Project information

Created on
February 16, 2026

updated docker compose to use only 3 backend replicas
Marco Schneider authored 40 minutes ago

Name	Last commit	Last update
.claude	apiified everything	1 month ago
.ddev	updated docker setup to include websocket	1 month ago
assets	updated game mode handling	1 month ago
bin	initial project setup + README.md	1 month ago
config	updated docker image versions and docu…	2 weeks ago
docker	add env and debug check again	1 month ago
docs	documented performance tests more	2 weeks ago
migrations	implemented refresh token logic	1 month ago
public	initial project setup + README.md	1 month ago
src	updated docker compose to use only 3 ba…	40 minutes ago
templates	added share link	3 weeks ago
traefik	documented performance tests more	2 weeks ago
websocket	updated docker setup to include websocket	1 month ago
.dockerignore	production docker setup	1 month ago
.editorconfig	initial project setup + README.md	1 month ago
.env	implemented refresh token logic	1 month ago
.env.dev	initial project setup + README.md	1 month ago
.env.docker.example	documented performance tests more	2 weeks ago
.gitignore	production docker setup	1 month ago
DOCKER.md	documented performance tests more	2 weeks ago
Dockerfile	updated docker compose to use only 3 ba…	40 minutes ago
LICENSE	initial project setup + README.md	1 month ago
README.md	updated docker compose to use only 3 ba…	40 minutes ago
composer.json	updated docker image versions and docu…	2 weeks ago
composer.lock	updated docker image versions and docu…	2 weeks ago
docker-compose.stack.yml	updated docker compose to use only 3 ba…	40 minutes ago
docker-compose.yml	update docker networking	3 weeks ago

Name	Last commit	Last update
k6-stress-test.js	updated docker image versions and docu...	2 weeks ago
symfony.lock	extended configuration	1 month ago

README.md

Dart Board Counter - Project Idea and Initial Plan

Installation

Please read the installation at Docker Setup.

Project Idea

Build a focused web app for tracking X01 dart games (301/501/etc.) with a fast input flow, clear scoreboard, and reliable game state handling.

Core concept:

- Users create a game mode and players.
- The app guides each throw (number, multiplier, bull, out).
- Scores, turns, bust rules, and win conditions are updated automatically.
- Finished and aborted games remain visible for review.

The goal is to replace paper scoring with a digital counter that is fast enough to use during real play.

How to stress test

Before these tests can be run, the docker setup must be completed and running. the port in the commands below must be changed to the port of the docker setup.

Preparations

First we need to get a JWT token from the API to make authenticated requests. The following command authenticates and sets the JWT environment variable or the current shell session.

```
export JWT=$(curl -s -X POST http://localhost:8088/api/login_check \
  -H "Content-Type: application/json" \
  -d '{"email":"demo@example.com","password":"demo1234"}' \
  | grep -o '"token":"[^"]*"' | cut -d'"' -f4)
```

Test if the JWT is set correctly.

```
curl -H "Authorization: Bearer $JWT" http://localhost:8088/api/me
```

This request loads the user profile and returns the user's information. If the response looks like this, you are ready.

```
{"user":{"id":"019c9e77-3e19-7d66-bd7e-24b48aa23263","email":"demo@example.com","roles":["ROLE_USER"],"crea
```

Stress Test Results

Approximate 1000 req/s with hey

With hey you usually cannot directly set “1000 requests per second”, you control concurrency instead. The resulting requests/sec is whatever the system can sustain. `concurrency ≈ requests per second × normal average response time` With an average endpoint latency of ≈ `200ms = 0.20s` `1000rps * 0.20s = 200 concurrency`. So with 200 concurrent connections you can sustain 1000 requests per second when the average response time is 200ms.

hey test 1 (local)

with 50 concurrent connections for 10 seconds:

```
hey -z 10s -c 50 \
  -H "Authorization: Bearer $JWT" \
  http://localhost:8088/api/me
```

▸ **Results...**

```
Summary:
  Total:        10.0271 secs
  Slowest:      0.3418 secs
  Fastest:      0.0048 secs
  Average:      0.0434 secs
  Requests/sec: 1149.4795


Response time histogram:
  0.005 [1]     |
  0.038 [5203]  |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.072 [5472]  |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.106 [681]   |■■■■■
  0.140 [118]   |■
  0.173 [40]    |
  0.207 [10]    |
  0.241 [0]     |
  0.274 [0]     |
  0.308 [0]     |
  0.342 [1]     |


Latency distribution:
  10%% in 0.0213 secs
  25%% in 0.0301 secs
  50%% in 0.0405 secs
  75%% in 0.0527 secs
  90%% in 0.0676 secs
  95%% in 0.0789 secs
  99%% in 0.1172 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0000 secs, 0.0000 secs, 0.0087 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0045 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0030 secs
  resp wait:    0.0408 secs, 0.0044 secs, 0.3377 secs
  resp read:    0.0026 secs, 0.0000 secs, 0.0946 secs

Status code distribution:
  [200] 11526 responses
```

hey test 2 (local)

with 200 concurrent connections for 10 seconds

```
hey -z 10s -c 200 \
  -H "Authorization: Bearer $JWT" \
  http://localhost:8088/api/me
```

▸ Results...

```
Summary:
  Total:        14.2676 secs
  Slowest:      6.9568 secs
  Fastest:      0.0049 secs
  Average:      0.1743 secs
  Requests/sec: 833.4268


Response time histogram:
  0.005 [1]     |
  0.700 [11571] |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  1.395 [7]     |
  2.090 [9]     |
  2.786 [50]    |
  3.481 [73]    |
  4.176 [30]    |
  4.871 [52]    |
  5.566 [46]    |
  6.262 [34]    |
  6.957 [18]    |


Latency distribution:
  10%% in 0.0161 secs
  25%% in 0.0232 secs
  50%% in 0.0396 secs
  75%% in 0.0966 secs
  90%% in 0.1878 secs
  95%% in 0.2630 secs
  99%% in 4.5990 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0000 secs, 0.1019 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0038 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0058 secs
  resp wait:    0.1703 secs, 0.0045 secs, 6.8203 secs
  resp read:    0.0033 secs, 0.0000 secs, 0.3296 secs

Status code distribution:
  [200] 11891 responses
```

hey test 3 (local)

with 500 concurrent connections for 10 seconds

```
hey -z 10s -c 500 \
  -H "Authorization: Bearer $JWT" \
  http://localhost:8088/api/me
```

▸ Results...

```
Summary:
  Total:        20.0727 secs
  Slowest:      15.0019 secs
  Fastest:      0.0052 secs
  Average:      0.5070 secs
  Requests/sec: 613.5208


Response time histogram:
  0.005 [1]     |
  1.505 [11804] |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  3.005 [12]    |
  4.504 [24]    |
  6.004 [55]    |
  7.504 [124]   |
  9.003 [94]    |
  10.503 [102]  |
  12.003 [63]   |
  13.502 [0]    |
  15.002 [36]   |


Latency distribution:
  10%% in 0.0465 secs
  25%% in 0.0803 secs
  50%% in 0.1497 secs
  75%% in 0.2619 secs
  90%% in 0.3810 secs
  95%% in 0.5835 secs
  99%% in 9.9373 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0022 secs, 0.0000 secs, 0.1528 secs
  DNS-lookup:   0.0001 secs, 0.0000 secs, 0.0060 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0056 secs
  resp wait:    0.4972 secs, 0.0049 secs, 14.8492 secs
  resp read:    0.0065 secs, 0.0000 secs, 0.5672 secs

Status code distribution:
  [200] 12315 responses
```

k6 1000req/s test

```
URI=http://localhost:8088 k6 run k6-stress-test.js
```

```
~/Sites/symfony-dart-helper (production) » URI=http://localhost:8088 k6 run k6-stress-test.js

         /\      Grafana   /‾‾/
    /\  /  \     |\  __   /  /
   /  \/    \    | |/ /  /   ‾‾\
  /          \   |   (  |  (‾)  |
 / __________ \  |_|\_\  \_____/

     execution: local
        script: k6-stress-test.js
        output: -

     scenarios: (100.00%) 1 scenario, 1000 max VUs, 40s max duration (incl. graceful stop):
              * constant_rps: 1000.00 iterations/s for 10s (maxVUs: 200-1000, gracefulStop: 30s)


  TOTAL RESULTS

    checks_total.......: 9699    702.911098/s
    checks_succeeded...: 100.00% 9699 out of 9699
    checks_failed......: 0.00%   0 out of 9699

    ✓ status is 200

    HTTP
    http_req_duration..............: avg=292.42ms min=3.84ms med=50.72ms max=7.27s p(90)=208.64ms p(95)=318.15ms
      { expected_response:true }...: avg=292.42ms min=3.84ms med=50.72ms max=7.27s p(90)=208.64ms p(95)=318.15ms
    http_req_failed................: 0.00%  0 out of 9699
    http_reqs......................: 9699   702.911098/s

    EXECUTION
    dropped_iterations.............: 303    21.959178/s
    iteration_duration.............: avg=292.57ms min=3.98ms med=50.92ms max=7.27s p(90)=208.82ms p(95)=318.31ms
    iterations.....................: 9699   702.911098/s
    vus............................: 9      min=9         max=367
    vus_max........................: 454    min=200       max=454

    NETWORK
    data_received..................: 4.9 MB 352 kB/s
    data_sent......................: 5.8 MB 419 kB/s



running (13.8s), 0000/0454 VUs, 9699 complete and 0 interrupted iterations
constant_rps ✓ [======================================] 0000/0454 VUs  10s  1000.00 iters/s
```