System design: simple but common mistakes

Kirill Parasotchenko

Issue 1. Idempotent id

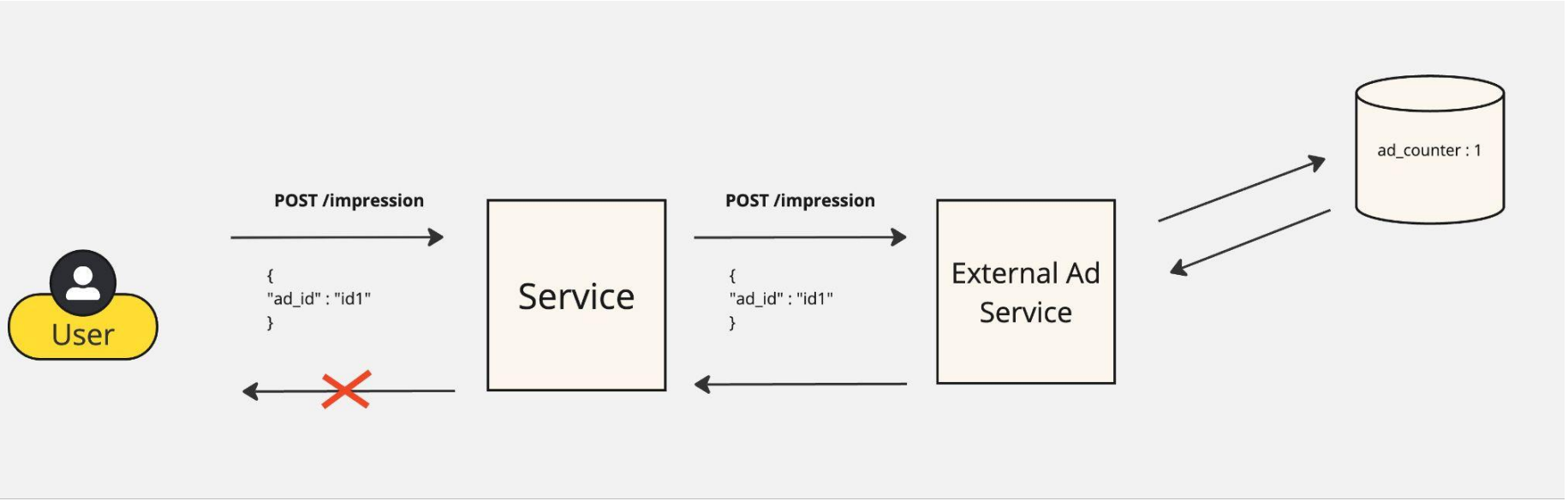
Case:

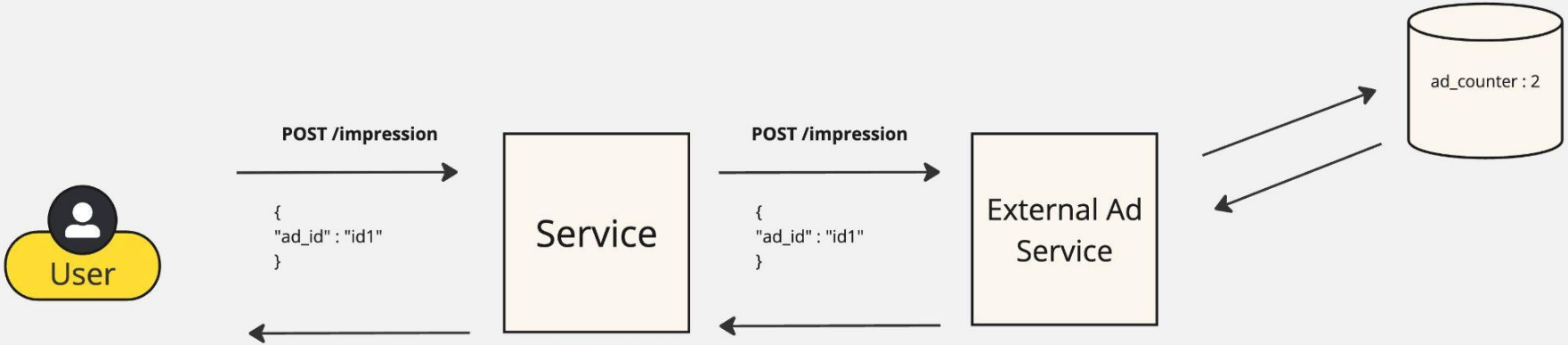
Communication between two services with data creation in the second one

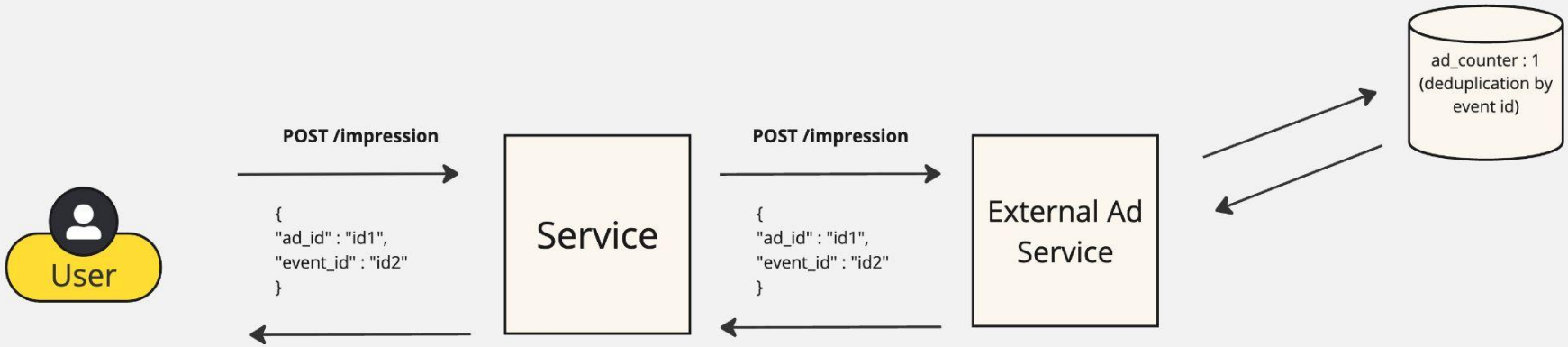
Potential issues:

Duplicated data

Example: Advertisement service. User sending an ad event




User
POST /impression
{
"ad_id" : "id1"
}
Service
POST /impression
{
"ad_id" : "id1"
}
External Ad Service
ad_counter : 2


User
POST /impression
{
"ad_id" : "id1",
"event_id" : "id2"
}
Service
POST /impression
{
"ad_id" : "id1",
"event_id" : "id2"
}
External Ad Service
ad_counter : 1
(deduplication by event id)

Issue 2. **External request inside transaction**

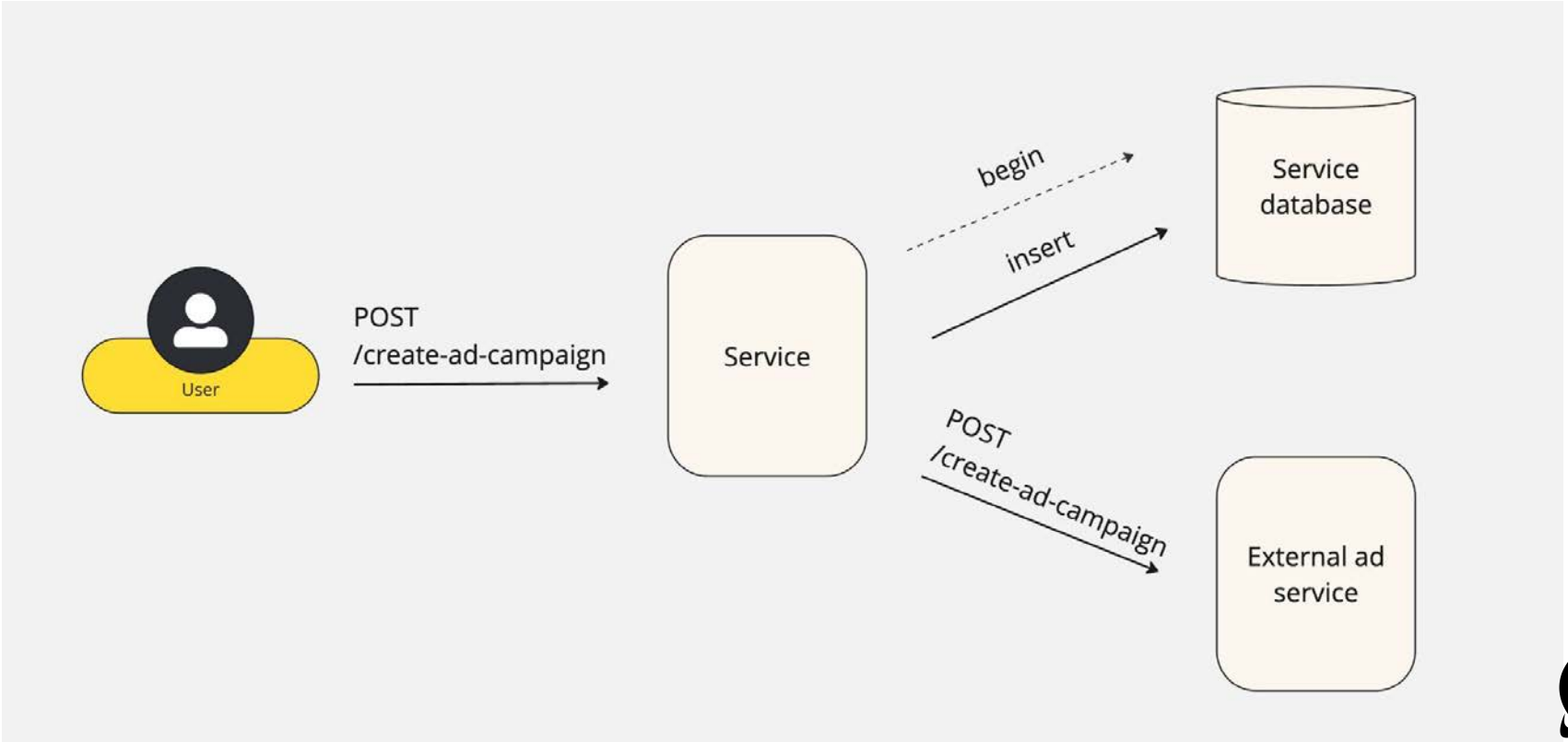
Case:

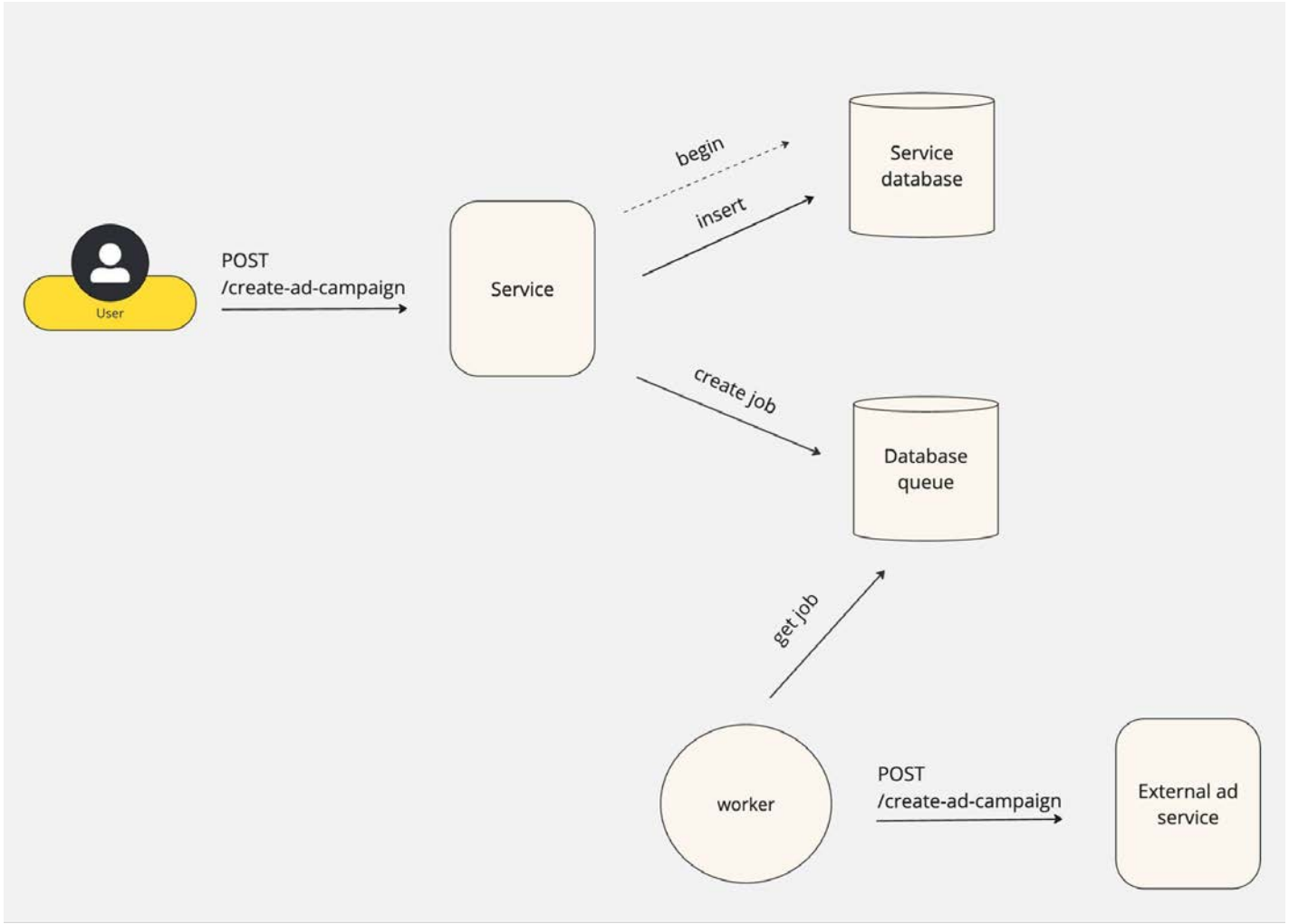
Creation/updating service db and data of an external service inside

Potential issues:

Exhausted db connection pool

Example. Advertisement service. Ad campaign creation




User
POST
/create-ad-campaign
Service
begin
insert
Service database
create job
Database queue
get job
worker
POST
/create-ad-campaign
External ad service

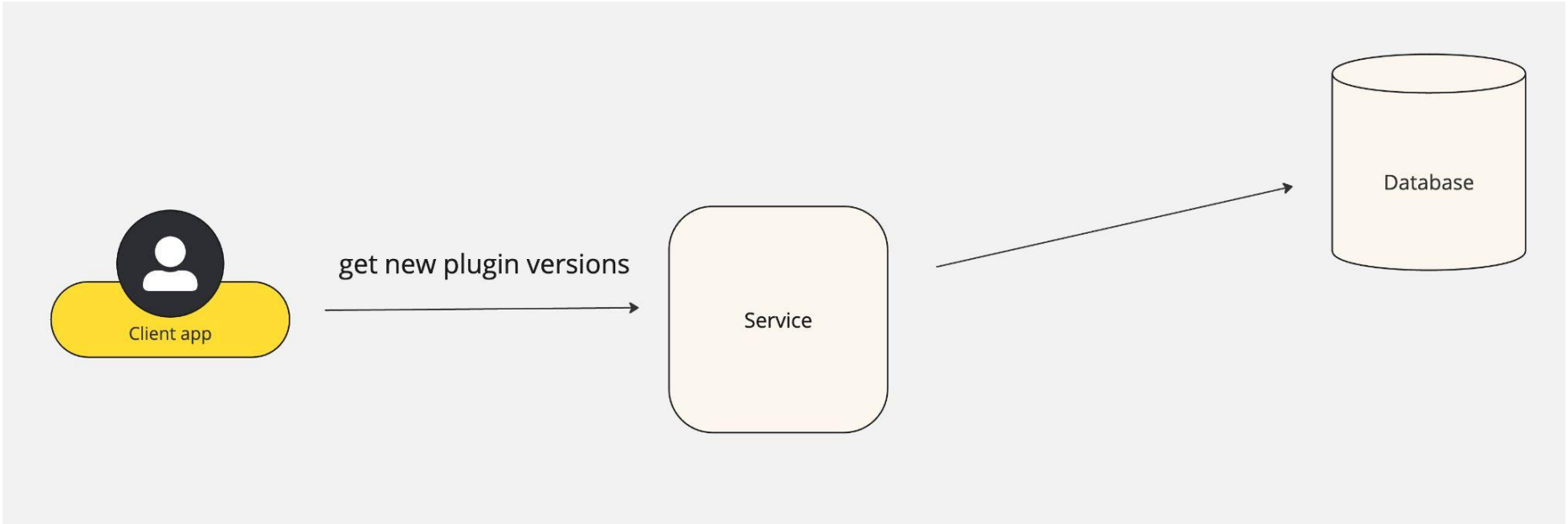
Issue 3. Requesting service at the same time

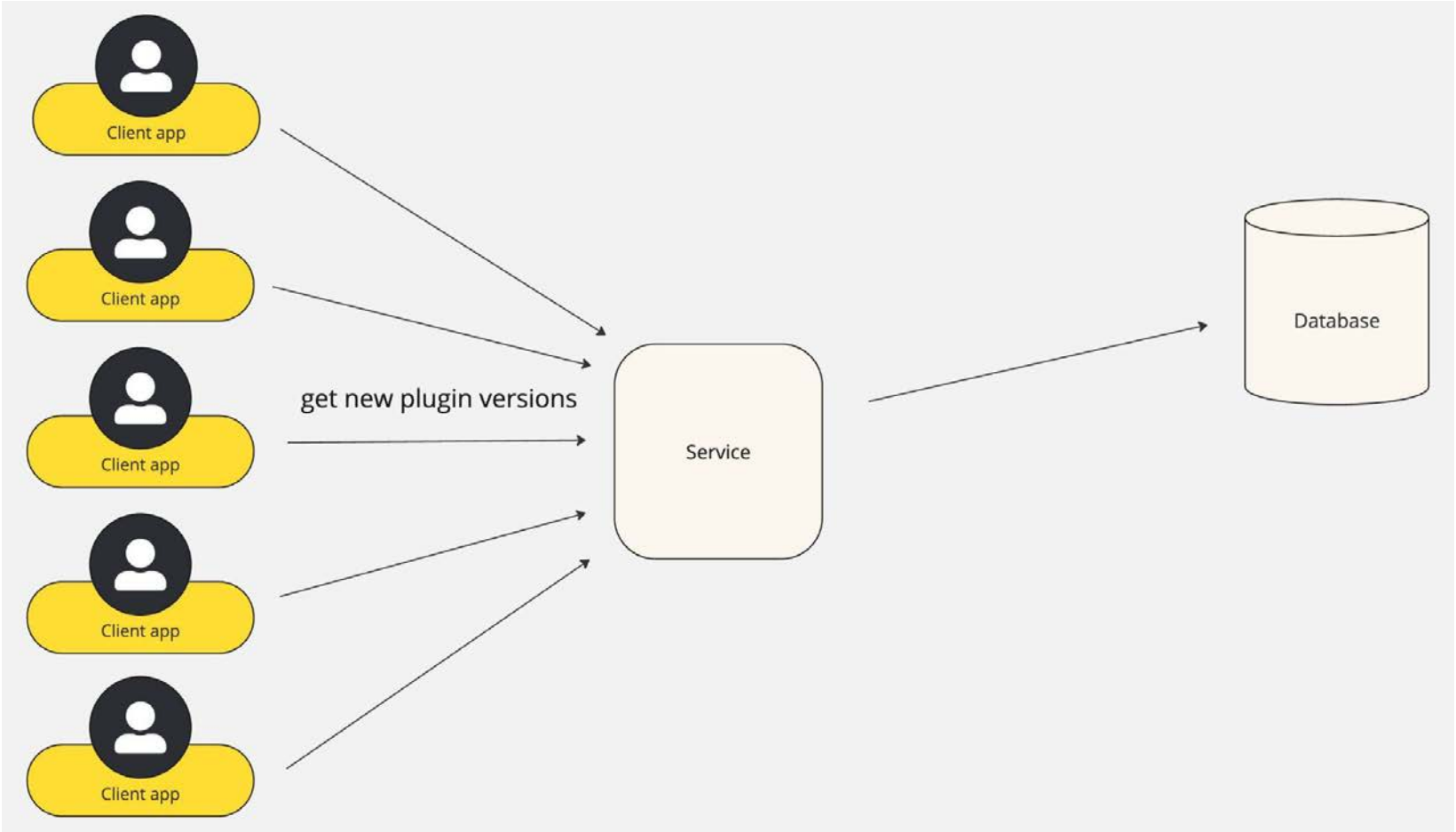
Case:

Multiple clients request a service at the same time

Potential issues:

Overloaded service leading to a temporary outage


Client app
get new plugin versions
Service
Database


Client app
Client app
Client app
Client app
Client app
get new plugin versions
Service
Database

Issue 4. Lack of rate limiter

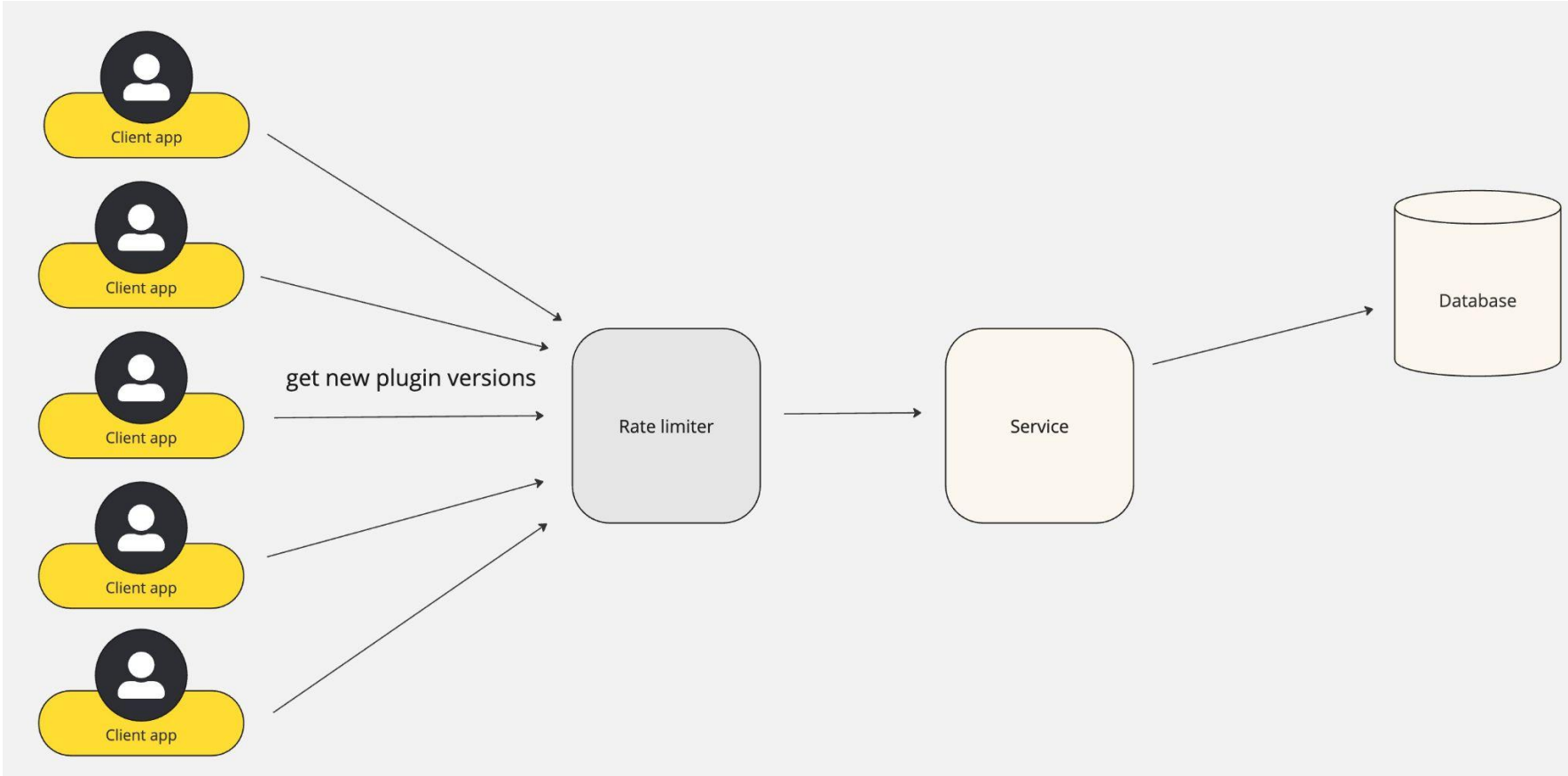
Case:

One or several clients use all the service resources. Meanwhile other clients can’t use the service. It happens due to DDoS attack or bad design.

Potential issues:

Service can’t handle all the clients

Overloaded service leading to a temporary outage


Client app
Client app
Client app
Client app
Client app
get new plugin versions
Rate limiter
Service
Database

Issue 5. Lack of memory limiter

Case:

Client sends a big request

Potential issues:

Temporary outage (OOM)

```go
func (h *FooHandler) ServeHTTP(writer http.ResponseWriter, request *http.Request) {
    var (
        msgPrefix = "FooHandler.ServeHTTP"
        user      User
    )

    log.Printf( format: "%s: request received\n", msgPrefix)

    if request.Method != http.MethodPost {
        writer.WriteHeader(http.StatusMethodNotAllowed)
        return
    }

    b, err := io.ReadAll(request.Body)
```

```go
func (h *FooHandler) ServeHTTP(writer http.ResponseWriter, request *http.Request) {
    var (
        msgPrefix = "FooHandler.ServeHTTP"
        user      User
    )

    log.Printf( format: "%s: request received\n", msgPrefix)

    if request.Method != http.MethodPost {
        writer.WriteHeader(http.StatusMethodNotAllowed)
        return
    }

    // 500KB
    request.Body = http.MaxBytesReader(writer, request.Body,  n: 512000)

    b, err := io.ReadAll(request.Body)
```

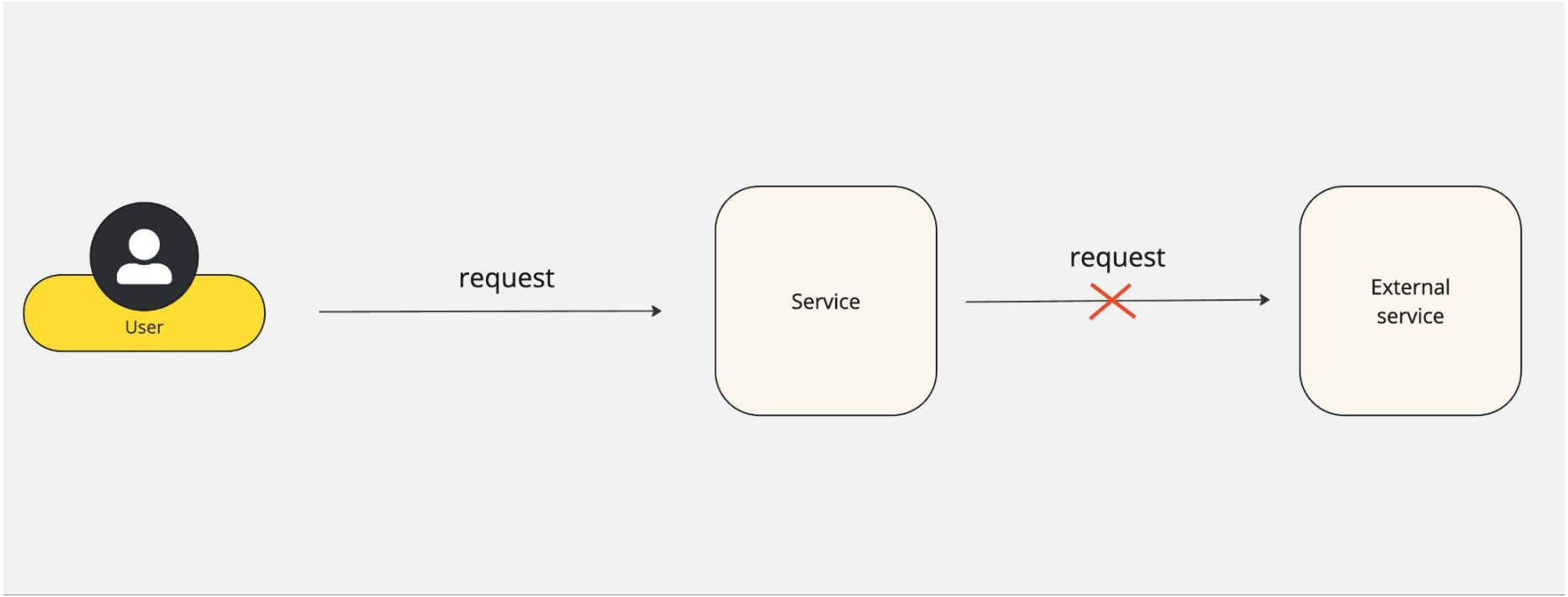
Issue 6. **No retries**

Case:

Request to an external service without retry in case of failure.

Potential issues:

High error rate of the service


User
request
Service
request
External service

Issue 7. There are retries but no backoff

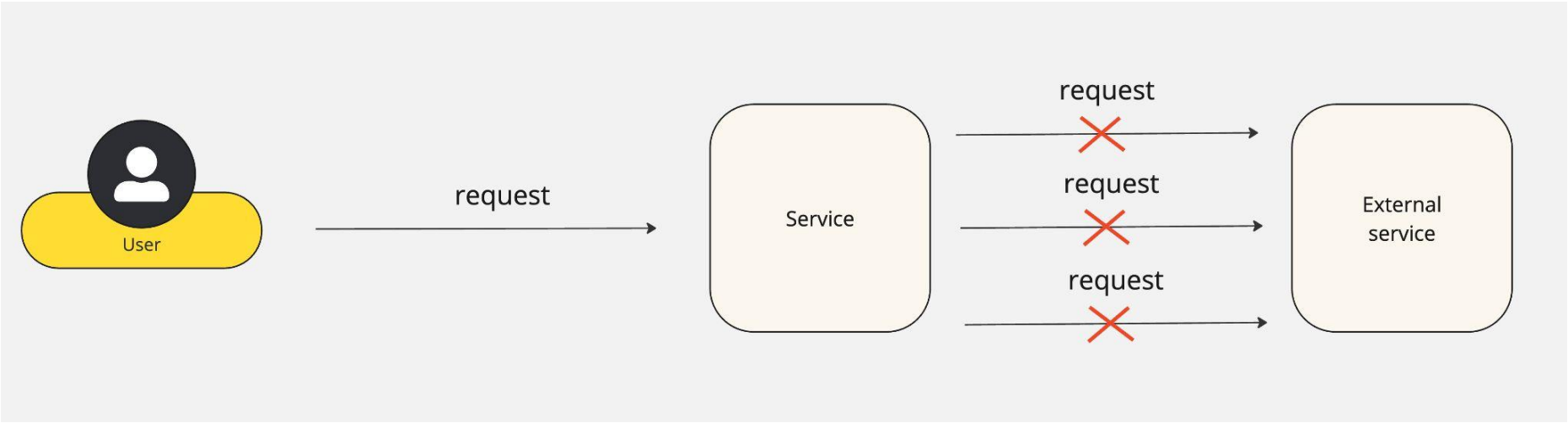
Case:

Request to an external service with retries but without backoff

Potential issues:

Overloaded external service

High error rate of the service


User
request
Service
request
request
request
External service

Backoff strategies

- Linear
- Linear with jitter
- Exponential
- Exponential with jitter

Thank you