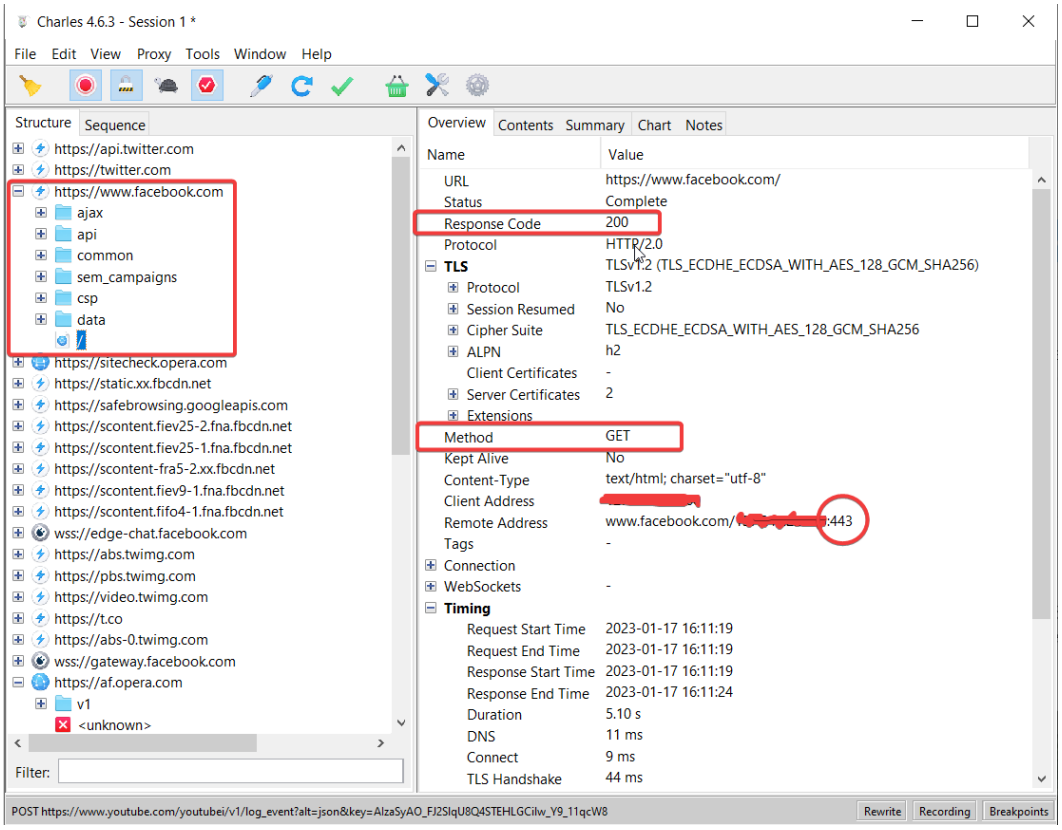
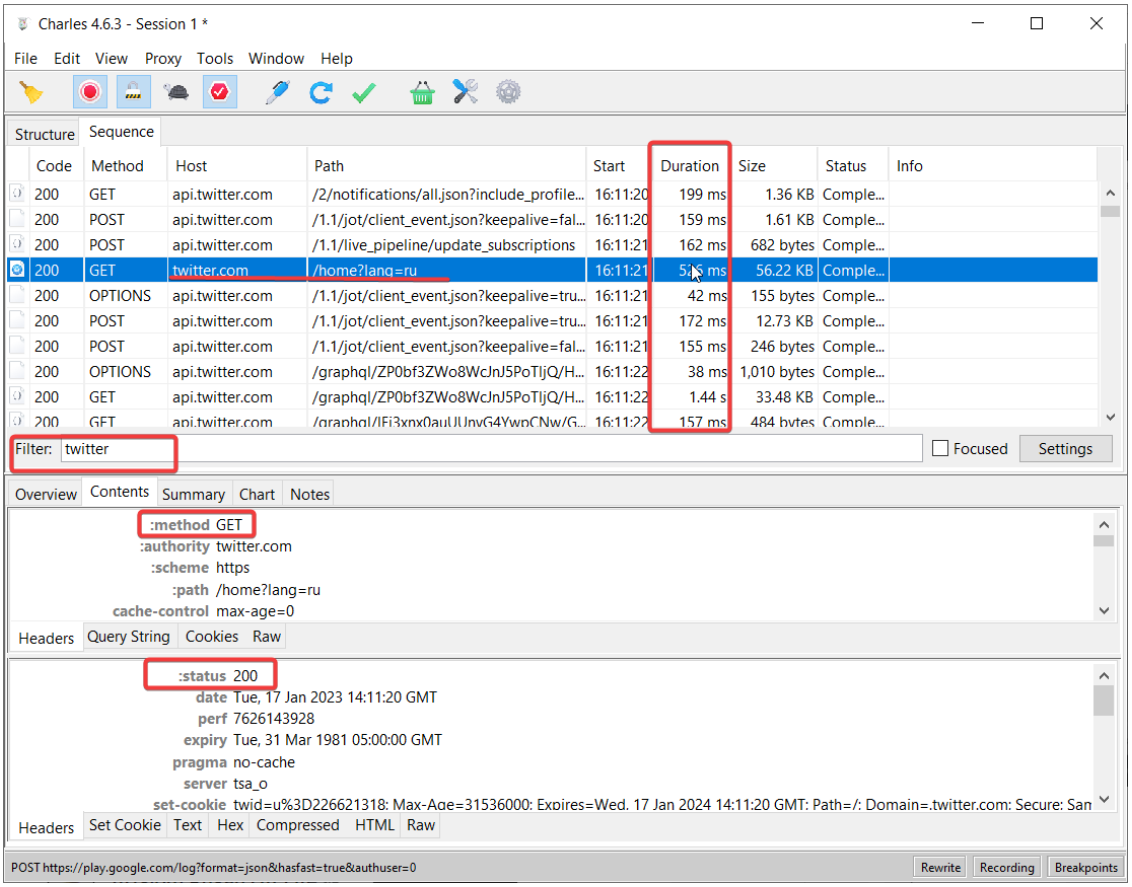
Charles Proxy is a proxy-server, “sniffer”, the traffic analyzer, is used to intercept requests not only from the browser (client), but also from desktop applications

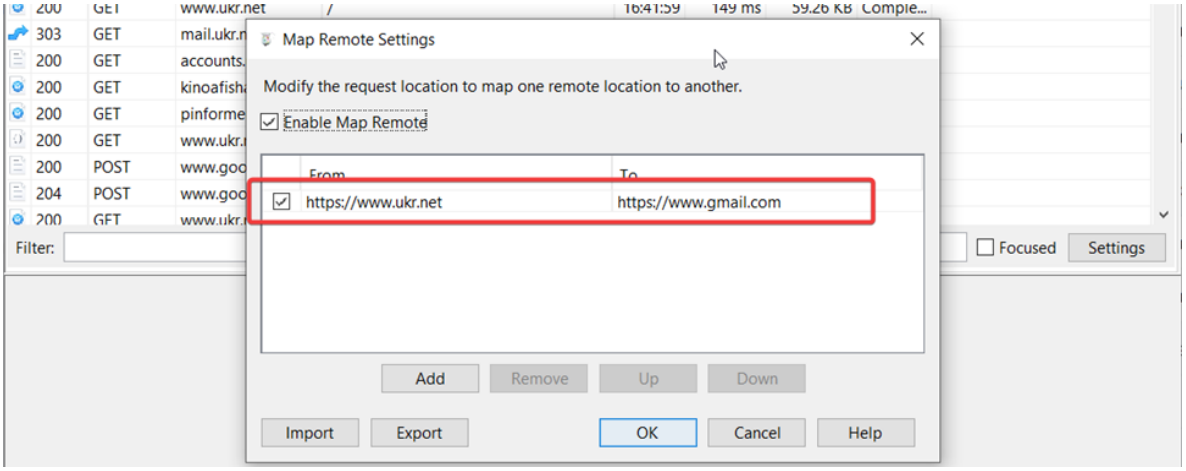


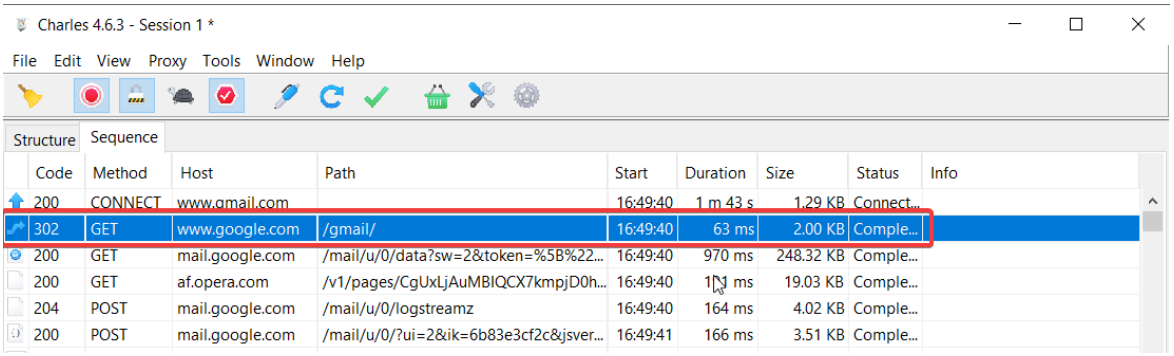
In intercepted requests, you can view the link, status, response code, request method, client address, and remote address with port. Herein, Charles Proxy is configured to intercept HTTPS requests as well.



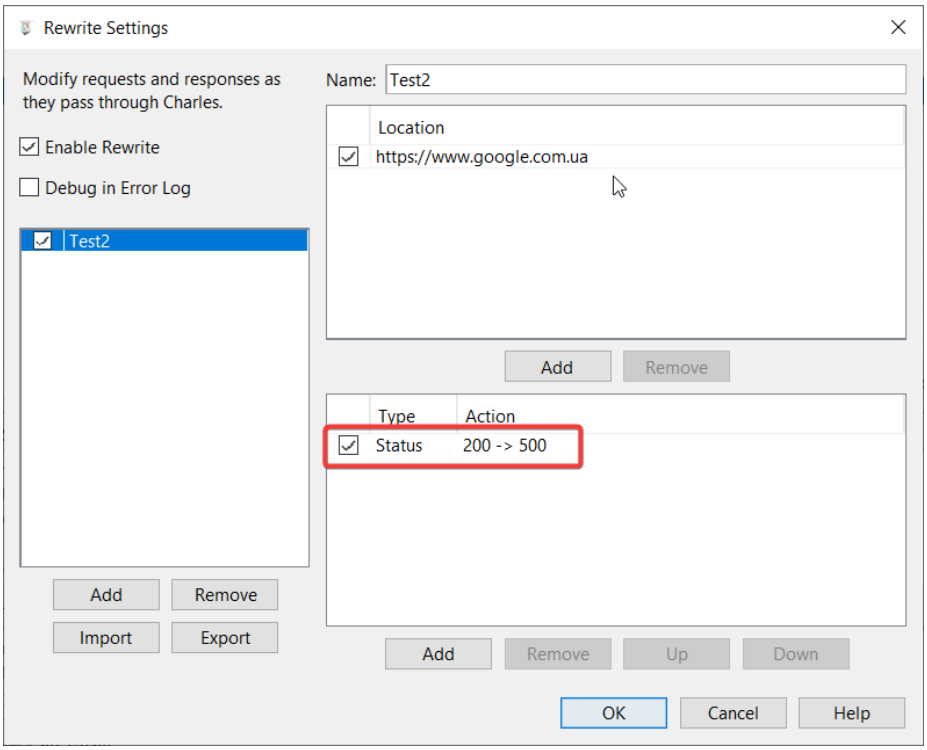
In the Sequence section, everything is very similar to Chrome DevTools. You can filter by the required URL, check the path by which the request is sent, its duration, and also view the content of the request and responses, check whether there are no duplicate requests.

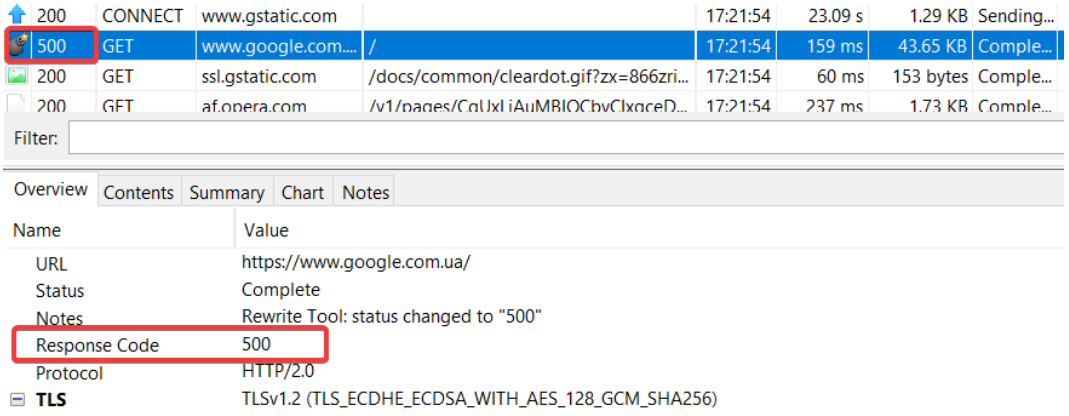
Map Remote tool changes the request location, and the response is transparently served from the new location as if that was the original request.



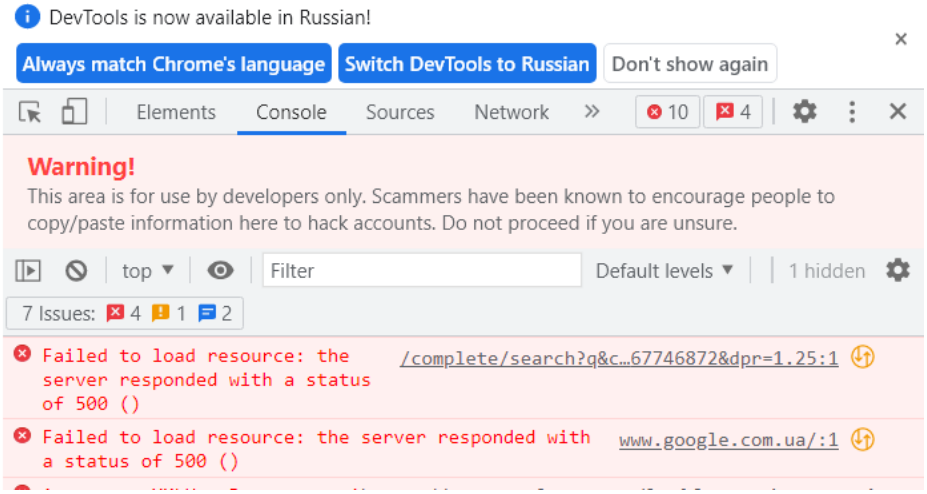


Rewrite tool helps to create rules that modify requests and responses as they pass through Charles, such as adding, removing or changing the header, response code, change of URL.

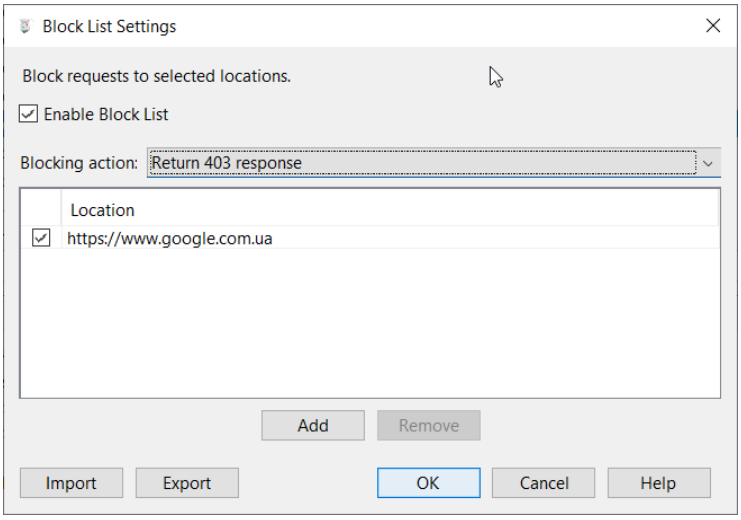


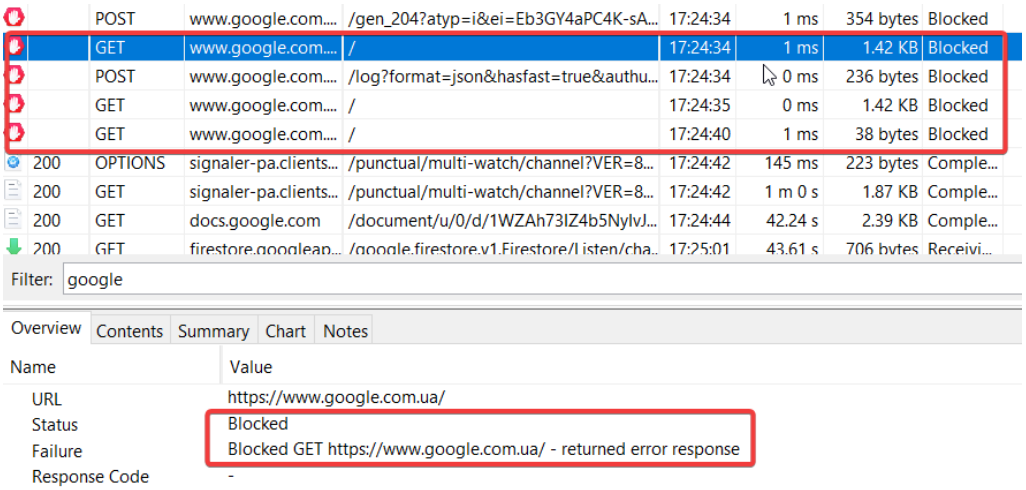


Though it did not influence the representation of https://www.google.com.ua, Charles shows that the server returned a 500 code and the Console tab in the browser shows that the server responded with a 500 error. It is not clear whether this is a bug or a feature of the system.



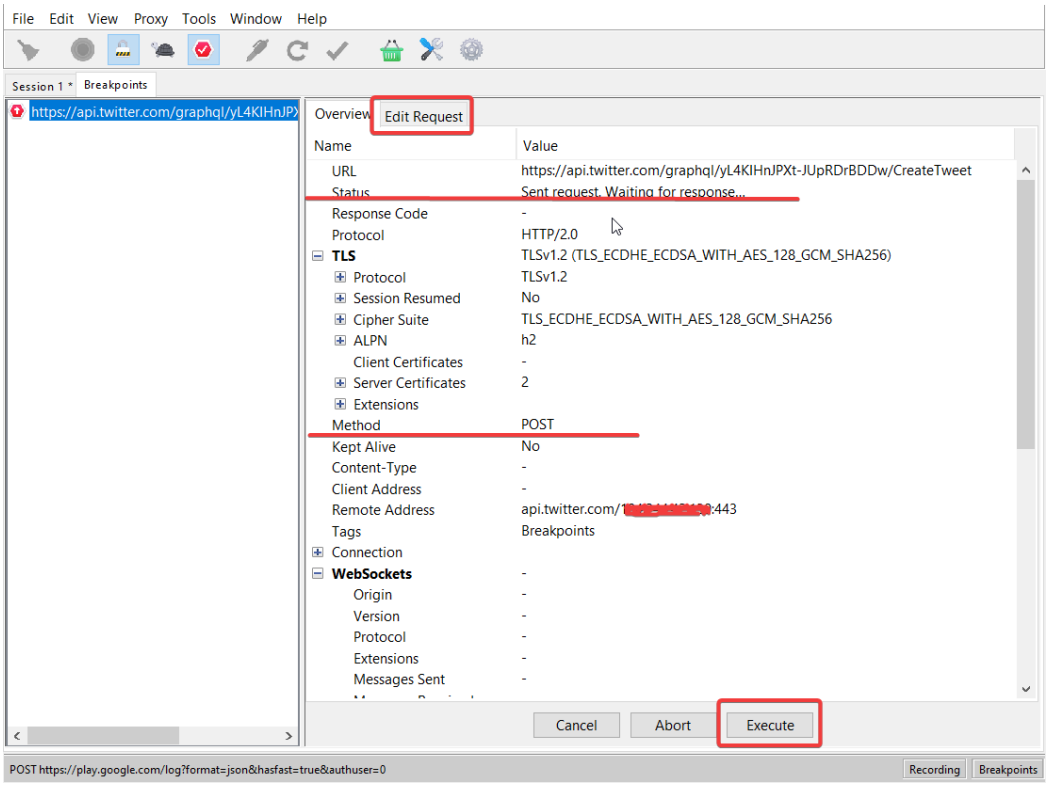
Block List tool makes it possible to enter domain names which should be blocked. Whenever the browser tries to reach out to the resource, it gets an error as a response. *Allow List* tool works opposite: every resource except for the domains listed will be blocked.



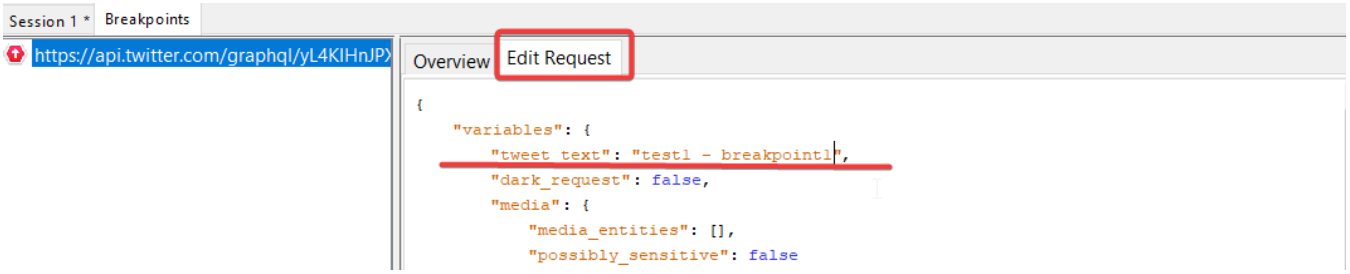


Charles and browser show the reaction to *Block action* correctly.
(Screenshot below says: *"Web page probably is temporarily unavailable or permanently moved to the new address"*)

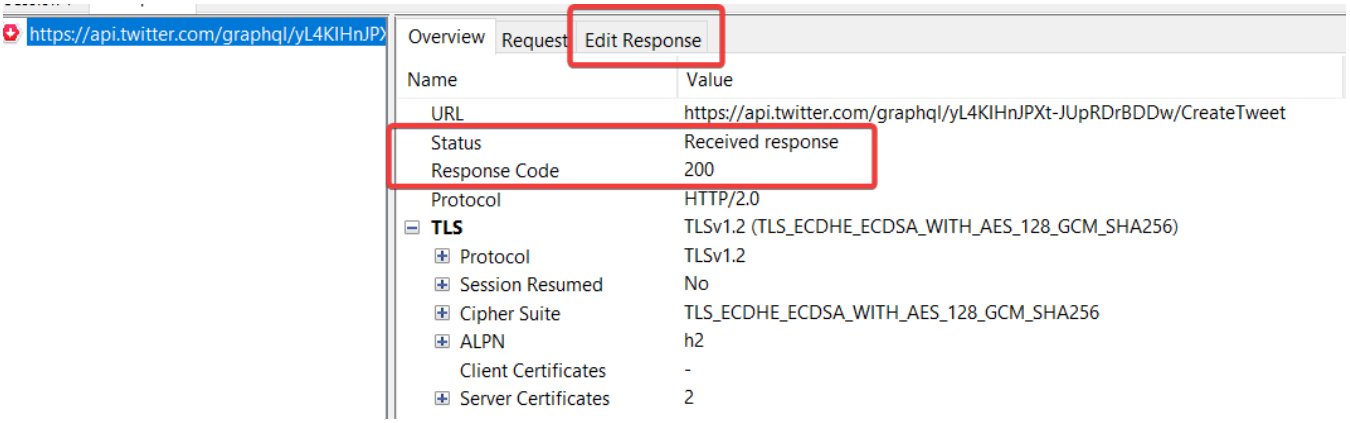
Breakpoints tool lets you intercept requests and responses before they are passed through Charles, and subsequently modify and edit them, to execute them or to block.
The first request is made, it is intercepted and breakpoints are set with the right mouse button. Repeat the request, and the request is intercepted on the way to the server.



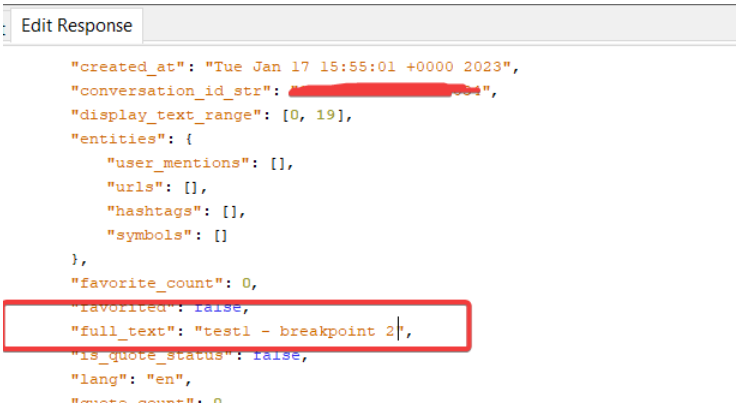
Change the text in the "Edit" tab. What we had when sending a request from the GUI: *test1*, we add *breakpoint1* to Charles. Click "Execute", and the request is sent further to the server.



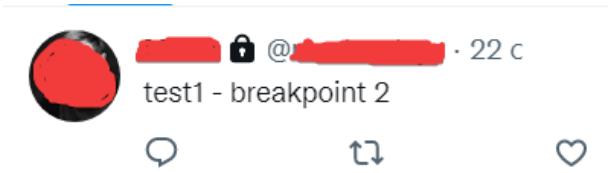
Then, Charles intercepts the server's response.



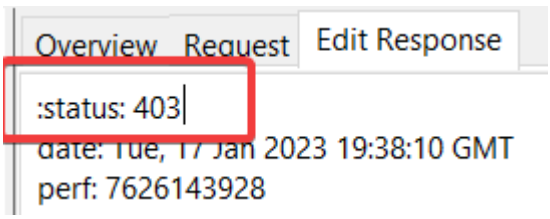
Again, we change the data in the intercepted response. Click "Execute" and the client should receive this data for display.



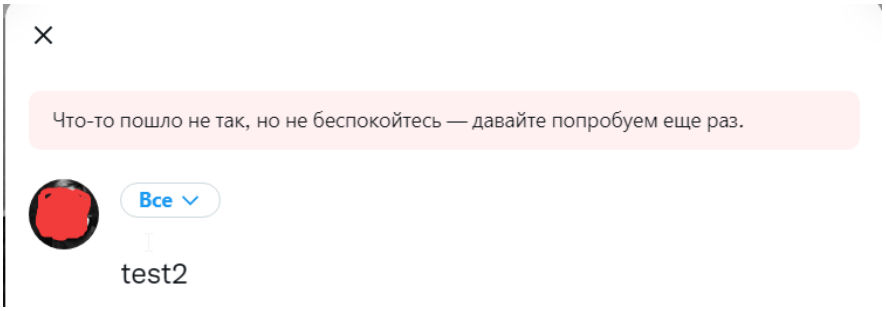
Результат:



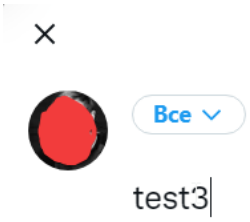
When intercepting and modifying requests at breakpoints, you can edit anything. For example, we will launch a request to publish a tweet again, leaving the request unchanged, but in the intercepted response from the server we will change the status code from 200 to 403.



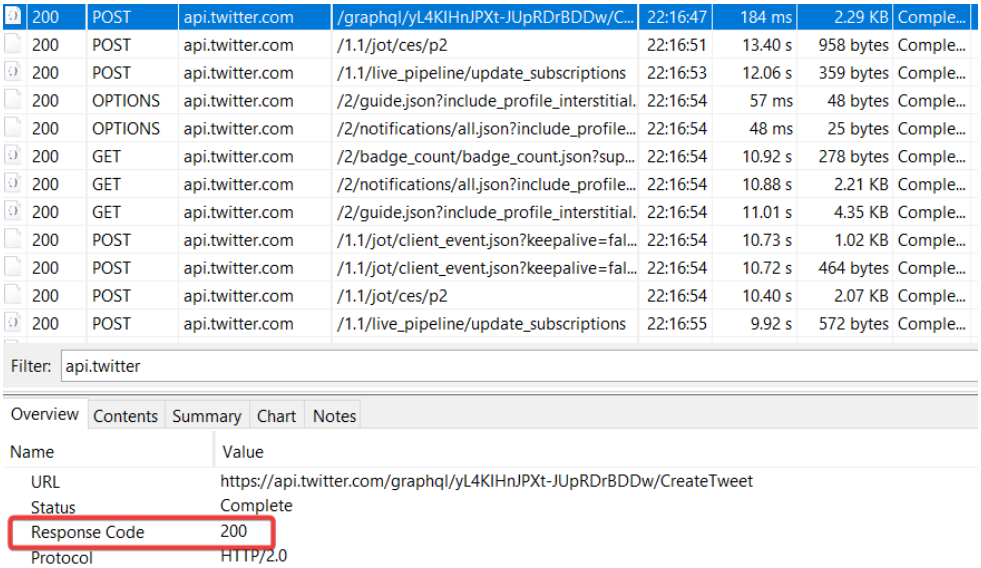
Outcome in Twitter is the following:
(Screenshot below says: *"Something went wrong, but don't worry, let's try again"*)



It is appropriate to carry out another test. It is known that the length of a tweet under the new rules is 280 characters. If we write more than 280 characters on the GUI at once, validation will work. But it is necessary to check it without the participation of the GUI, but with a request immediately to the server.
We create a tweet that looks normal at the beginning and complies with the rules.



When the request is intercepted, we change the content of the tweet to a text that will exceed the limit. I chose a paragraph of text that contained 292 characters. We receive a response and see that the server returned a status code 200.



200	POST	api.twitter.com	/graphql/yL4KIHnJPXt-JUpRDrBDDw/C...	22:16:47	184 ms	2.29 KB	Comple...
200	POST	api.twitter.com	/1.1/jot/ces/p2	22:16:51	13.40 s	958 bytes	Comple...
200	POST	api.twitter.com	/1.1/live_pipeline/update_subscriptions	22:16:53	12.06 s	359 bytes	Comple...
200	OPTIONS	api.twitter.com	/2/guide.json?include_profile_interstitial.	22:16:54	57 ms	48 bytes	Comple...
200	OPTIONS	api.twitter.com	/2/notifications/all.json?include_profile...	22:16:54	48 ms	25 bytes	Comple...
200	GET	api.twitter.com	/2/badge_count/badge_count.json?sup...	22:16:54	10.92 s	278 bytes	Comple...
200	GET	api.twitter.com	/2/notifications/all.json?include_profile...	22:16:54	10.88 s	2.21 KB	Comple...
200	GET	api.twitter.com	/2/guide.json?include_profile_interstitial.	22:16:54	11.01 s	4.35 KB	Comple...
200	POST	api.twitter.com	/1.1/jot/client_event.json?keepalive=fal...	22:16:54	10.73 s	1.02 KB	Comple...
200	POST	api.twitter.com	/1.1/jot/client_event.json?keepalive=fal...	22:16:54	10.72 s	464 bytes	Comple...
200	POST	api.twitter.com	/1.1/jot/ces/p2	22:16:54	10.40 s	2.07 KB	Comple...
200	POST	api.twitter.com	/1.1/live_pipeline/update_subscriptions	22:16:55	9.92 s	572 bytes	Comple...

Filter: api.twitter

Overview | Contents | Summary | Chart | Notes

Name	Value
URL	https://api.twitter.com/graphql/yL4KIHnJPXt-JUpRDrBDDw/CreateTweet
Status	Complete
Response Code	200
Protocol	HTTP/2.0

We look in the browser and see the result: the client displays an error on the GUI, which means that the request was processed correctly. (text on the screenshot: *Your tweet length exceeds the limit*).