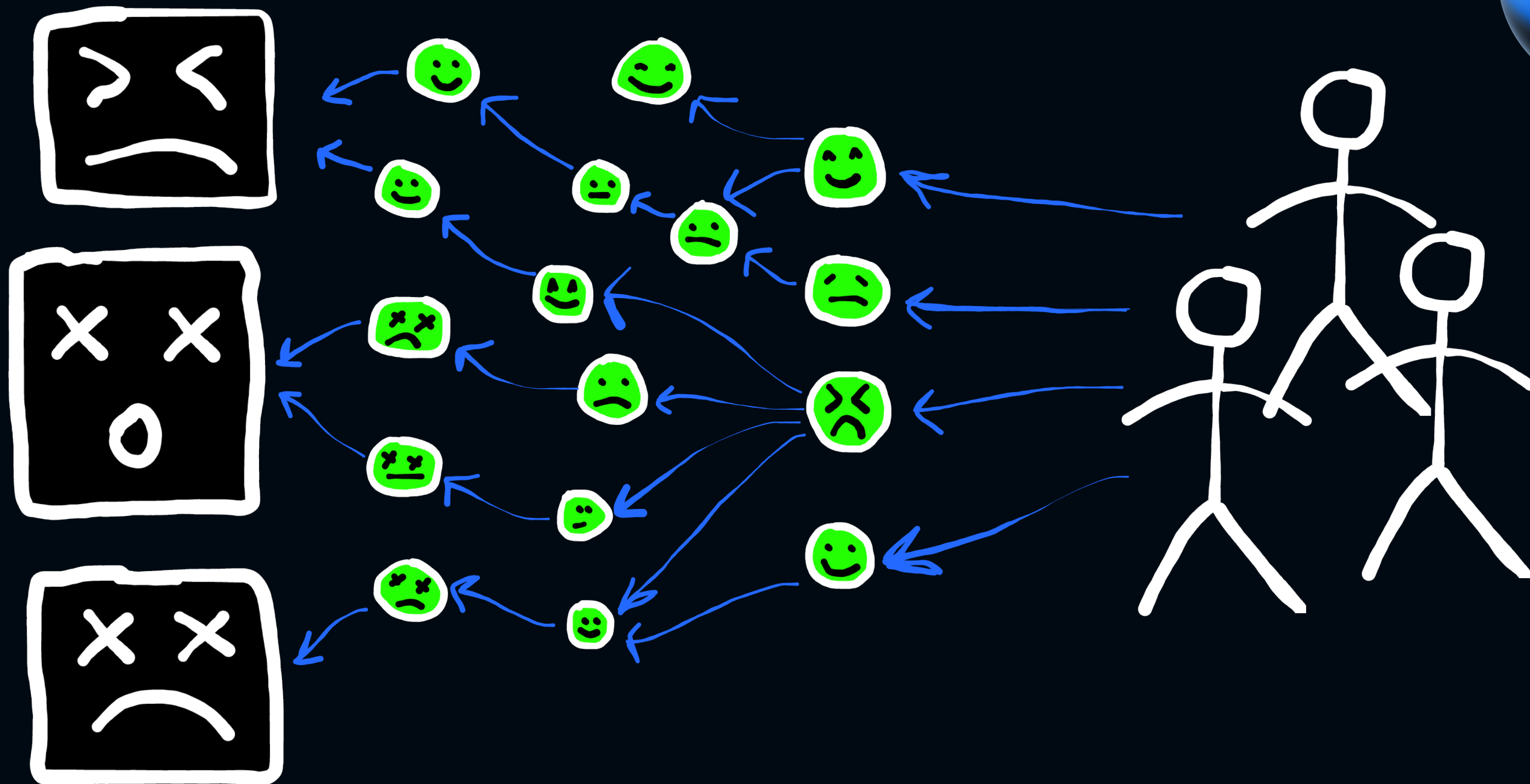
Что случилось



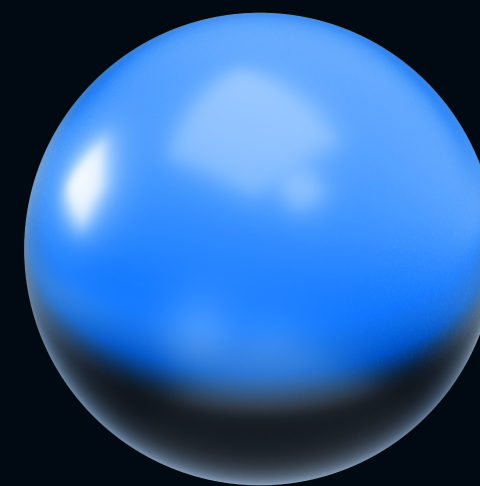
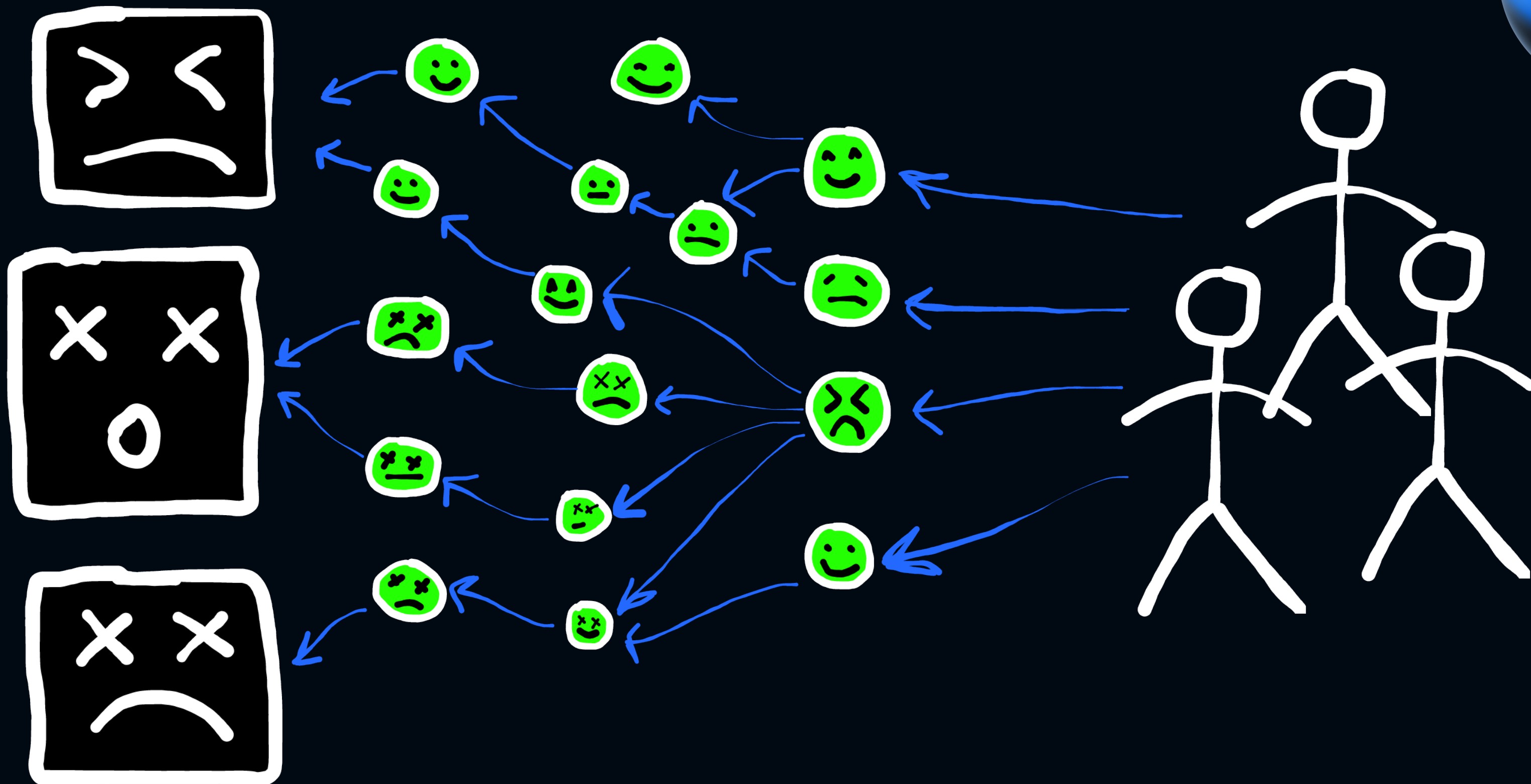
Что случилось



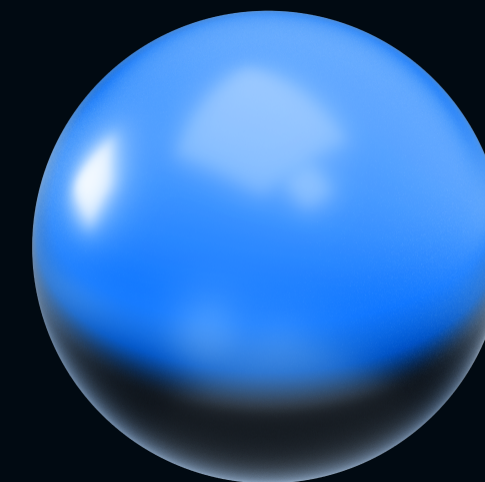
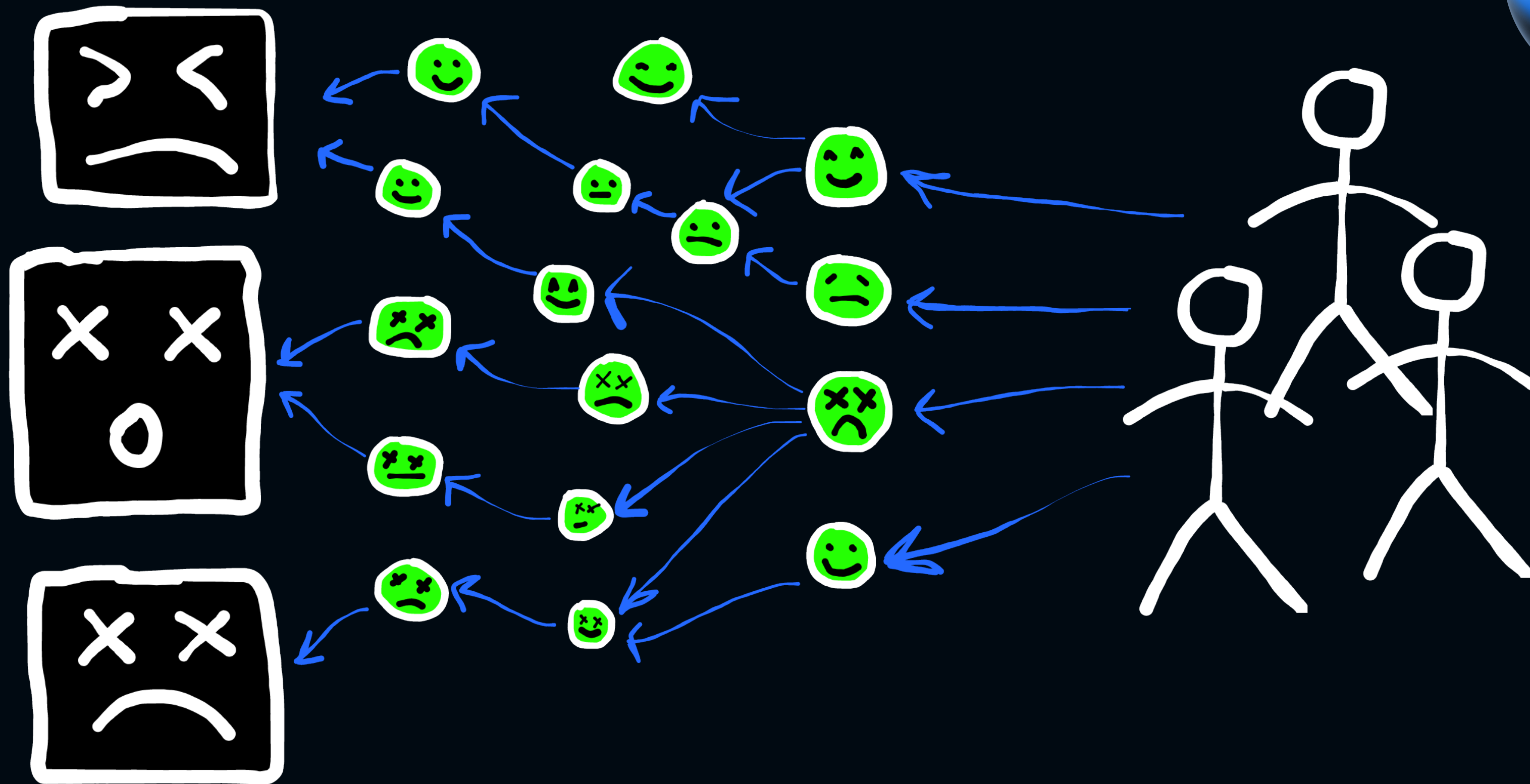
Что случилось



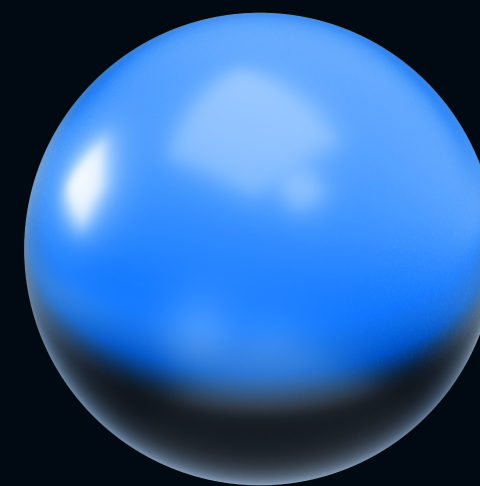
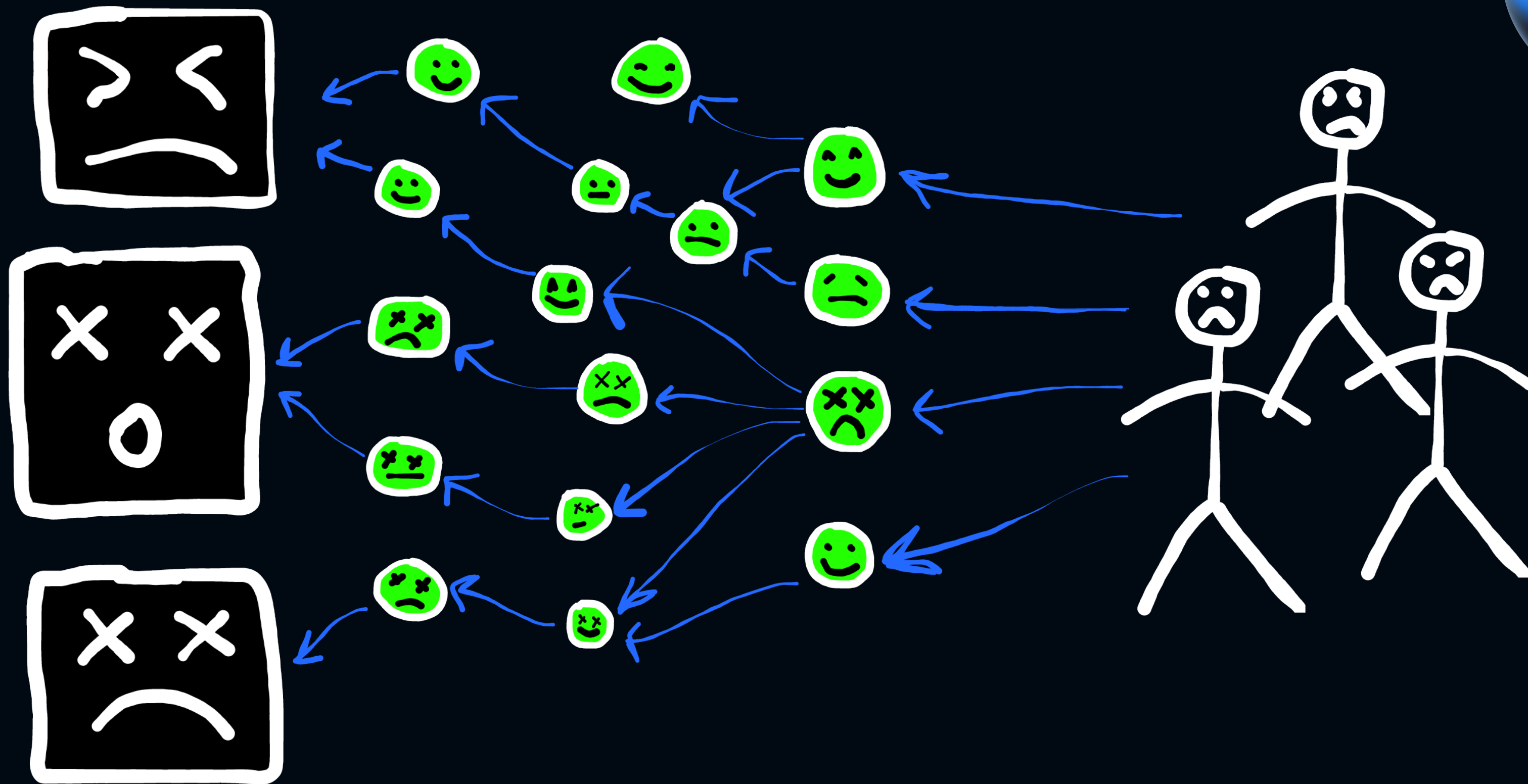
Что случилось



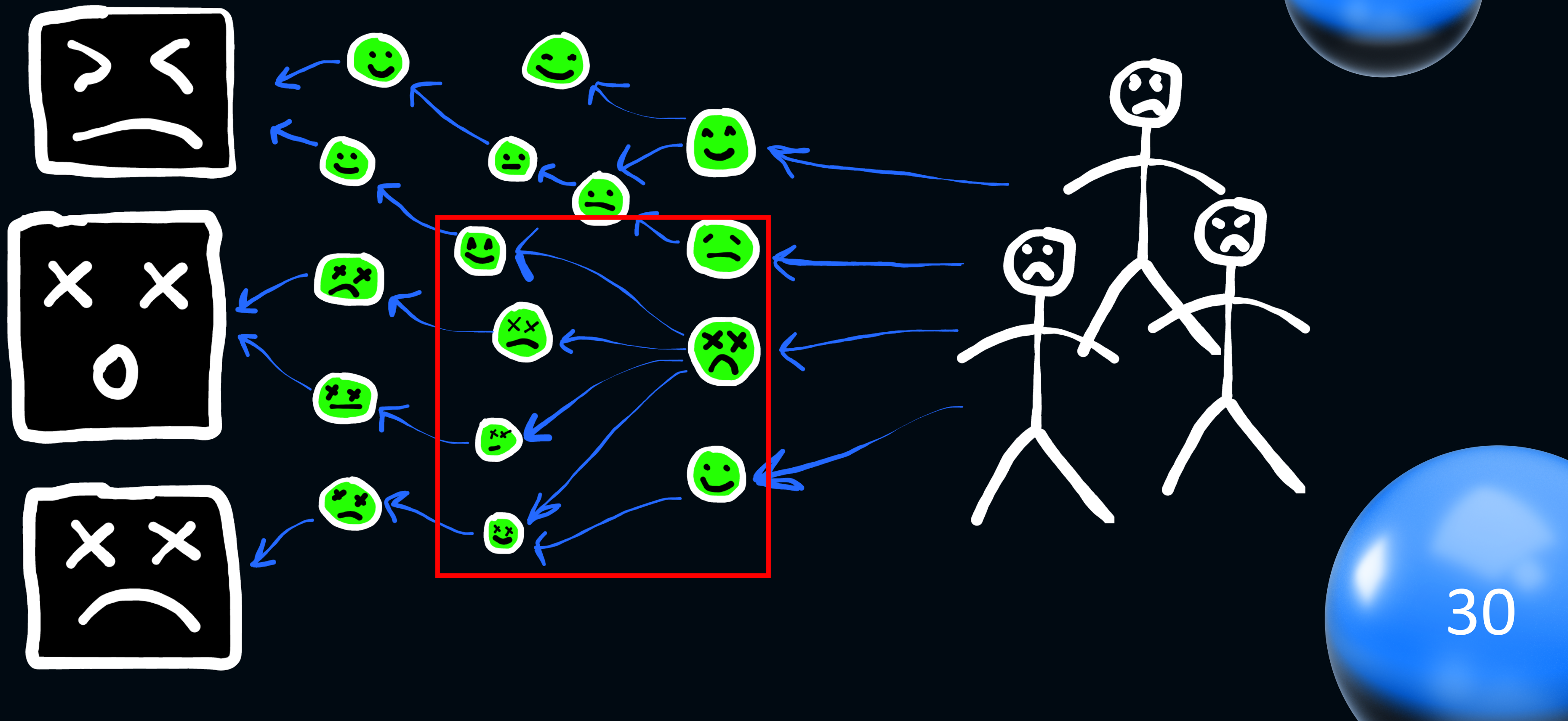
Что случилось



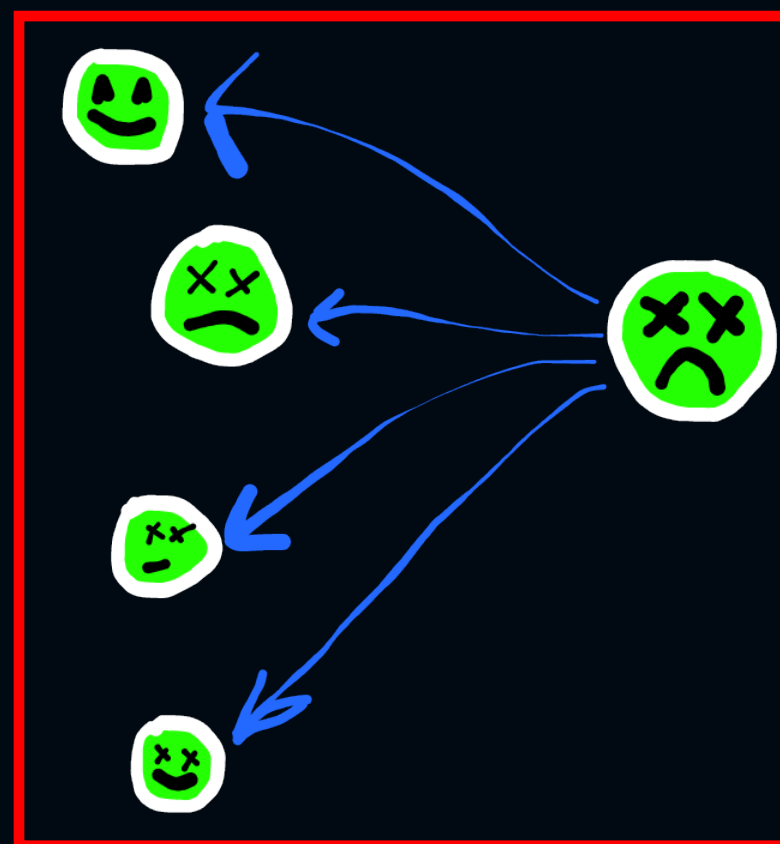
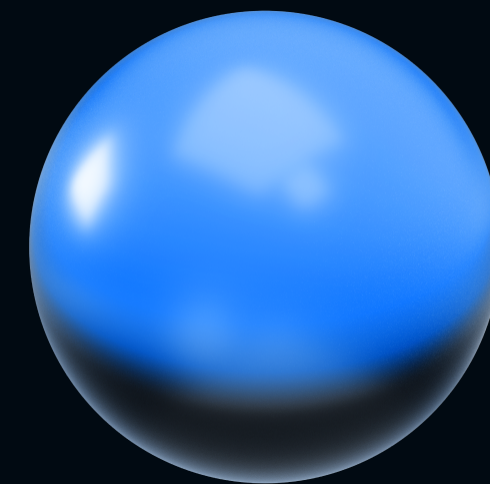
Что случилось



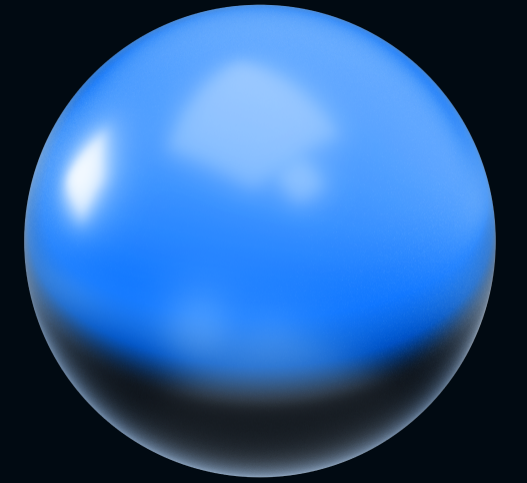
Что случилось



Что случилось



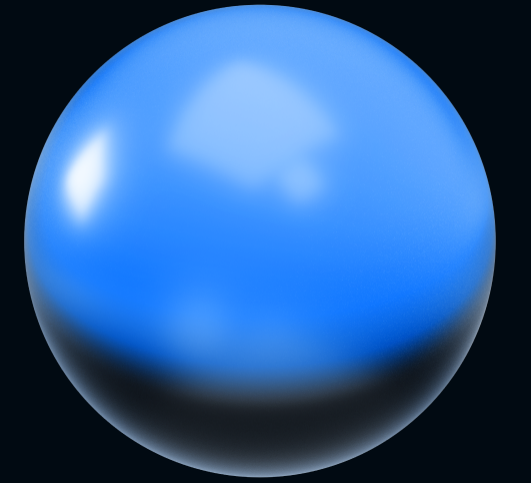
Пример проблемного сервиса



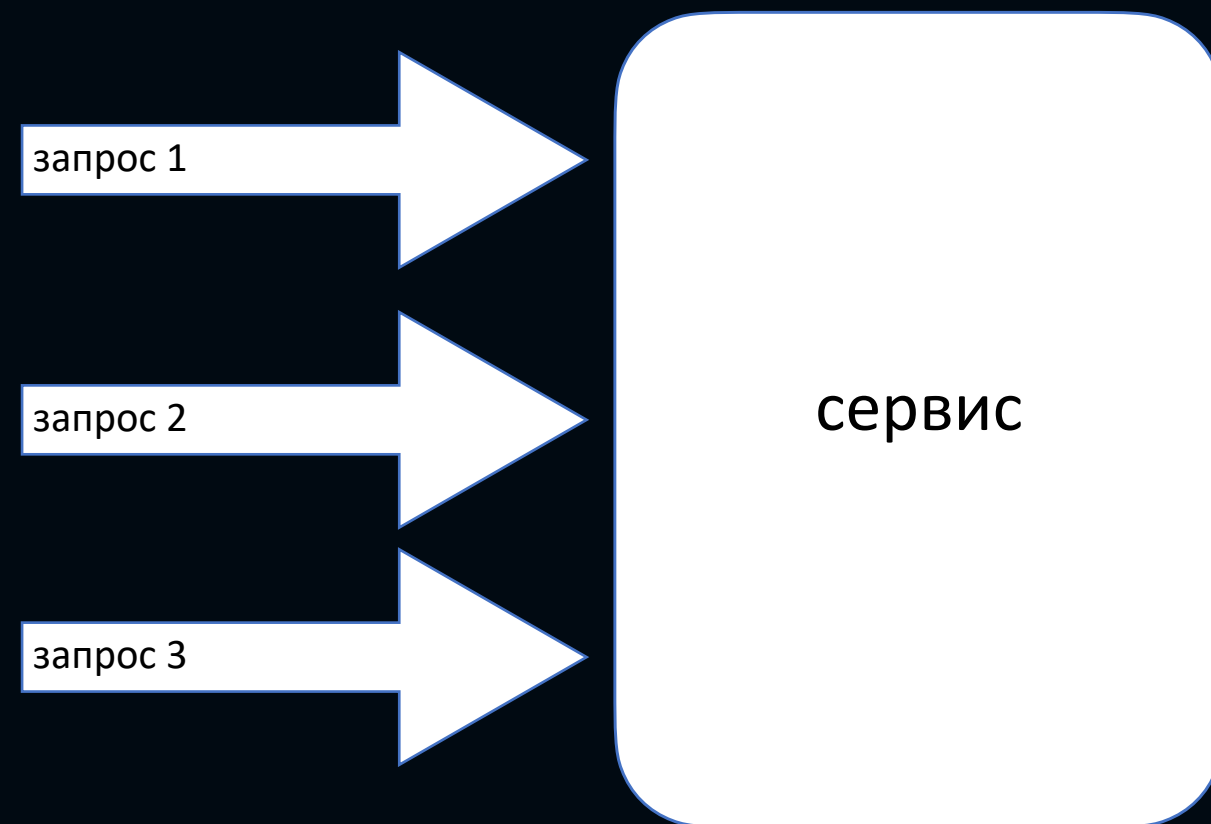
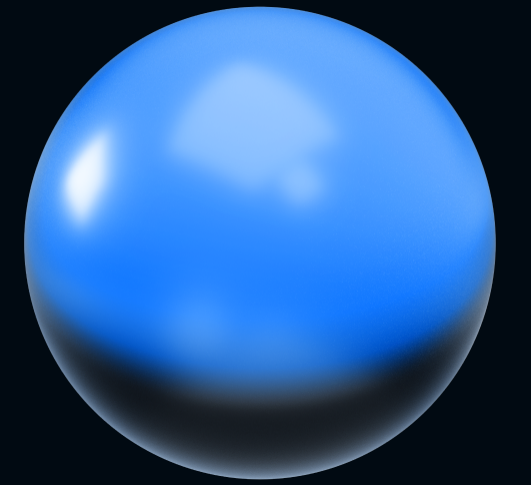
```
fun getPostAuthData(userData: UserData): TPostAuthDataResponse =  
    Observable  
        .zip(  
            getDashboardWithoutTasks(userData),  
            getCustomerFeatures(userData),  
            getProfile(userData),  
            getUserClientBlock(userData),  
            mapper::toResponse  
        )  
        .toBlocking()  
        .single()
```



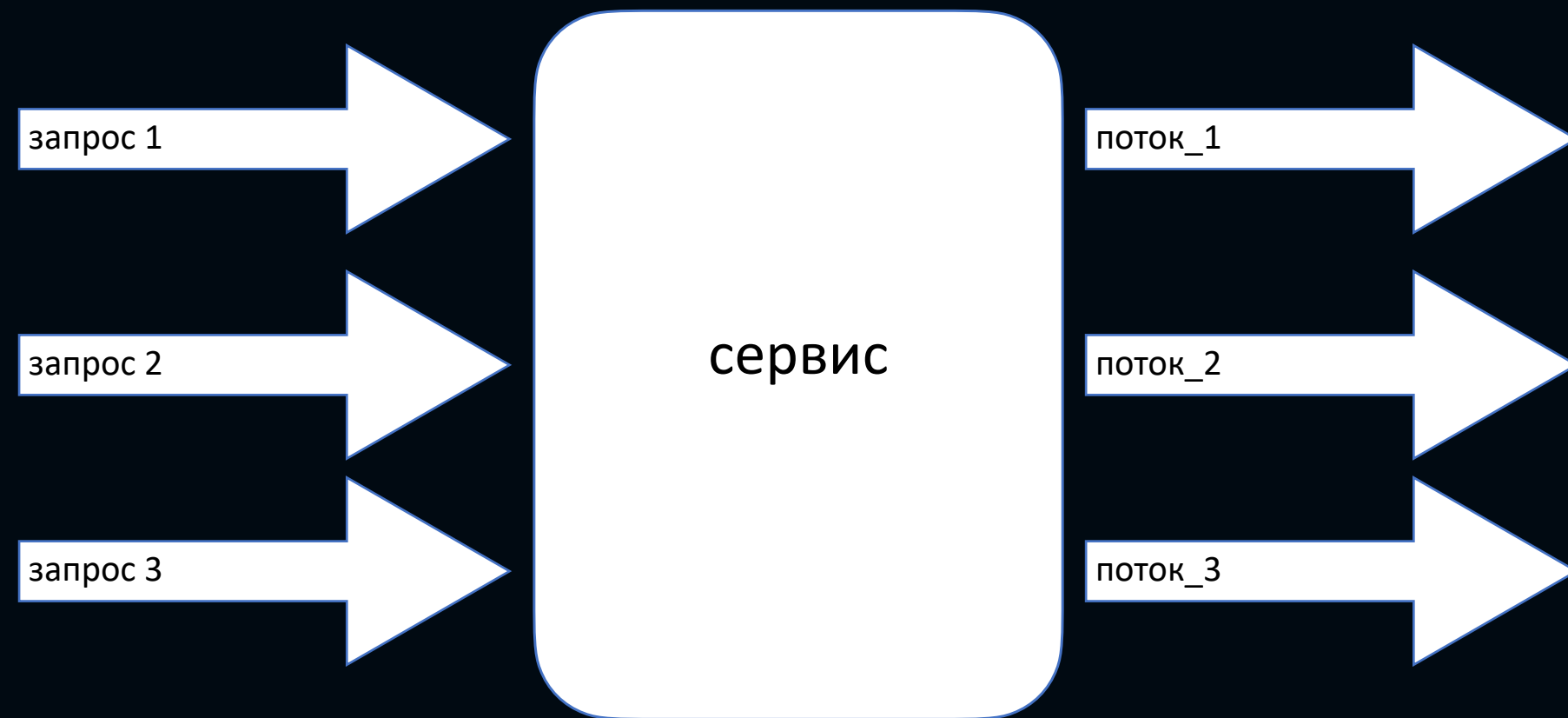
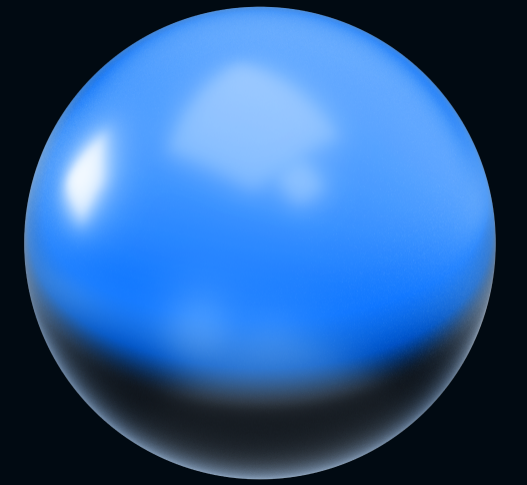
Пример проблемного сервиса



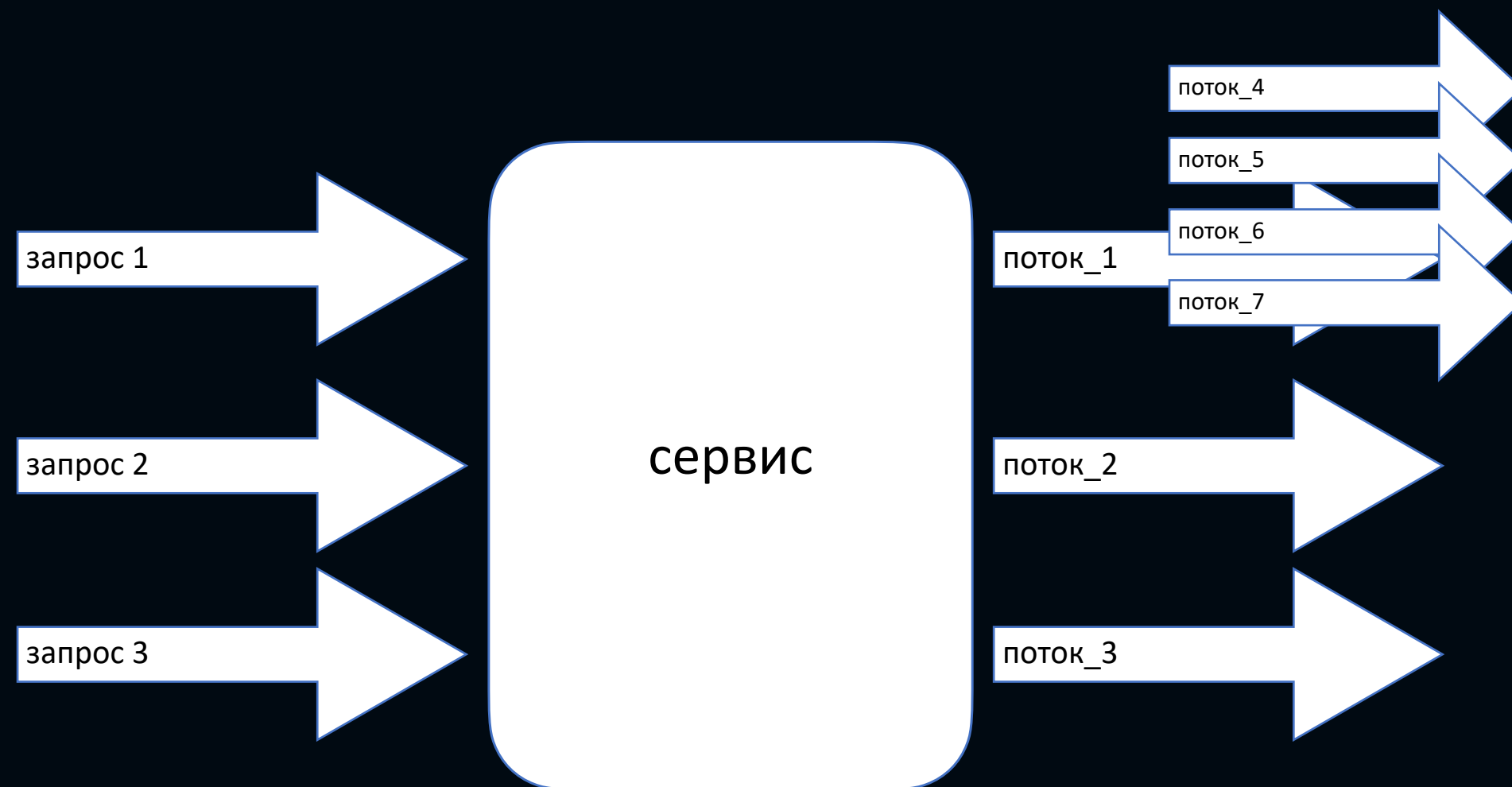
Пример проблемного сервиса



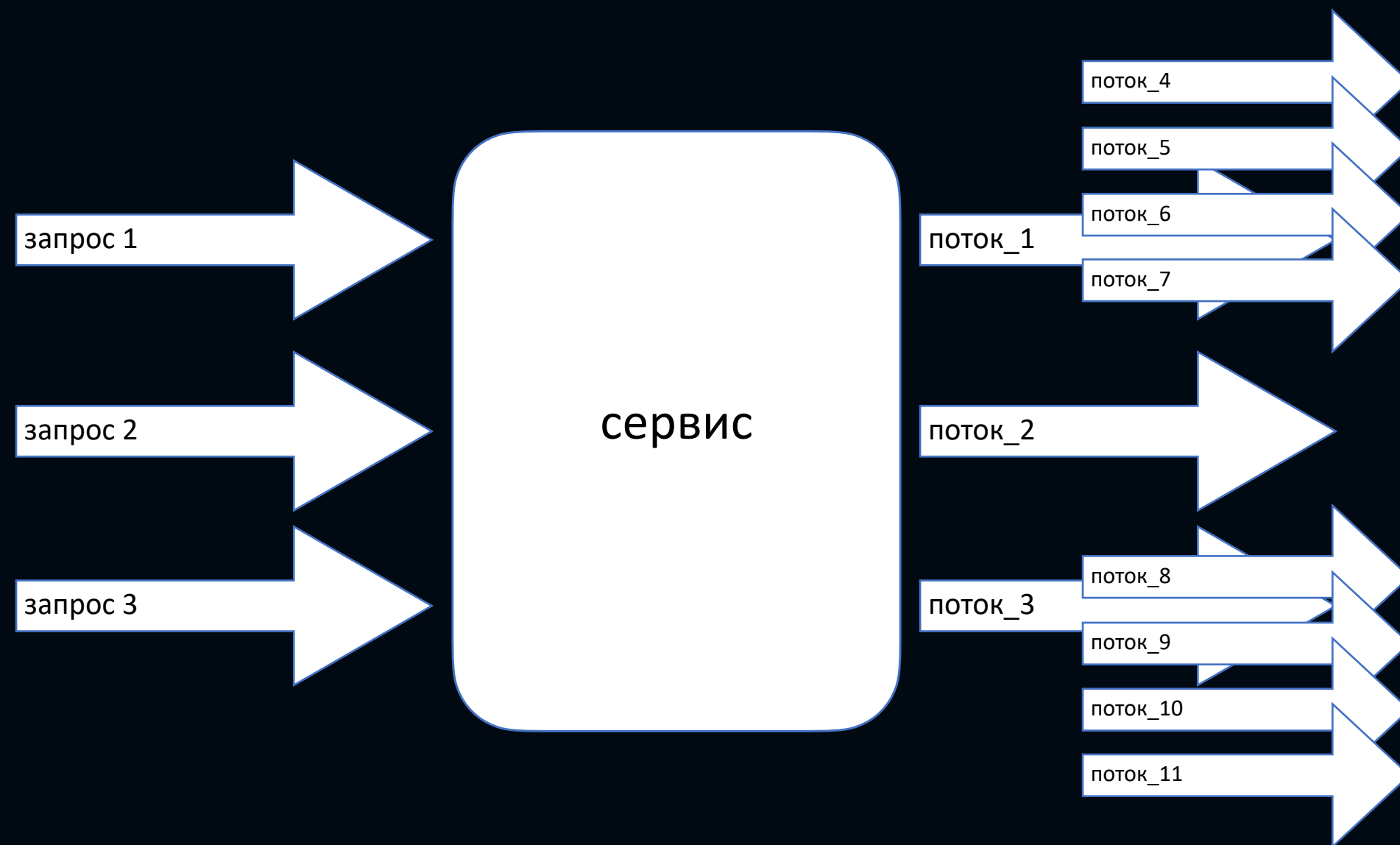
Пример проблемного сервиса



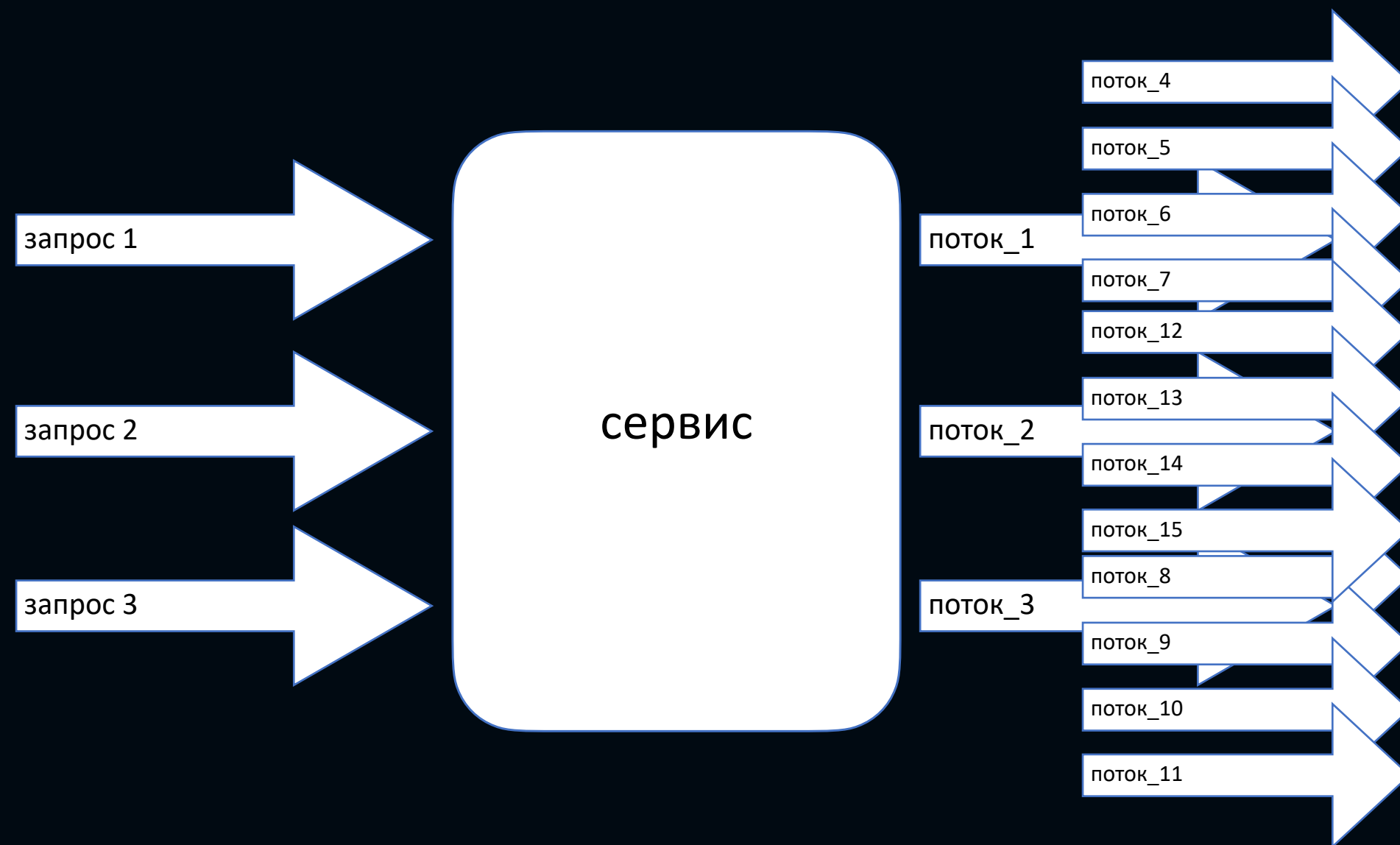
Пример проблемного сервиса



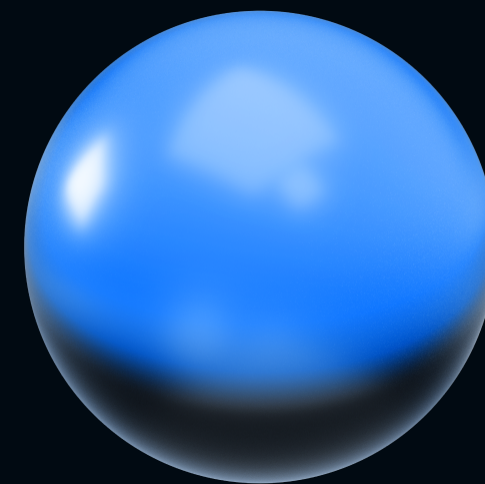
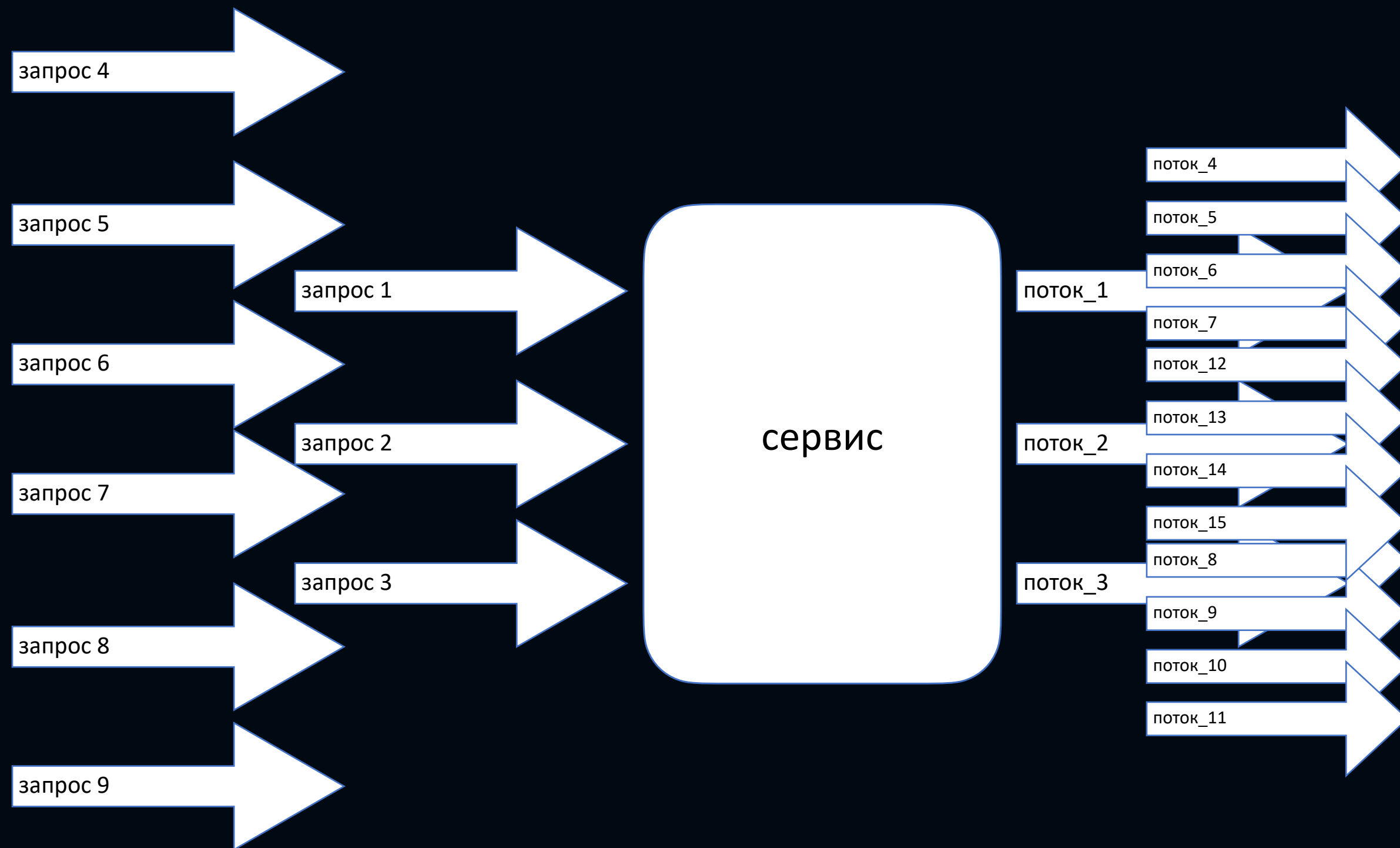
Пример проблемного сервиса



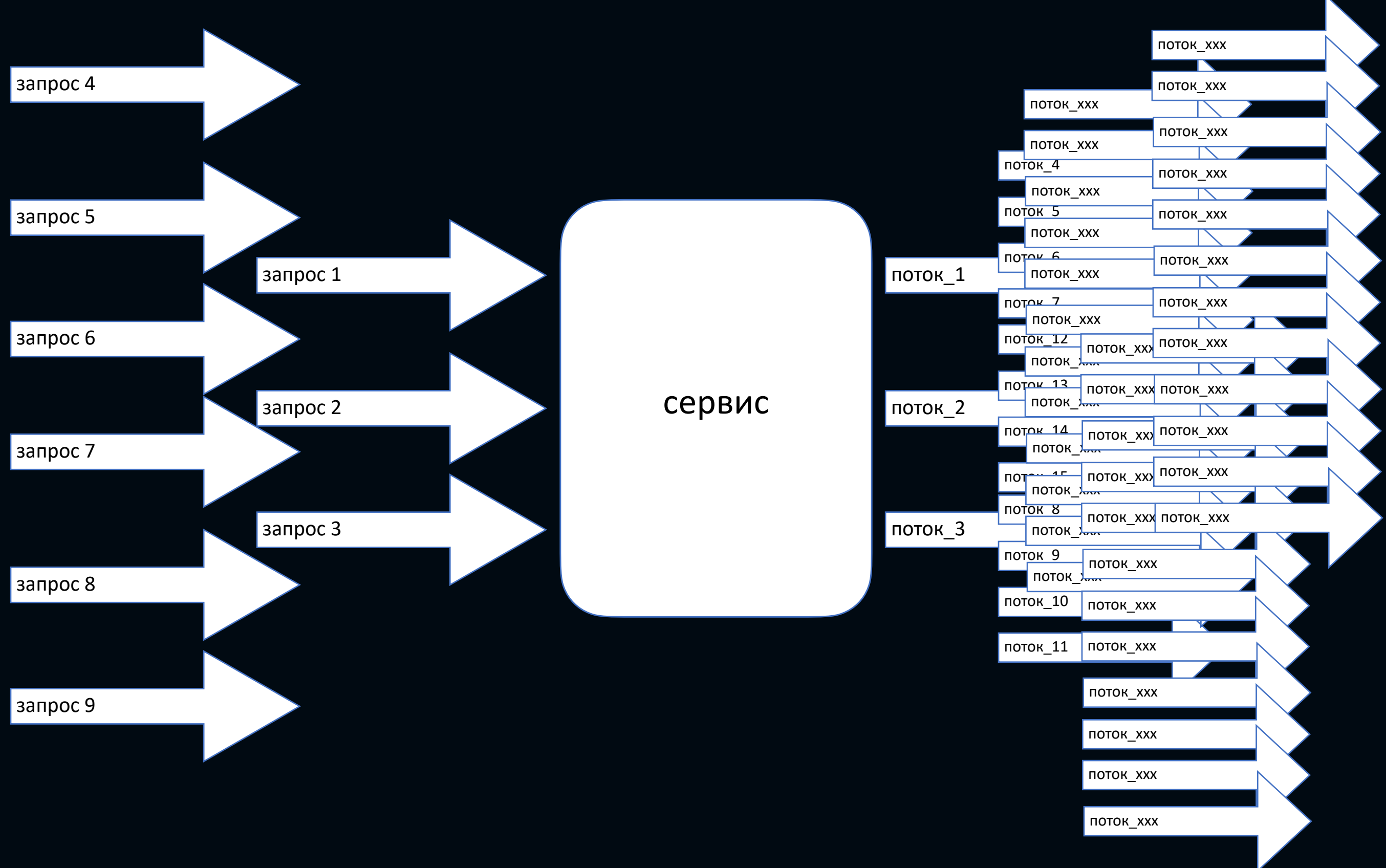
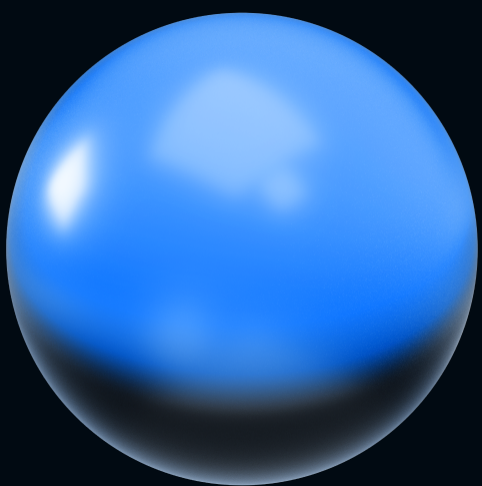
Пример проблемного сервиса



Пример проблемного сервиса

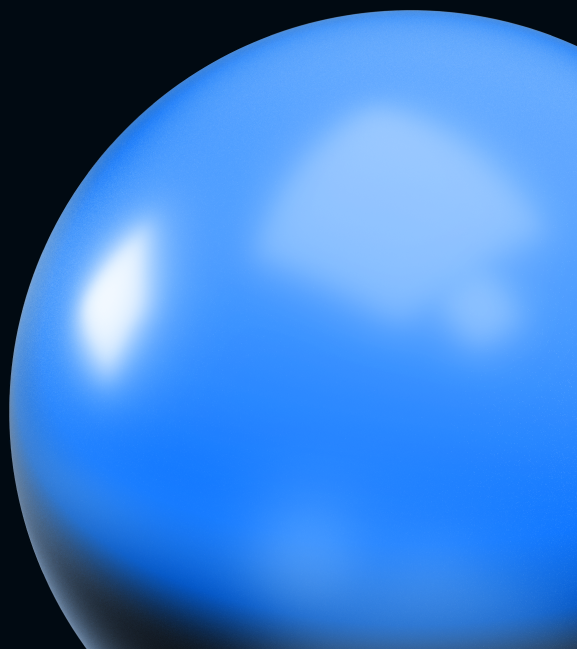
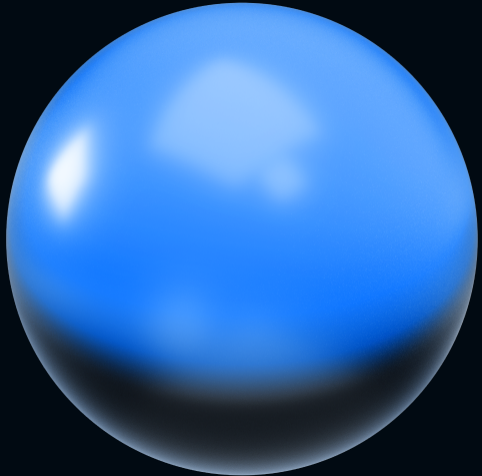
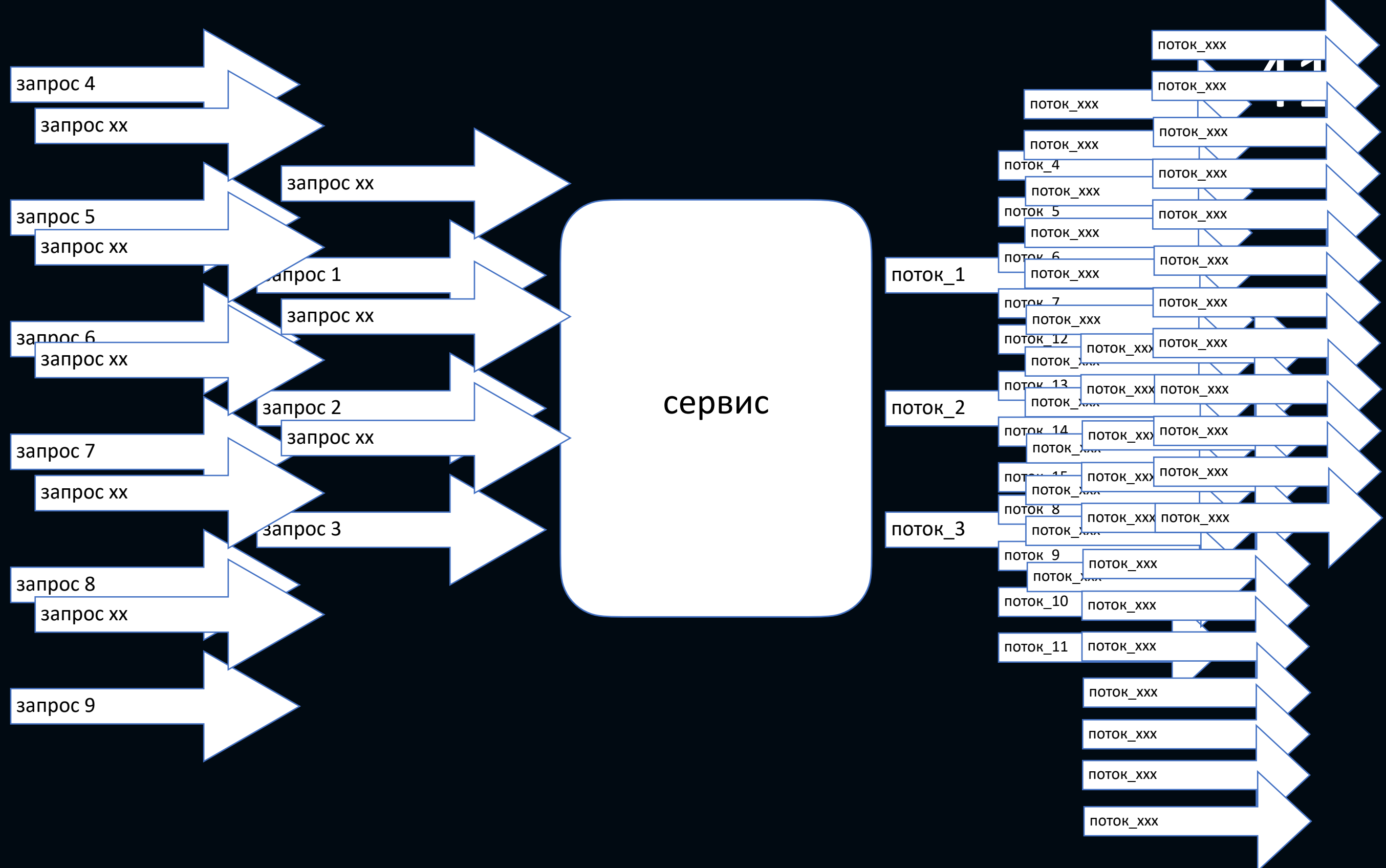


Пример проблемного сервиса

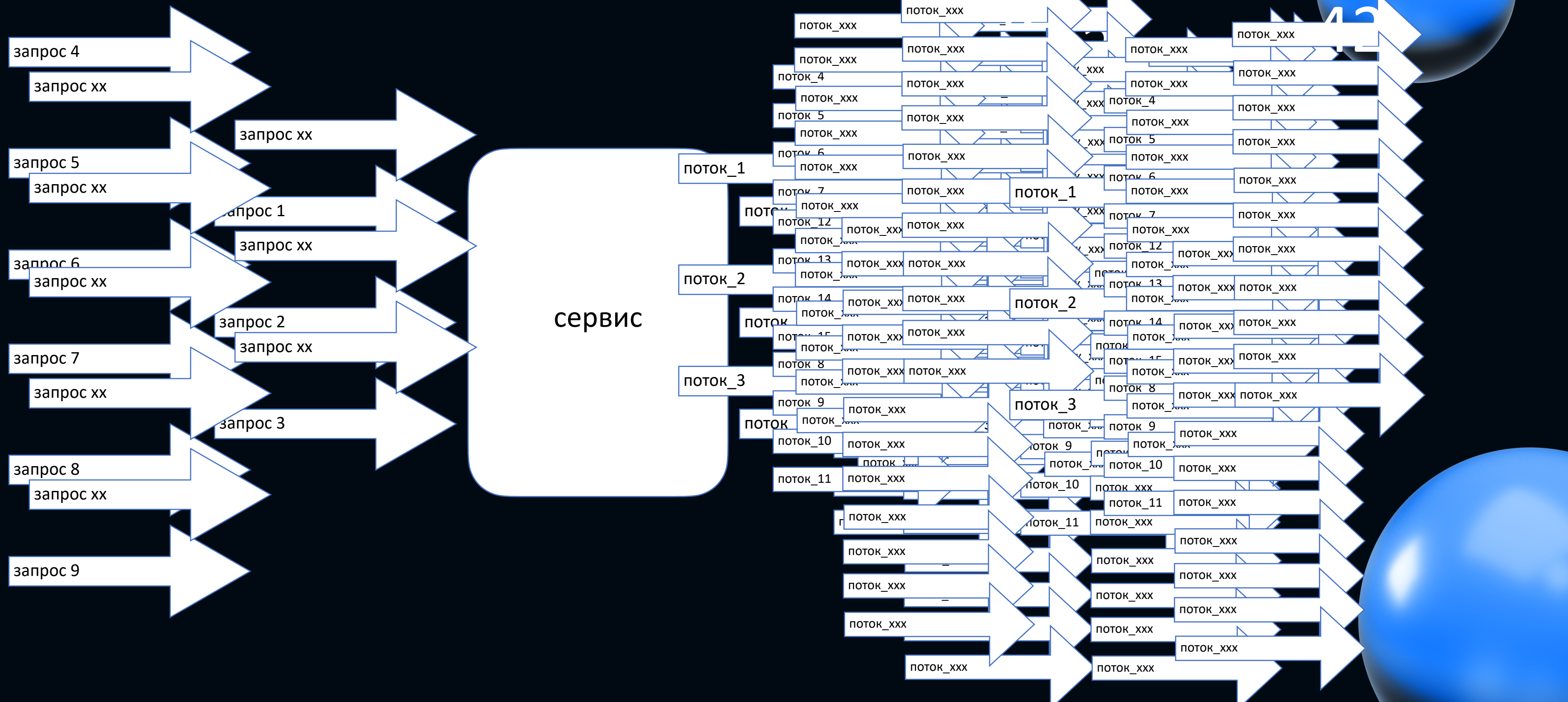


40

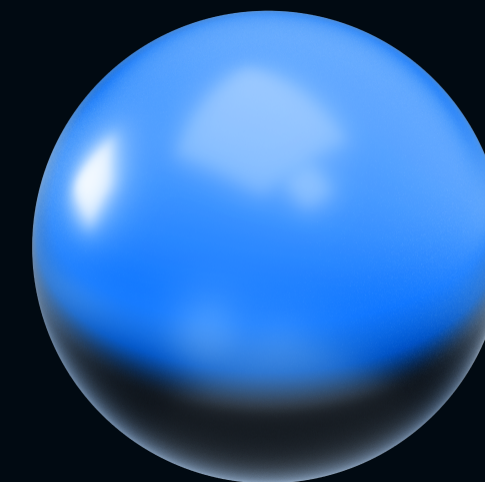
Пример проблемного сервиса



Пример проблемного сервиса

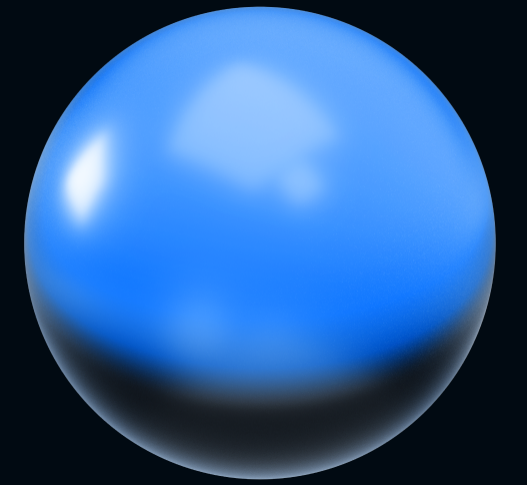


Как решали



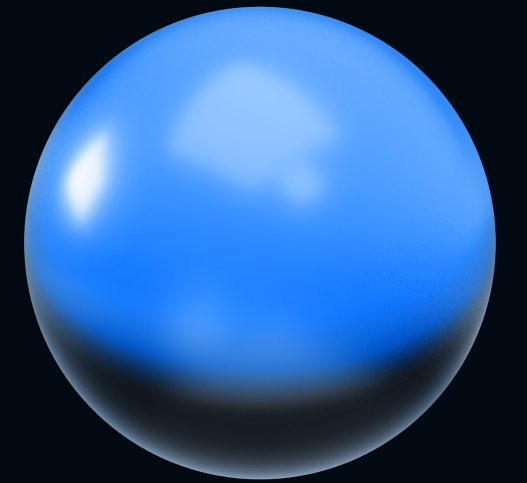
Как решали

- Увеличивали кол-во инстансов у сервиса



Как решали

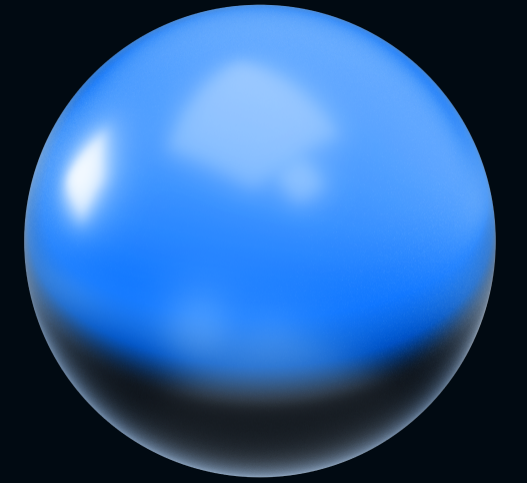
- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов



45

Как решали

- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса

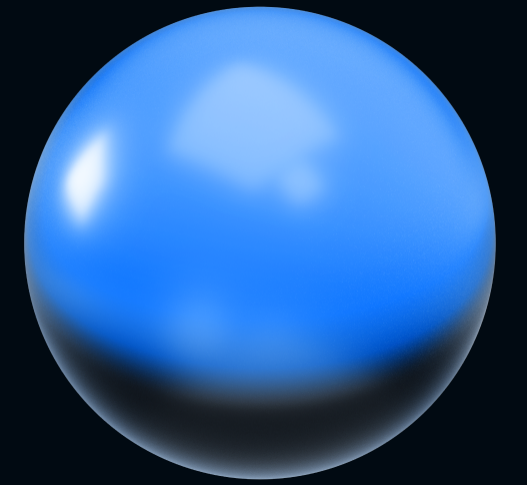


Как решали

- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса
- Увеличили лимит тредов на RHEL: ulimit -u и прочие

Как решали

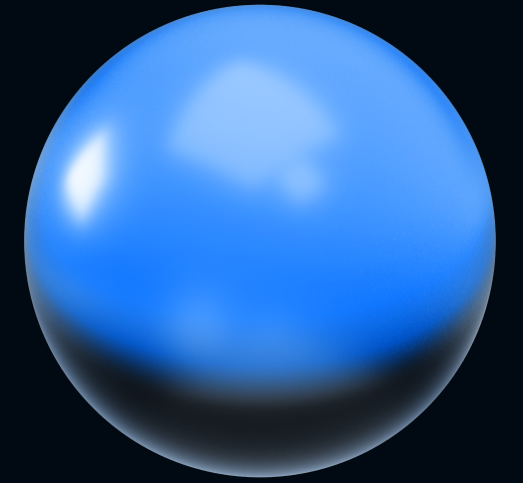
- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса
- Увеличили лимит тредов на RHEL: ulimit -u и прочие
- Сняли лимит тредов на докере: pids-max -1



Как решали

- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса
- Увеличили лимит тредов на RHEL: ulimit -u и прочие
- Сняли лимит тредов на докере: pids-max -1
- Поставили таймауты на клиенты к нижележащим сервисам (где можно)

Как решали

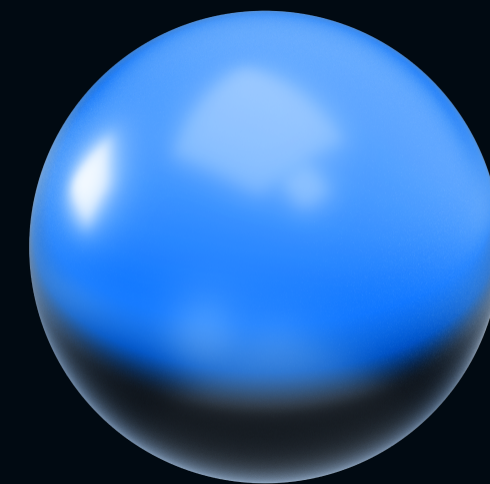


- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса
- Увеличили лимит тредов на RHEL: ulimit -u и прочие
- Сняли лимит тредов на докере: pids-max -1
- Поставили таймауты на клиенты к нижележащим сервисам (где можно)
- ... Увеличивали количество инстансов еще раз

Как решали

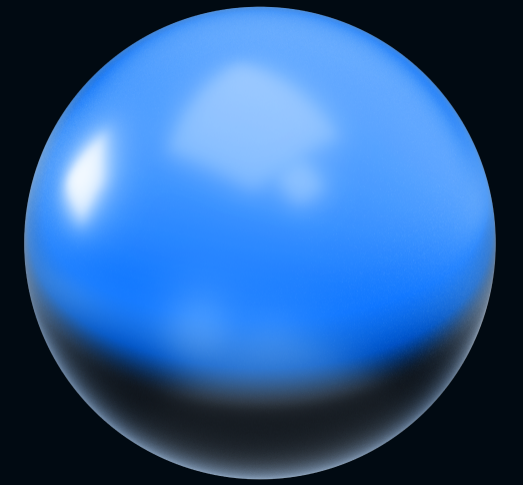
- Увеличивали кол-во инстансов у сервиса
- Накидывали сервису ресурсов
- Увеличили дефолтовый тредпул томката у сервиса
- Увеличили лимит тредов на RHEL: ulimit -u и прочие
- Сняли лимит тредов на докере: pids-max -1
- Поставили таймауты на клиенты к нижележащим сервисам (где можно)
- ... Увеличивали количество инстансов еще раз
- Mesos убивает контейнеры при использовании > 8 к тредов

**Стали думать
в сторону кода**



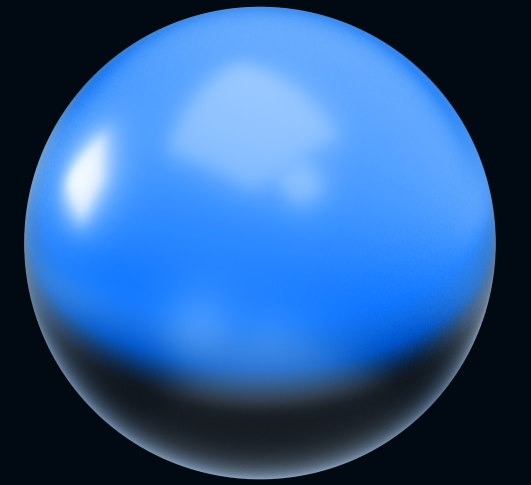
Стали думать в сторону кода

- «Проблемных» сервисов немного, и в них мало логики

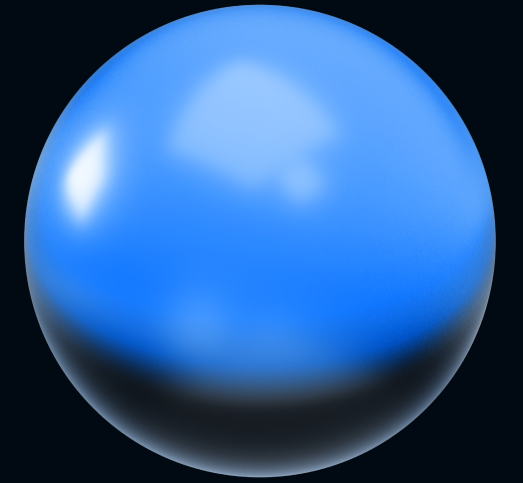


Стали думать в сторону кода

- «Проблемных» сервисов немного, и в них мало логики
- Реактивный подход может помочь в нашем случае

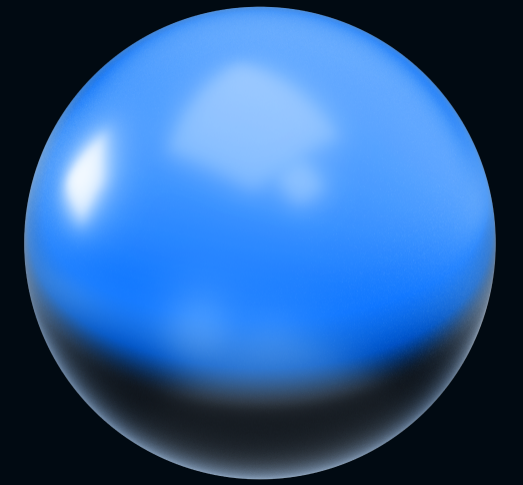


Стали думать в сторону кода



- «Проблемных» сервисов немного, и в них мало логики
- Реактивный подход может помочь в нашем случае
- Но наши кастомные трифтовые обработчики жестко завязаны на сервлеты

Стали думать в сторону кода




- «Проблемных» сервисов немного, и в них мало логики
- Реактивный подход может помочь в нашем случае
- Но наши кастомные трифтовые обработчики жестко завязаны на сервлеты
- webflux не умеет из коробки работать с apache thrift

google.com/search?q=webflux+thrift&oq=webflux+thrift&aqs=chrome..69i57j35i39j0i512l5j69i61.3018j0j7

Google webflux thrift


All Videos Images Maps Books More Tools

About 28,400 results (0.27 seconds)

 **ARMERIA.dev**
https://armeria.dev › docs › advanced-spring-webflux-...


Using Armeria with Spring WebFlux

Rich support for Apache **Thrift** and gRPC, including the fancy web console that enables you to send **Thrift** and gRPC requests from a web browser · Ability to run ...
You've visited this page 2 times. Last visit: 3/15/23

 **GitHub**
https://github.com › spring-framework › issues

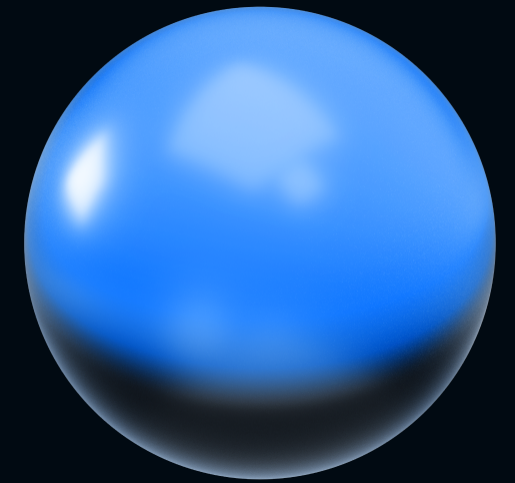
Add Thrift RPC support to WebFlux [SPR-16466] #21011

Feb 4, 2018 — I concerned into building high-performance, reactive microservices using Spring **Webflux** and Spring Boot 2.0. Now one possible way of ...
You visited this page on 3/13/23.

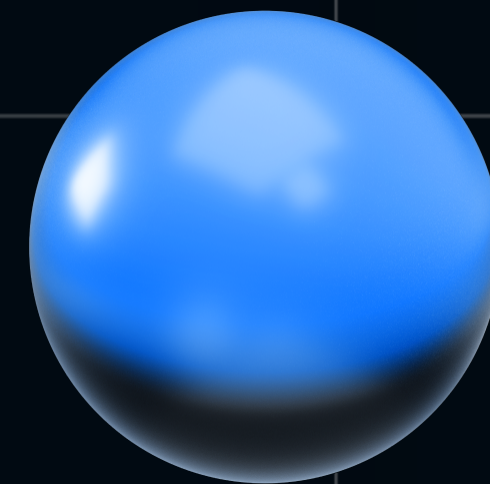
 **jpoint.ru**
https://jpoint.ru › talks · Translate this page

Доклад Реактивщина с Apache Thrift + Project Armeria

Реактивщина с Apache **Thrift** + Project Armeria: как **WebFlux** прикручивали без REST ·
Почему в данной ситуации масштабирование не помогает? · Чем поможет project ...



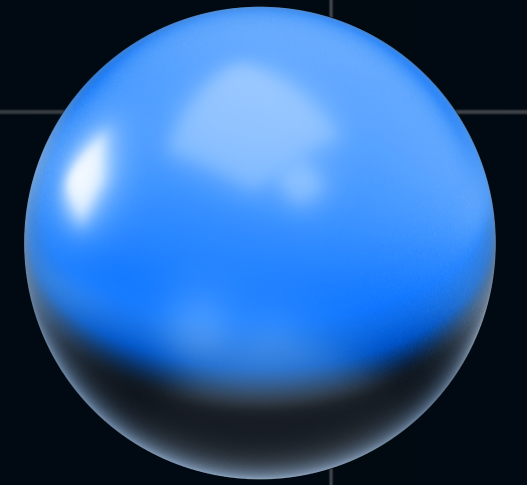
**Подшло решение: project
armeria**



58

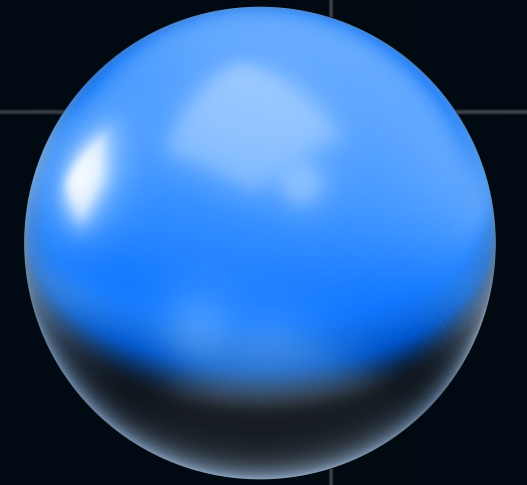
Подшло решение: project armeria

- Фреймворк: http-сервер и много обвязок для микросервисов (логирование, трейсинг, метрики...)



Подшло решение: project armeria

- Фреймворк: http-сервер и много обвязок для микросервисов (логирование, трейсинг, метрики...)
- Умеет в thrift, gRPC, REST, статичные файлы, и хостить все это на одном порту



Подшло решение: project armeria

- Фреймворк: http-сервер и много обвязок для микросервисов (логирование, трейсинг, метрики...)
- Умеет в thrift, gRPC, REST, статичные файлы, и хостить все это на одном порту
- Асинхронный. От авторов netty, работает на netty

Подшло решение: project armeria

- Фреймворк: http-сервер и много обвязок для микросервисов (логирование, трейсинг, метрики...)
- Умеет в thrift, gRPC, REST, статичные файлы, и хостить все это на одном порту
- Асинхронный. От авторов netty, работает на netty
- Есть спринговый стартер, легко интегрируется со спрингом

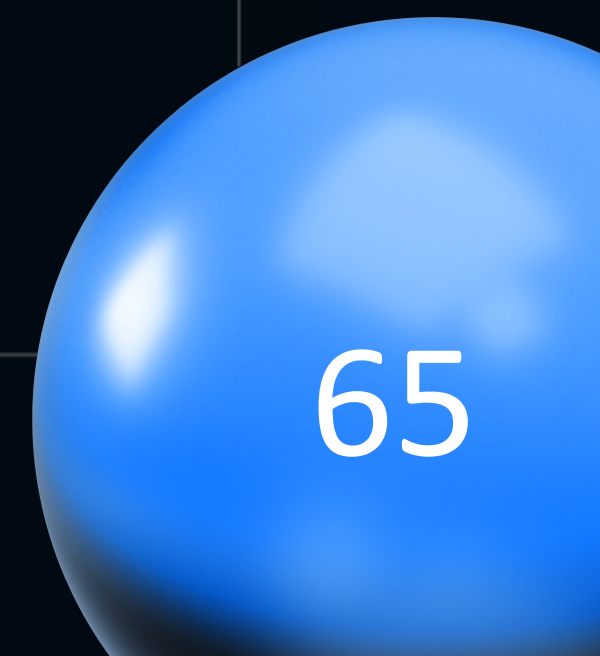
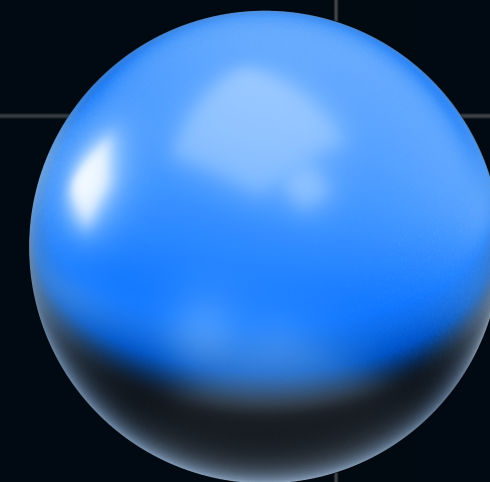
Подшло решение: project armeria

- Фреймворк: http-сервер и много обвязок для микросервисов (логирование, трейсинг, метрики...)
- Умеет в thrift, gRPC, REST, статичные файлы, и хостить все это на одном порту
- Асинхронный. От авторов netty, работает на netty
- Есть спринговый стартер, легко интегрируется со спрингом
- HTTP/2, https+http на одном порту, имеет встроенный аналог swagger ui...

Подшло решение: project armeria

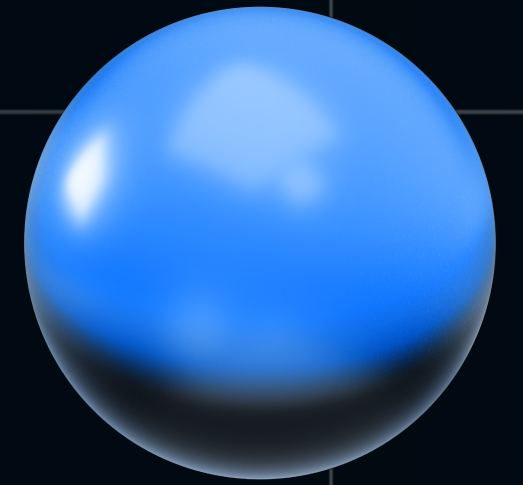
- Фреймворк: http-сервер и много обвязок для микросервисов (**логирование**, **трейсинг**, метрики...)
- **Умеет в thrift**, gRPC, REST, статичные файлы, и хостить все это на одном порту
- Асинхронный. От авторов netty, **работает на netty**
- Есть спринговый стартер, **легко интегрируется со спрингом**
- HTTP/2, https+http на одном порту, имеет встроенный аналог swagger ui...

Было/стало



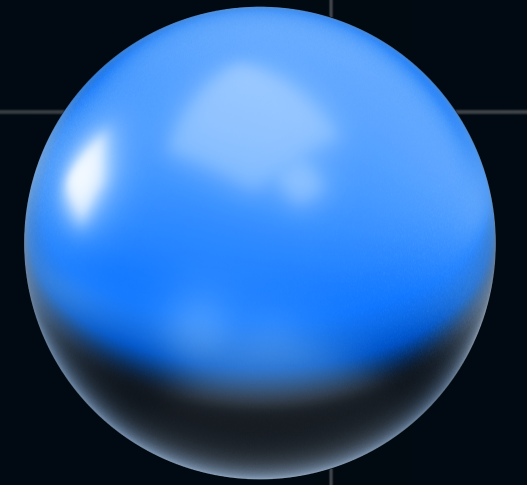
Было/стало

- Базовый спринговый стартер:
 - был `spring-boot-starter`
 - стал `armeria-spring-boot2-webflux-starter`



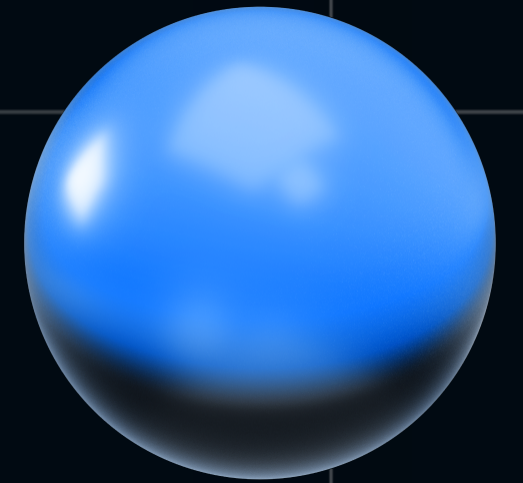
Было/стало

- Базовый спринговый стартер:
 - был `spring-boot-starter`
 - стал `armeria-spring-boot2-webflux-starter`
- Http-сервер
 - был `tomcat`
 - стал `netty`

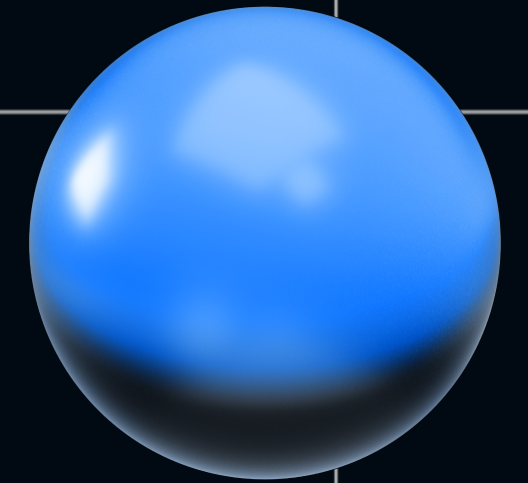


Было/стало

- Базовый спринговый стартер:
 - был `spring-boot-starter`
 - стал `armeria-spring-boot2-webflux-starter`
- Http-сервер
 - был `tomcat`
 - стал `netty`
- Трифтовый обработчик
 - был `aatarasoff/spring-thrift-starter`
 - стал `armeria-thrift`



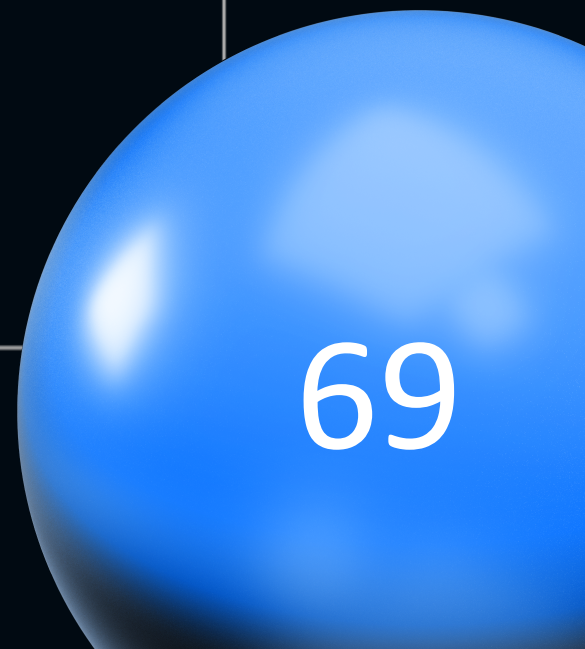
Примеры кода: контроллер



```
@Slf4j
@Validated
@ThriftController(path = "/tapi/banners")
@RequiredArgsConstructor
public class DashboardBannersController implements DashboardBannersApiService.AsyncIface {

    private final DashboardBannersService dashboardBannersService;

    @Override
    public void getDashboardBanners(
        @ValidUserData UserData userData,
        @OrganizationId String organizationId,
        @NotNull TDashboardBannersRequestInfo requestInfo,
        AsyncMethodCallback<TDashboardBannersResponse> resultHandler) {
        dashboardBannersService.getDashboardBanners(userData, organizationId, requestInfo)
            .subscribe(resultHandler::onComplete);
    }
}
```



Примеры кода: сервис

```
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Service
@RequiredArgsConstructor
public class DashboardBannersService {

    public Mono<TDashboardBannersResponse> getDashboardBanners(
        UserData userData,
        String orgId,
        TDashboardBannersRequestInfo requestInfo) {
        return Flux.fromIterable(bannersServiceList) Flux<BannersService>
            .flatMap(bannerService -> bannerService.getBanners(userData, orgId, requestInfo))
            .flatMap(Flux::fromIterable) Flux<DashboardBannerDTO>
            .collectList() Mono<List<DashboardBannerDTO>>
            .map(mapper::toThrift);
    }
}
```

Примеры кода: клиент

```
import com.linecorp.armeria.common.thrift.ThriftFuture;
import reactor.core.publisher.Mono;
import org.springframework.stereotype.Component;

@Component
@RequiredArgsConstructor
public class MaintenanceApiGateway {

    private final MaintenanceApiService.AsyncIface maintenanceInfoClient;

    @SneakyThrows
    public Mono<MaintenanceInfo> getMaintenanceInfo(String os) {
        var future = new ThriftFuture<MaintenanceInfo>();
        maintenanceInfoClient.getMaintenanceInformation(os, future);
        return Mono.fromFuture(future);
    }
}
```


Примеры кода: клиент

```
import com.linecorp.armeria.common.thrift.ThriftFuture;
import reactor.core.publisher.Mono;
import org.springframework.stereotype.Component;

@Component
@RequiredArgsConstructor
public class MaintenanceApiGateway {

    private final MaintenanceApiService.AsyncIface maintenanceInfoClient;

    @SneakyThrows
    public Mono<MaintenanceInfo> getMaintenanceInfo(String os) {
        var future = new ThriftFuture<MaintenanceInfo>();
        maintenanceInfoClient.getMaintenanceInformation(os, future);
        return Mono.fromFuture(future);
    }
}
```

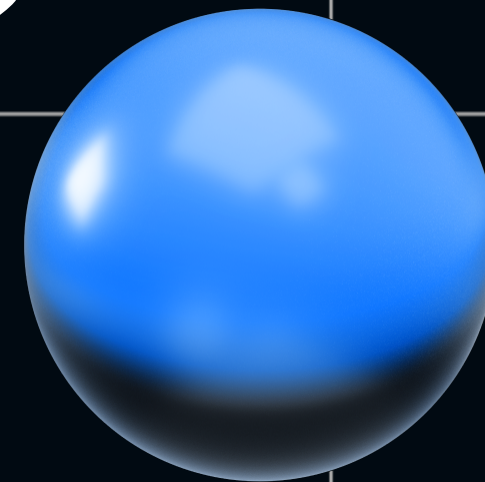
Armeria
ThriftFuture

Apache thrift

(принимает future
и кладет туда
результат)

Reactor
Mono
(Принимает future)

Какова именно роль Armeria?



Какова именно роль Armeria?

- Реактивные трифтовые обработчики и клиенты

Какова именно роль Armeria?

- Реактивные трифтовые обработчики и клиенты
(К webflux'у трифт прикрутить не очень просто)

Какова именно роль Armeria?

- Реактивные трифтовые обработчики и клиенты
(К webflux'у трифт приколотить не очень просто)
- Подменяет веб-сервер в webflux, с netty.. на netty

Какова именно роль Armeria?

- Реактивные трифтовые обработчики и клиенты
(К webflux'у трифт приколотить не очень просто)
- Подменяет веб-сервер в webflux, с netty.. на netty
- Берет на себя request dispatching

Какова именно роль Armeria?

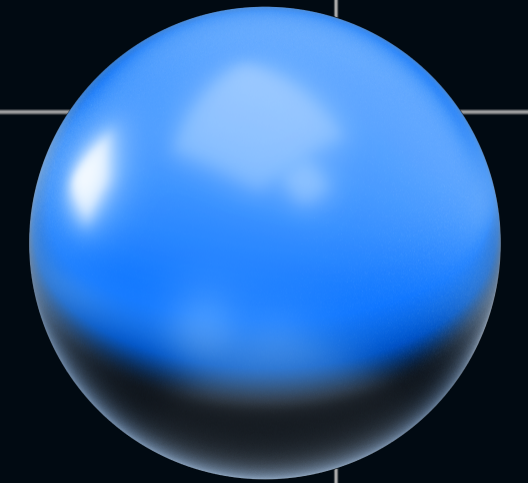
- Реактивные трифтовые обработчики и клиенты
(К webflux'у трифт приколотить не очень просто)
- Подменяет веб-сервер в webflux, с netty.. на netty
- Берет на себя request dispatching
- Управляет обработкой запросов: thread pool, event loop

Примеры кода: Kotlin

```
import com.linecorp.armeria.server.ServiceRequestContext
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.asCoroutineDispatcher
import org.springframework.stereotype.Component

@Component
class CoroutinesHandler {
    fun coroutineScope() = CoroutineScope(
        ServiceRequestContext
            .current()
            .eventLoop()
            .asCoroutineDispatcher()
    )
}
```

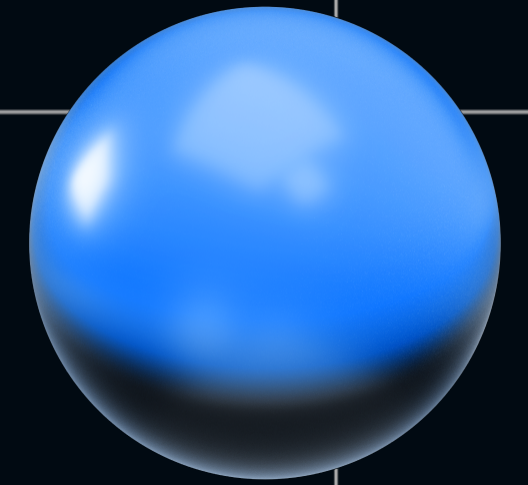
Примеры кода: Kotlin



```
@ThriftController(path = "/tapi/front")
class FrontController(
    private val dashboardService: DashboardServiceGateway,
    private val featuresService: FeaturesServiceGateway,
    private val profilesApiService: ProfilesApiServiceGateway,
    private val permissionsApiService: PermissionsApiServiceGateway,
    private val coroutinesHandler: CoroutinesHandler
) : FrontService.AsyncIface {

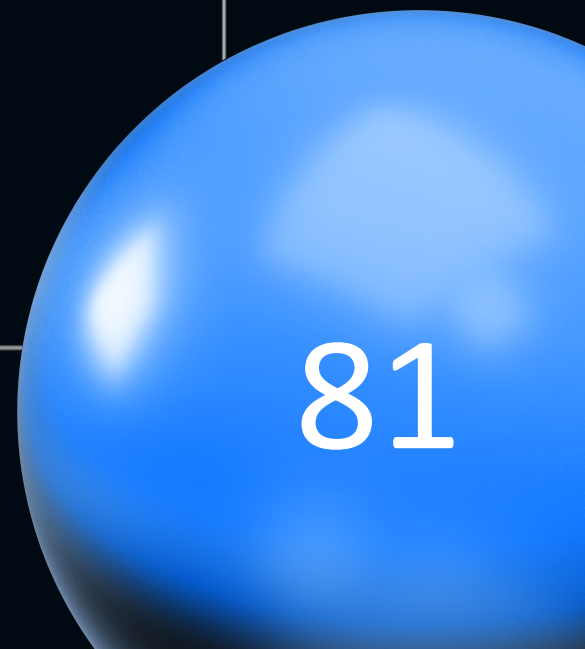
    override fun getPostAuthData(userData: UserData,
                                   resultHandler: AsyncMethodCallback<TPostAuthDataResponse>) {
        coroutinesHandler.coroutineScope().future { this: CoroutineScope
            TPostAuthDataResponse().apply { this: TPostAuthDataResponse
                dashboardResponse = getDashboardWithoutTasks(userData)
                features = getCustomerFeatures(userData)
                profile = getProfile(userData)
                isNeedsToCheckAuthorizedPersonDocuments = getUserClientBlock(userData)
            }
        }
    }
}
```


Примеры кода: Kotlin

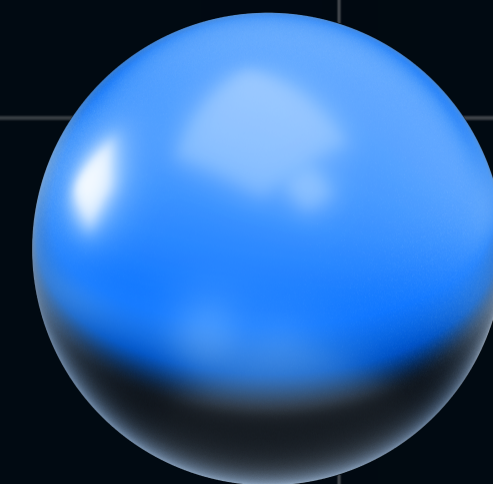


```
import org.springframework.stereotype.Component
import com.linecorp.armeria.common.thrift.ThriftFuture
import kotlinx.coroutines.future.asDeferred

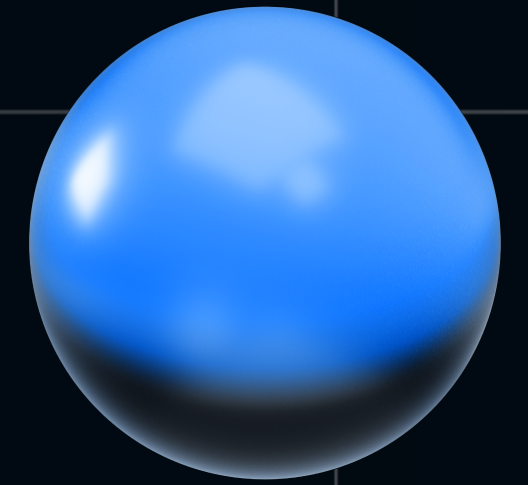
@Component
class DashboardServiceGateway(
    private val client: DashboardService.AsyncIface
) {
    suspend fun getDashboard(userData: UserData,
                             projection: OrganizationInfoProjection): DashboardInfoResponse {
        return ThriftFuture<DashboardInfoResponse>().also { future ->
            client.getDashboardWithoutTasks(
                userData,
                projection,
                future
            )
        }.asDeferred().await()
    }
}
```



**Проблема 0: код сложнее
поддерживать**

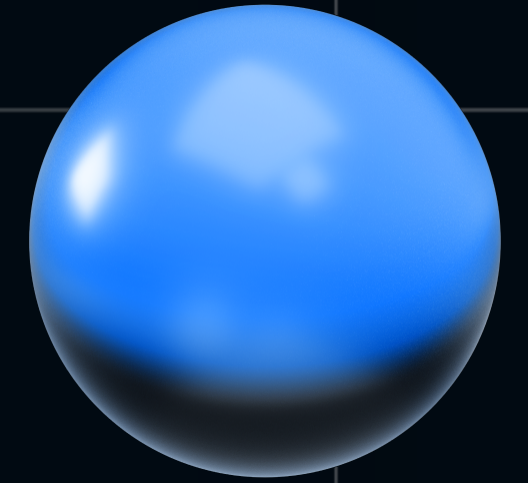


Проблема 0: код сложнее поддерживать: было



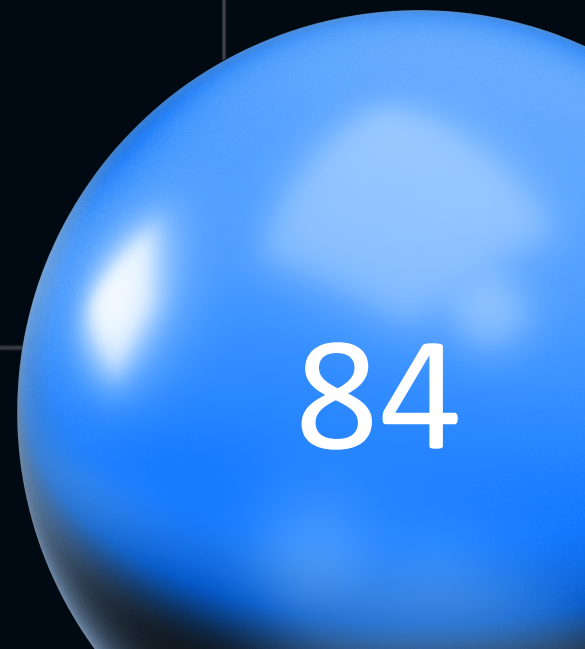
```
public TDashboardBannersResponse getDashboardBanners(  
    UserData userData,  
    String orgId,  
    TDashboardBannersRequestInfo requestInfo) {  
    return mapper.toThrift(  
        bannersServiceList  
            .stream() Stream<BannersService>  
            .flatMap(bannersService -> bannersService.getBanners(userData, orgId, requestInfo)  
                .stream()) Stream<DashboardBannerDTO>  
            .collect(Collectors.toList()));  
}
```

Проблема 0: код сложнее поддерживать: стало

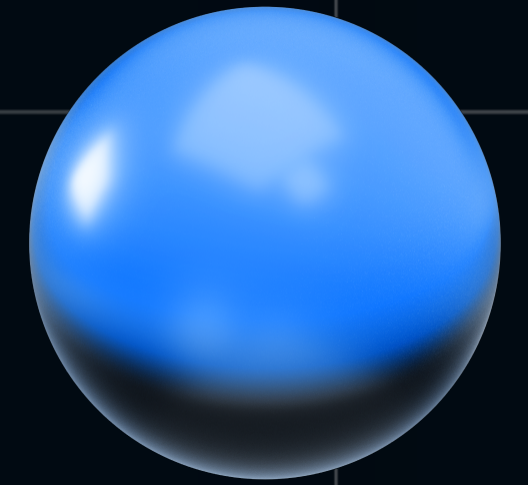


```
public TDashboardBannersResponse getDashboardBanners(  
    UserData userData,  
    String orgId,  
    TDashboardBannersRequestInfo requestInfo) {  
    return mapper.toThrift(  
        bannersServiceList  
            .stream() Stream<BannersService>  
            .flatMap(bannersService -> bannersService.getBanners(userData, orgId, requestInfo)  
                .stream()) Stream<DashboardBannerDTO>  
            .collect(Collectors.toList()));  
}
```

```
public Mono<TDashboardBannersResponse> getDashboardBanners(  
    UserData userData,  
    String orgId,  
    TDashboardBannersRequestInfo requestInfo) {  
    return Flux.fromIterable(bannersServiceList) Flux<BannersService>  
        .flatMap(bannerService -> bannerService.getBanners(userData, orgId, requestInfo))  
        .flatMap(Flux::fromIterable) Flux<DashboardBannerDTO>  
        .collectList() Mono<List<DashboardBannerDTO>>  
        .map(mapper::toThrift);  
}
```



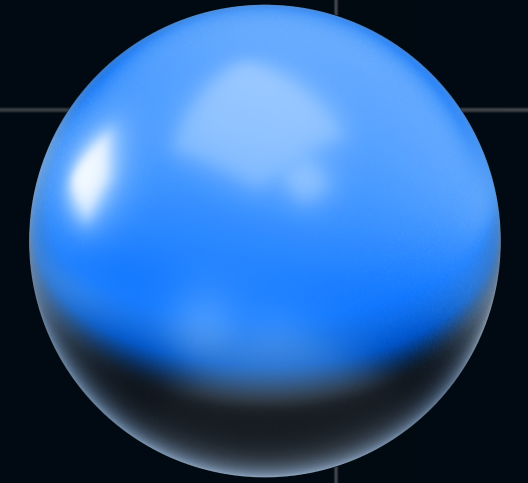
Проблема 0: код сложнее поддерживать: было



```
public List<TaskTransaction> getTasksByOrganization(UserData userData,
                                                    String organizationId,
                                                    TaskType taskType,
                                                    TasksTransactionFilter filter) {

    return transactionSourceContainer
        .getTransactions(userData, organizationId, getCollectCondition(taskType, filter)).stream()
        .filter(taskTransaction -> filter.getTypes()
            .contains(taskTransaction.getTransactionType()))
        .collect(Collectors.toList());
}
```


Проблема 0: код сложнее поддерживать: стало



```
public List<TaskTransaction> getTasksByOrganization(UserData userData,
                                                    String organizationId,
                                                    TaskType taskType,
                                                    TasksTransactionFilter filter) {

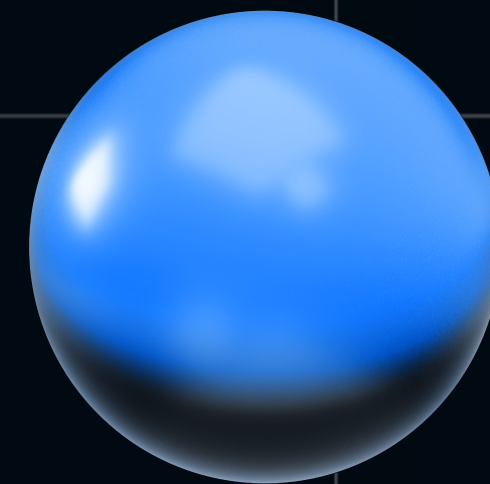
    return transactionSourceContainer
        .getTransactions(userData, organizationId, getCollectCondition(taskType, filter)).stream()
        .filter(taskTransaction -> filter.getTypes()
            .contains(taskTransaction.getTransactionType()))
        .collect(Collectors.toList());
}
```

```
public Mono<List<TaskTransaction>> getTasksByOrganization(UserData userData,
                                                         String organizationId,
                                                         TaskType taskType,
                                                         TasksTransactionFilter filter) {

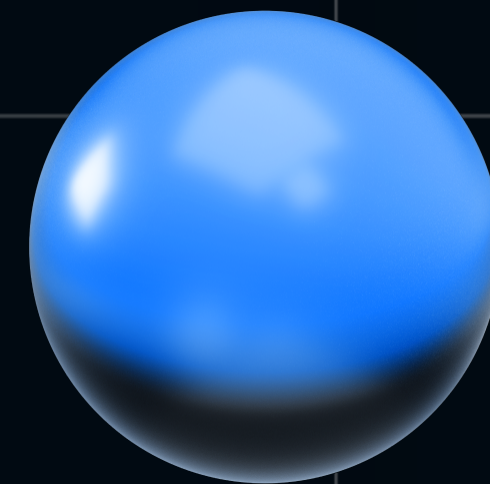
    return getTransactionsProvider().getTransactions(userData, organizationId, getCollectCondition(taskType, filter))
        .map(taskTransactions ->
            taskTransactions.stream()
                .filter(transaction -> filter.getTypes().contains(transaction.getTransactionType()))
                .collect(Collectors.toList())
        );
}
```



**Проблема 0: код сложнее
поддерживать (не всегда)**

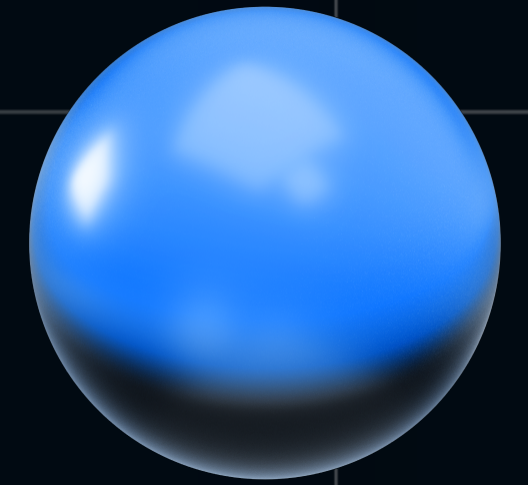


Проблема 0: код сложнее поддерживать (не всегда)



```
fun getPostAuthData(userData: UserData): TPostAuthDataResponse =  
    Observable  
        .zip(  
            getDashboardWithoutTasks(userData),  
            getCustomerFeatures(userData),  
            getProfile(userData),  
            getUserClientBlock(userData),  
            mapper::toResponse  
        )  
        .toBlocking()  
        .single()
```

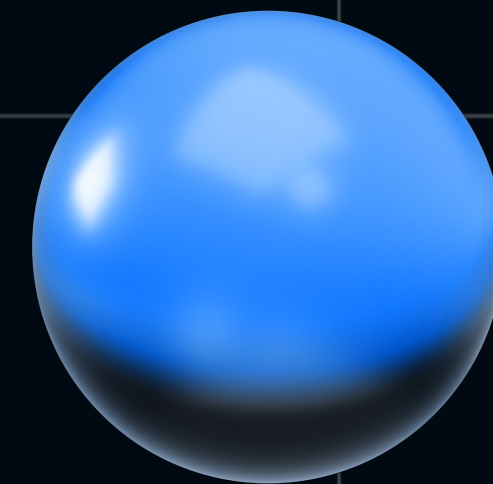
Проблема 0: код сложнее поддерживать (не всегда)



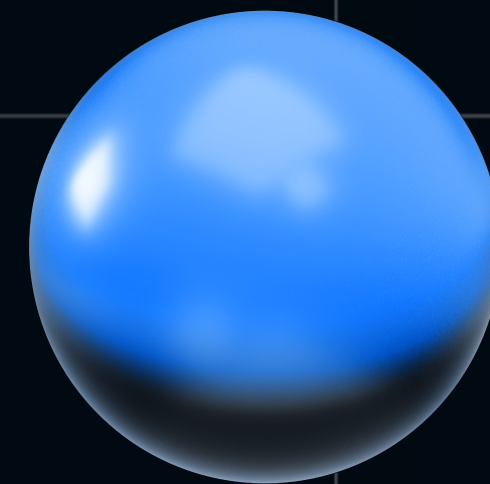
```
fun getPostAuthData(userData: UserData): TPostAuthDataResponse =  
    Observable  
        .zip(  
            getDashboardWithoutTasks(userData),  
            getCustomerFeatures(userData),  
            getProfile(userData),  
            getUserClientBlock(userData),  
            mapper::toResponse  
        )  
        .toBlocking()  
        .single()
```

```
override fun getPostAuthData(userData: UserData,  
    resultHandler: AsyncMethodCallback<TPostAuthDataResponse>) {  
    coroutinesHandler.coroutineScope().future { this: CoroutineScope  
        TPostAuthDataResponse().apply { this: TPostAuthDataResponse  
            dashboardResponse = getDashboardWithoutTasks(userData)  
            features = getCustomerFeatures(userData)  
            profile = getProfile(userData)  
            isNeedsToCheckAuthorizedPersonDocuments =  
                getUserClientBlock(userData)  
        }  
    }  
}
```


Проблема 1: пришлось адаптировать библиотеки



Проблема 1: пришлось адаптировать библиотеки



- Логирование (кастомные штуки, типа айдишник юзера в отдельном поле в эластике) - перенос из MDC в контекст запроса

Проблема 1: пришлось адаптировать библиотеки: было

```
Used as method interceptor to add messages into Spans.  
16 public class UserIdSpanRecorderAdvice implements MethodBeforeAdvice, ThriftAdvice {  
17  
18     private static final String USER_ID_NAME = "user_id";  
19  
20     @Override  
21     public void before(Method method, Object[] args, Object target) {  
22         logUserId(getUserIdFromArgs(args, defaultUserId: "---"));  
23     }  
24  
25     private void logUserId(String userId) {  
26         MDC.put(USER_ID_NAME, userId);  
27     }  
}
```

Проблема 1: пришлось адаптировать библиотеки: было



```
Used as method interceptor to add messages into Spans.  
16 public class UserIdSpanRecorderAdvice implements MethodBeforeAdvice, ThriftAdvice {  
17  
18     private static final String USER_ID_NAME = "user_id";  
19  
20     @Override  
21     public void before(Method method, Object[] args, Object target) {  
22         logUserId(getUserIdFromArgs(args, defaultUserId: "---"));  
23     }  
24  
25     private void logUserId(String userId) {  
26         MDC.put(USER_ID_NAME, userId);  
27     }  
}
```

```
12 public class DurationSpanRecorderBeforeAdvice implements MethodBeforeAdvice, ThriftAdvice {  
13  
14     @Override  
15     public void before(Method method, Object[] args, Object target) throws Throwable {  
16         long startTime = currentTimeMillis();  
17         MDC.put(PROCESSING_START_TAG, String.valueOf(startTime));  
18     }  
19 }  
20
```



Проблема 1: пришлось адаптировать библиотеки: стало

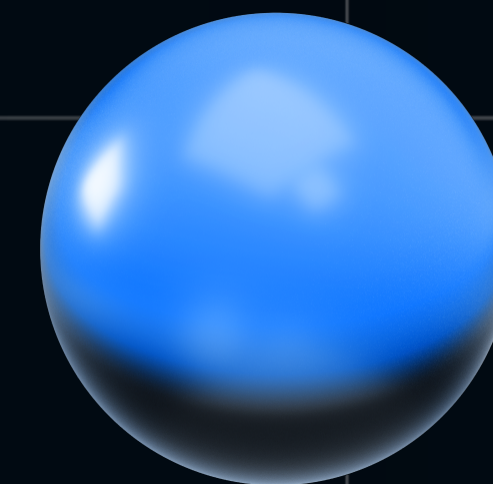


```
27 import brave.propagation.TraceContext;
28 import com.linecorp.armeria.common.logging.RequestScopedMdc;
29
30 public class LoggingServiceDecorator extends SimpleDecoratingRpcService {
31     public void putTagsToMDC(RequestContext ctx, @NonNull RpcRequest content, String phase) {
32         final String method = content.method();
33         final List<Object> params = content.params();
34         TraceContext traceContext = TraceContextUtil.traceContext(ctx);
35         if (traceContext != null) {
36             Map<String, String> mdcProperties = new HashMap<>();
37             mdcProperties.put(PHASE_TAG, phase);
38             mdcProperties.put(TRACEID, traceContext.traceIdString());
39             mdcProperties.put(API_METHOD, method);
40             mdcProperties.put(USER_ID, decoratorUtil.getUserIdFromArgs(params, UNKNOWN_USER));
41             RequestScopedMdc.putAll(ctx, mdcProperties);
42         }
43     }
```



Проблема 1: пришлось адаптировать библиотеки

- Error-handling (маппинг во внутренний формат ошибок)



Проблема 1: пришлось адаптировать библиотеки: было

```
public class ThriftExceptionHandler {  
  
    // ...  
  
    private CompositeException transform(AccessDeniedException ex) {  
        log.warn(ACCESS_DENIED_MESSAGE, ex);  
        return responseFor(INTERNAL_ERROR, i: 403, ex);  
    }  
  
    private CompositeException transform(UserNotFoundException ex) {  
        log.error(USER_NOT_FOUND_MESSAGE, ex);  
        return responseFor(INTERNAL_ERROR, i: 404, ex);  
    }  
  
    private CompositeException transform(WSTimeOutException ex) {  
        log.error(WEB_SERVICE_TIME_OUT_MESSAGE, ex);  
        return responseFor(WS_TIMEOUT, i: 500, ex);  
    }  
}
```

Проблема 1: пришлось адаптировать библиотеки: было

```
import org.springframework.aop.AfterReturningAdvice;
import org.springframework.aop.MethodBeforeAdvice;
import org.springframework.aop.ThrowsAdvice;

@Slf4j
public class TAPIControllerLoggerInterceptor implements MethodBeforeAdvice, AfterReturningAdvice, ThrowsAdvice {

    // .....

    @Override
    public void afterThrowing(Method method, Object[] args, Object target, Exception e) throws Throwable {
        exceptionHandler.handle(e);
    }
}
```


Проблема 1: пришлось адаптировать библиотеки: было

```
public static class DefaultThriftConfigurer implements ThriftConfigurer {
    @Autowired(required = false)
    private MeterRegistry meterRegistry;

    @Autowired
    private LoggingThriftMethodInterceptor loggingThriftMethodInterceptor;

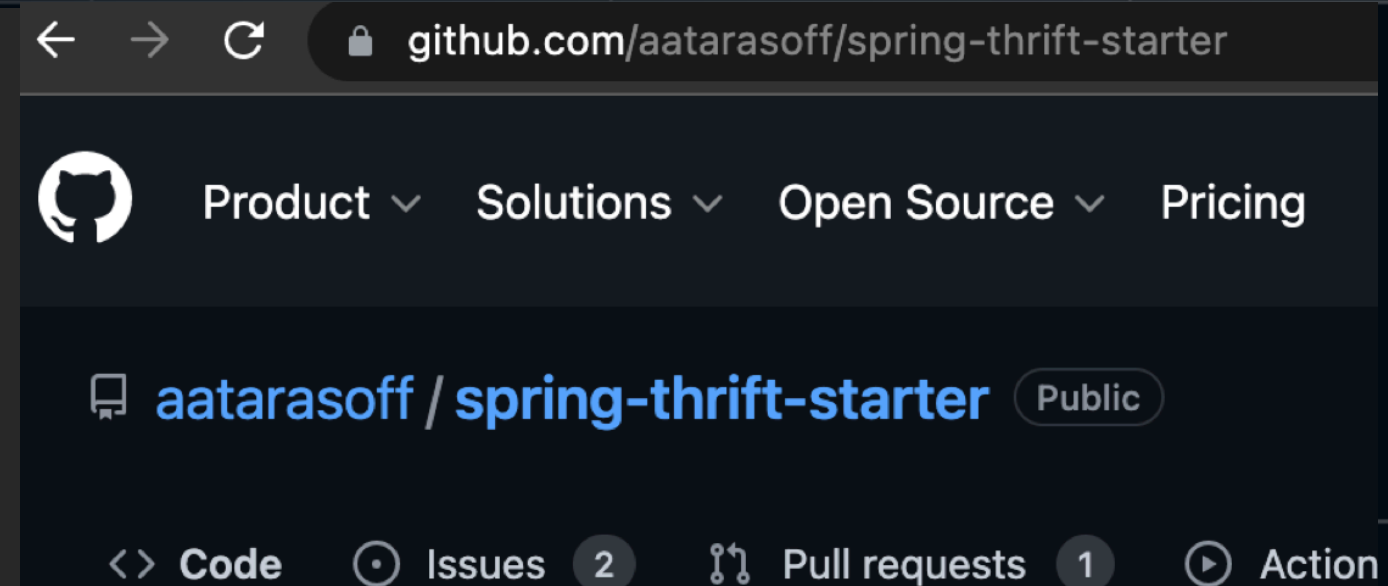
    public void configureProxyFactory(ProxyFactory proxyFactory) {
        if (meterRegistry != null) {
            proxyFactory.addAdvice(new MetricsThriftMethodInterceptor(meterRegistry));
        }
        proxyFactory.addAdvice(loggingThriftMethodInterceptor);
    }
}
```

Проблема 1: пришлось адаптировать библиотеки: было

```
public static class DefaultThriftConfigurer implements ThriftConfigurer {
    @Autowired(required = false)
    private MeterRegistry meterRegistry;

    @Autowired
    private LoggingThriftMethodInterceptor loggingThriftMethodInterceptor;

    public void configureProxyFactory(ProxyFactory proxyFactory) {
        if (meterRegistry != null) {
            proxyFactory.addAdvice(new MetricsThriftMethodInterceptor(meterRegistry));
        }
        proxyFactory.addAdvice(loggingThriftMethodInterceptor);
    }
}
```



Проблема 1: пришлось адаптировать библиотеки: стало

```
@Component
@RequiredArgsConstructor
public class ExceptionHandler implements ArmeriaExceptionHandler {

    private final ThriftExceptionHandler thriftExceptionHandler;

    @Override
    public RpcResponse apply(ServiceRequestContext ctx, Throwable cause) {
        if (cause instanceof Exception) {
            try {
                thriftExceptionHandler.handle((Exception) cause);
            } catch (CompositeException ex) {
                return RpcResponse.ofFailure(ex);
            }
        }
        return RpcResponse.ofFailure(cause);
    }
}
```

100

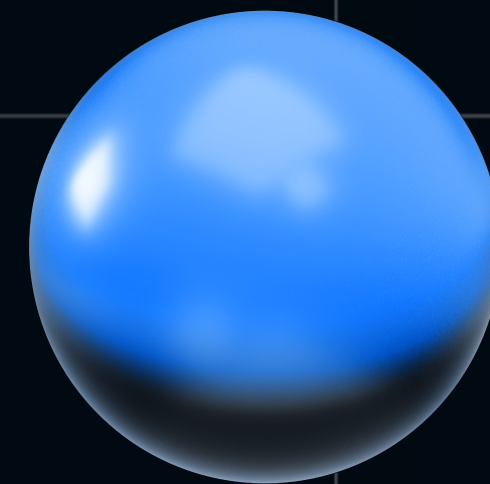
Проблема 1: пришлось адаптировать библиотеки: стало

```
@Bean
public ArmeriaServerConfigurator armeriaServerConfigurator(
    ApplicationContext context,
    List<Function<HttpService, HttpService>> httpDecorators,
    List<DecoratingRpcServerFunction> rpcDecorators,
    ArmeriaExceptionHandler exceptionHandler,
    MeterRegistry registry
) {
```


Проблема 1: пришлось адаптировать библиотеки: стало

```
THttpClientBuilder tHttpClientBuilder = THttpClient.builder()  
    .addService(bean)  
    .defaultSerializationFormat(ThriftSerializationFormats.BINARY)  
    .exceptionHandler(exceptionHandler);
```

Проблема 2: error-handling



103

Проблема 2: error-handling

```
@ThriftController(path = "/api")
public class AccountsController implements AccountsApiService.AsyncIface {

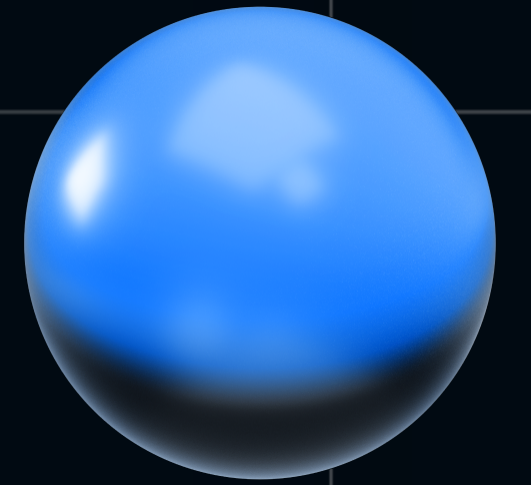
    @Override
    public void getAccountsInfo(UserData userData,
                               String organizationId,
                               AsyncMethodCallback<List<AccountInfo>> resultHandler) {

        try {
            Assert.hasLength(organizationId, ERROR_MESSAGE);
            var future = permissionsService.checkPermission(userData, organizationId)
                .thenCompose(ifSuccess -> accountListService.getAccountInfoList(userData, organizationId));
            subscribeToResult(resultHandler, future);
        } catch (Exception e) {
            logAndReturnError(resultHandler, e);
        }
    }

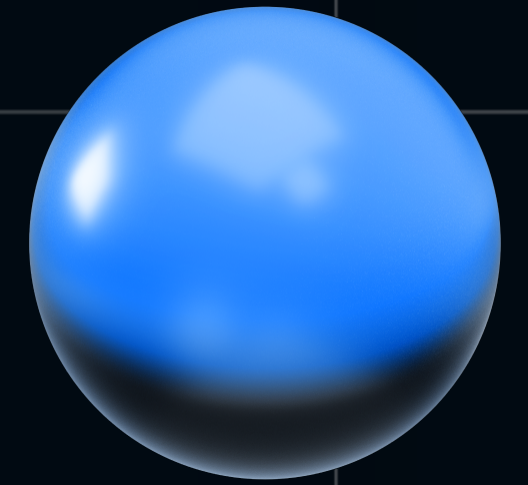
    private <T> void subscribeToResult(AsyncMethodCallback<T> resultHandler, CompletableFuture<T> future) {
        future.thenAccept(resultHandler::onComplete)
            .exceptionally(t -> logAndReturnError(resultHandler, new Exception(t)));
    }

    private void logAndReturnError(AsyncMethodCallback<?> resultHandler, Exception e) {
        log.error("Catch exception outside of future", e);
        resultHandler.onError(e);
    }
}
```

Проблема 3: асинхронный logback

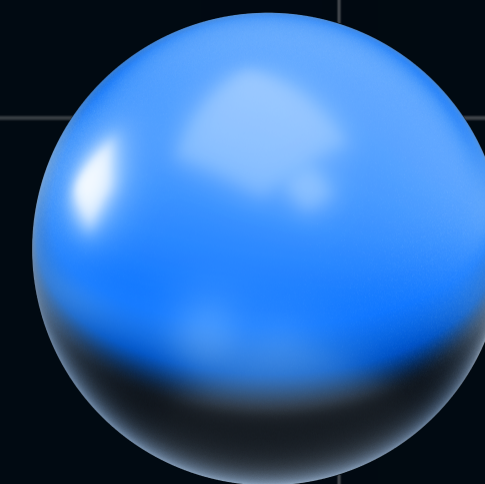


Проблема 3: асинхронный logback



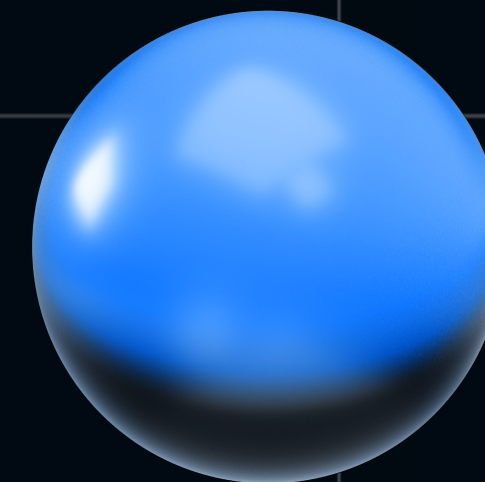
```
<appender name="COMPOSITE_LOGSTASH_ASYNC"  
  class="net.logstash.logback.appender.LoggingEventAsyncDisruptorAppender">  
  <includeCallerData>${includeCallerData}</includeCallerData>  
  <ringBufferSize>${ringBufferSize}</ringBufferSize>  
  <producerType>${producerType}</producerType>  
  <waitStrategyType>${waitStrategyType}</waitStrategyType>  
  <daemon>${daemon}</daemon>  
  <droppedWarnFrequency>${droppedWarnFrequency}</droppedWarnFrequency>  
  
  <appender-ref ref="CONSOLE"/>  
  <appender-ref ref="LOGSTASH"/>  
</appender>
```

Armeria: за и против



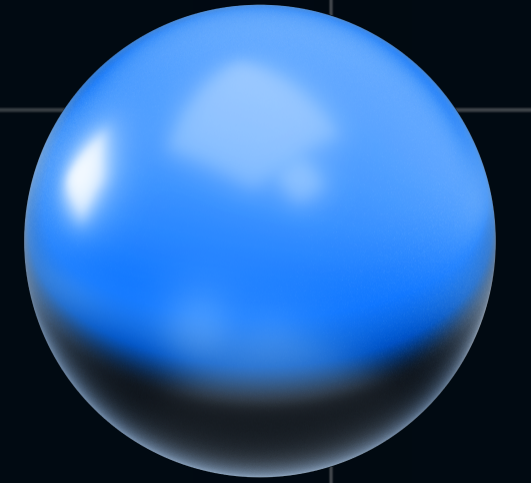
Armeria: за и против

+ работает с thrift, gRPC



108

Armeria: за и против

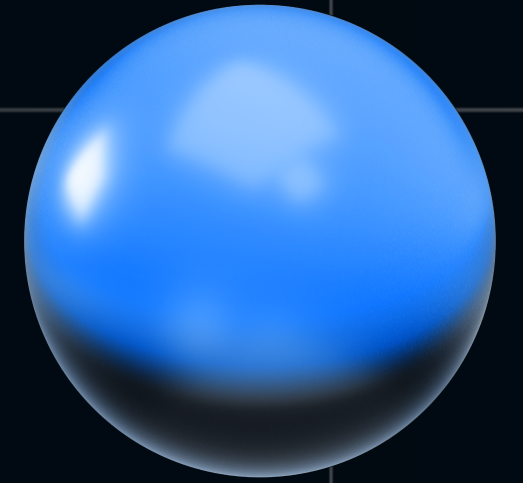


+ работает с thrift, gRPC

- если у вас только рест - берите webflux; к нему больше библиотек и больше информации



Armeria: за и против



+ работает с thrift, gRPC

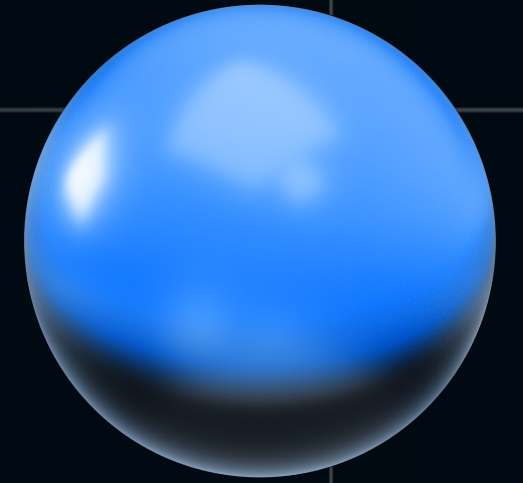
+ http/2

- если у вас только рест - берите webflux; к нему больше библиотек и больше информации



110

Armeria: за и против



+ работает с thrift, gRPC

+ http/2

+ микросервисные «обязки»: circuit

breaker, client-side балансировка,

service-discovery

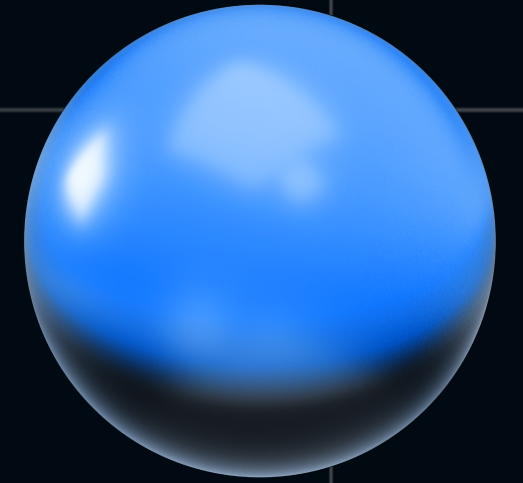
- если у вас только рест - берите

webflux; к нему больше библиотек и

больше информации



Armeria: за и против



+ работает с thrift, gRPC

+ http/2

+ микросервисные «обязки»: circuit

breaker, client-side балансировка,

service-discovery

- если у вас только рест - берите

webflux; к нему больше библиотек и

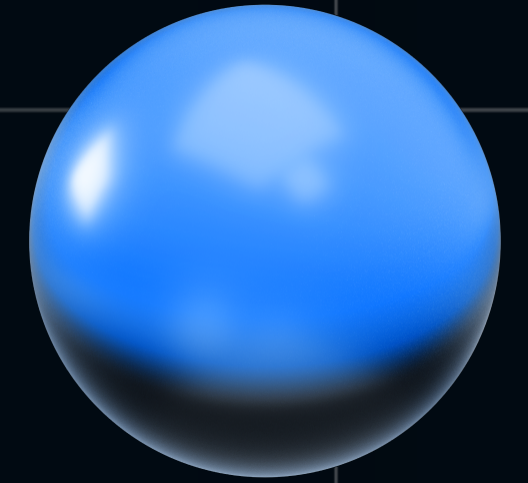
больше информации

- но к webflux этих библиотек все равно

больше



Armeria: за и против



+ работает с thrift, gRPC

+ http/2

+ микросервисные «обязки»: circuit

breaker, client-side балансировка,

service-discovery

+ Хорошо расширяем за счет

декораторов

- если у вас только рест - берите

webflux; к нему больше библиотек и

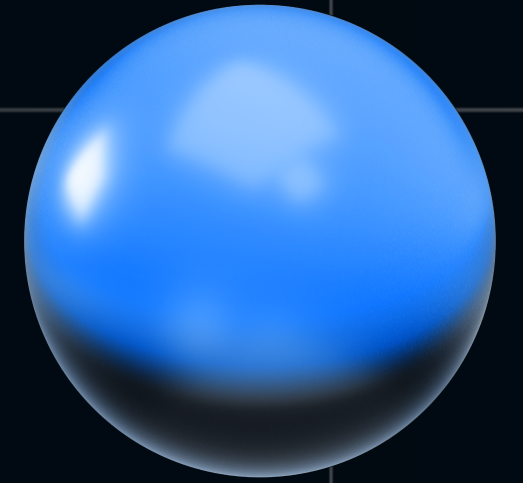
больше информации

- но к webflux этих библиотек все равно

больше



Armeria: за и против

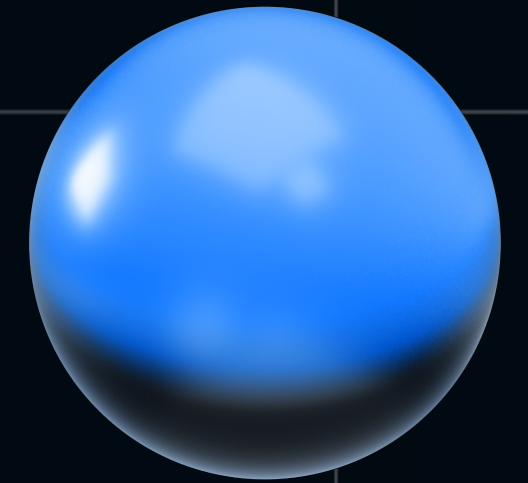


- + работает с thrift, gRPC
- + http/2
- + микросервисные «обвязки»: circuit breaker, client-side балансировка, service-discovery
- + Хорошо расширяем за счет декораторов
- + Хорошо интегрируется со спрингом

- если у вас только рест - берите webflux; к нему больше библиотек и больше информации
- но к webflux этих библиотек все равно больше



Armeria: за и против



stackoverflow.com/questions/tagged/spring-webflux

overflow About Products For Teams [spring-webflux]

Questions tagged [spring-webflux]

Ask Question

Spring Framework 5 includes a new spring-webflux module. The module contains support for reactive HTTP and WebSocket clients as well as for reactive server web applications including REST, HTML browser, and WebSocket style interactions. WebFlux can run on Servlet containers with support for the Servlet 3.1 non-blocking I/O API as well as on other async runtimes such as Netty and Undertow.

[Learn more...](#) [Top users](#) [Synonyms \(3\)](#)

5,752 questions

Newest Active Bountied 2 Unanswered More Filter

stackoverflow.com/questions/tagged/armeria

flow About Products For Teams [armeria]

Questions tagged [armeria]

Ask Question

Armeria is an open-source asynchronous HTTP/2 RPC/REST client/server library built on top of Java 8, Netty, Thrift and gRPC. Its primary goal is to help engineers build high-performance asynchronous microservices that use HTTP/2 as a session layer protocol.

[Learn more...](#) [Top users](#) [Synonyms](#)

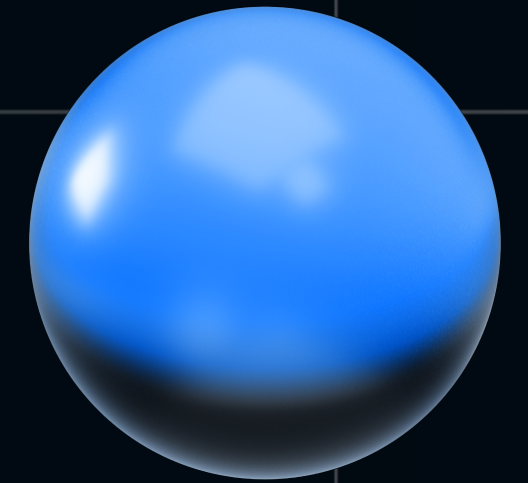
31 questions

Newest Active Bountied Unanswered More Filter



115

Armeria: за и против



stackoverflow.com/questions/tagged/spring-webflux

overflow About Products For Teams [spring-webflux]

Questions tagged [spring-webflux]

Ask Question

Spring Framework 5 includes a new spring-webflux module. The module contains support for reactive HTTP and WebSocket clients as well as for reactive server web applications including REST, HTML browser, and WebSocket style interactions. WebFlux can run on Servlet containers with support for the Servlet 3.1 non-blocking I/O API as well as on other async runtimes such as Netty and Undertow.

[Learn more...](#) [Top users](#) [Synonyms \(3\)](#)

5,752 questions Newest Active Bountied 2 Unanswered More Filter

stackoverflow.com/questions/tagged/armeria

flow About Products For Teams [armeria]

Questions tagged [armeria]

Ask Question

Armeria is an open-source asynchronous HTTP/2 RPC/REST client/server library built on top of Java 8, Netty, Thrift and gRPC. Its primary goal is to help engineers build high-performance asynchronous microservices that use HTTP/2 as a session layer protocol.

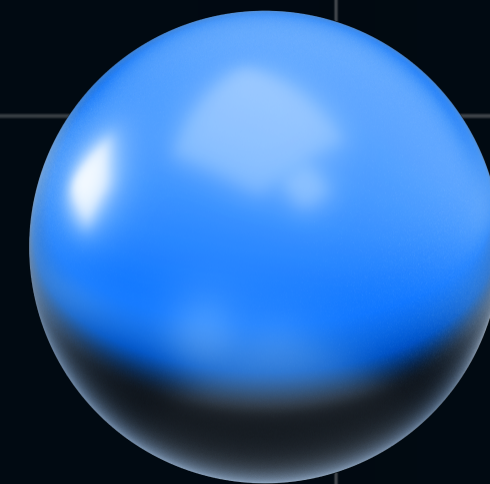
[Learn more...](#) [Top users](#) [Synonyms](#)

31 questions Newest Active Bountied Unanswered More Filter



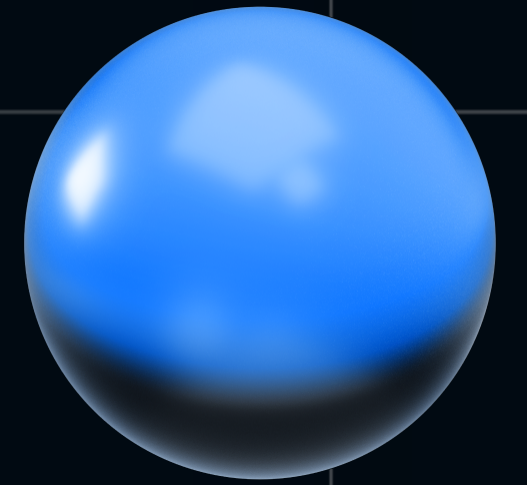
116

Результат



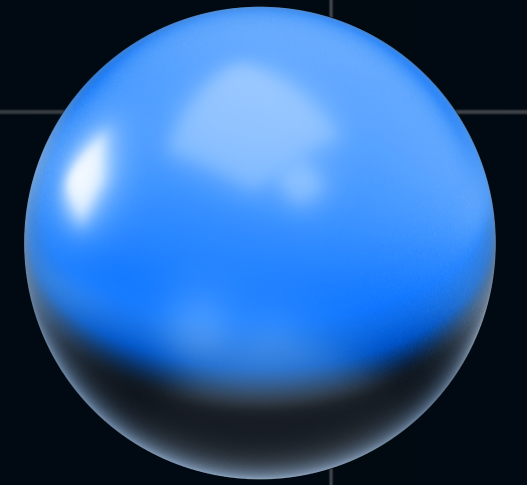
Результат

- Проблема решена



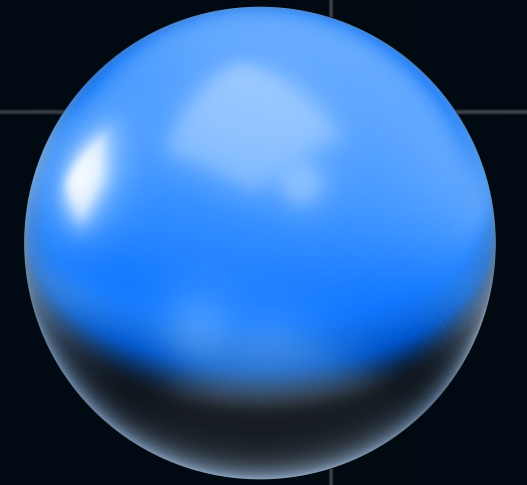
Результат

- Проблема решена
- Аналогичные проблемы решаются по мере поступления



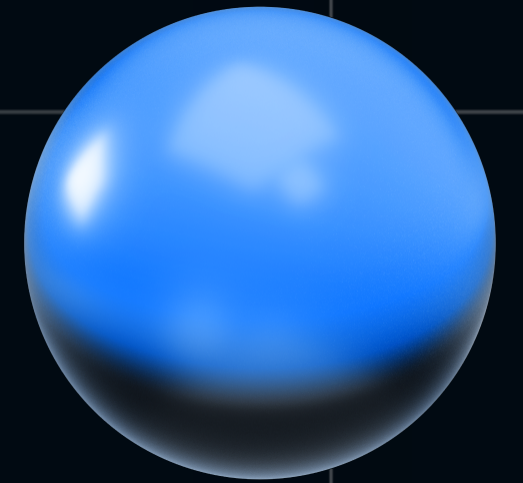
Результат

- Проблема решена
- Аналогичные проблемы решаются по мере поступления
- 10 / 200 микросервисов реактивные



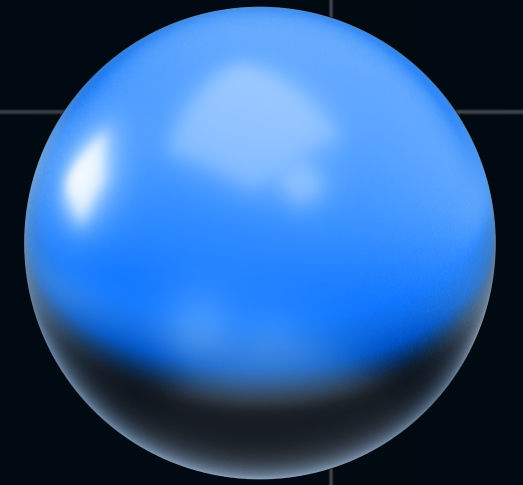
Результат

- Проблема решена
- Аналогичные проблемы решаются по мере поступления
- 10 / 200 микросервисов реактивные
- Реактивные микросервисы не пишем



Результат

- Проблема решена
- Аналогичные проблемы решаются по мере поступления
- 10 / 200 микросервисов реактивные
- Реактивные микросервисы не пишем
- Переписываем с синхронных на реактивные по мере необходимости

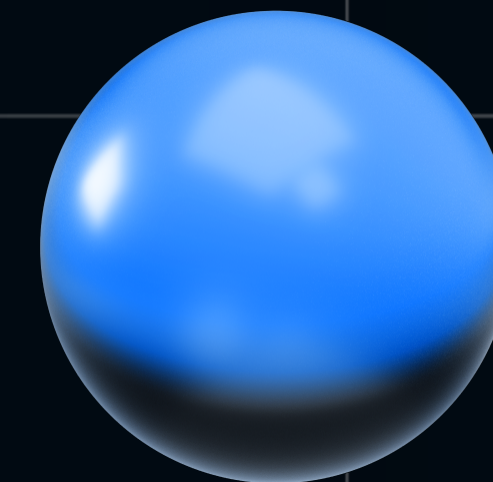


122

Спасибо!



@KTO_VIKTOR



123