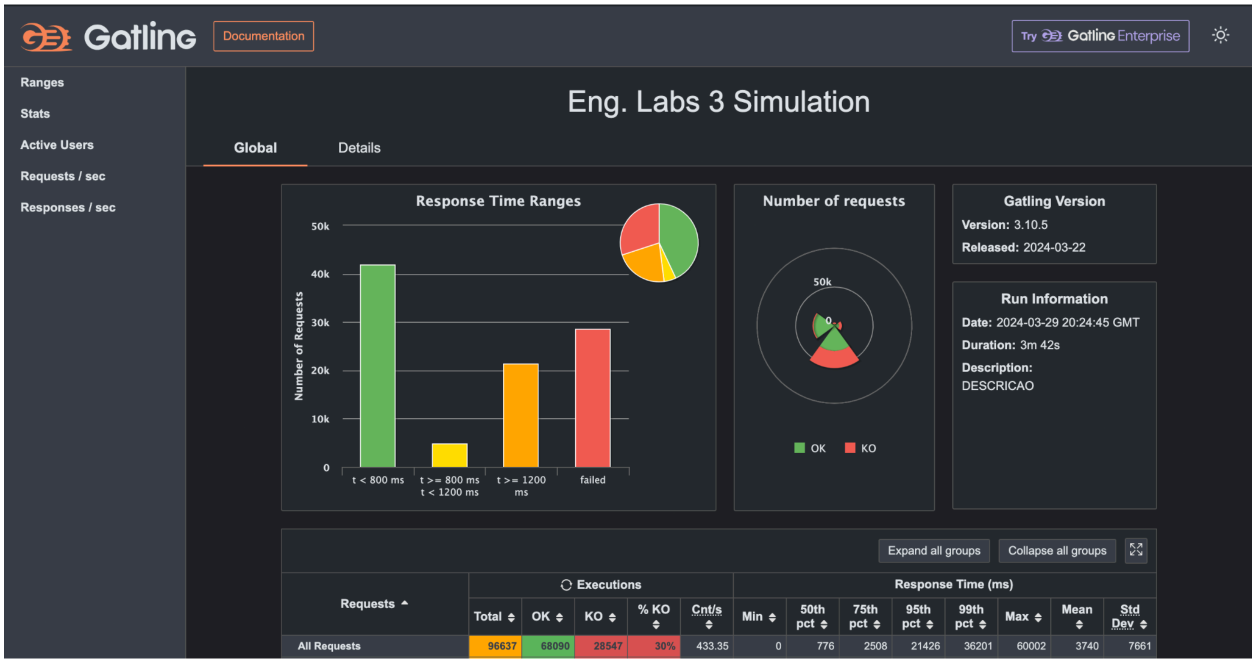
Qwasar Master's of Science in Computer Science Programs

Engineering Lab 3
API Under Stress



Eng. Labs 3 Simulation

Gatling Version
Version: 3.10.5
Released: 2024-03-22

Run Information
Date: 2024-03-29 20:24:45 GMT
Duration: 3m 42s
Description:
DESCRICAO

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	96637	68090	28547	30%	433.35	0	776	2508	21426	36201	60002	3740	7661

Problem Statement

With the surge of digital services, the demand for robust backend systems has never been higher. These systems are the backbone of web services, applications, and online platforms. Ensuring their reliability and performance, especially under stress, is crucial for maintaining user satisfaction and service continuity.

Objective

To design, develop, and test a resilient API that can effectively handle high loads, unexpected spikes in traffic, and various types of stress scenarios, ensuring minimal service disruption and consistent performance.

Solution

The APIs will utilize databases such as Postgres, MySQL, or MongoDB, ensuring robust data management and retrieval capabilities. For deployment purposes, the API will be containerized using `docker-compose`, and to emulate a real-world environment, specific CPU and memory constraints will be imposed during the deployment and test process.

Gatling will be used for effective load testing and performance measurement. This will ensure that we can simulate high-concurrency scenarios and rigorously stress-test the API.

Routes:

- POST /warrior – creates warrior;

name	mandatory, string max 100 characters.
dob(Date Of Birth)	mandatory, string to date in format AAAA-DD-MM (year, day, month).
fight_skills	Mandatory, list/array with max 20 entries of strings max 250 characters

API must return status code 201 - created with header "Location: /name/[:id]" where [:id] is the id – in UUID format of any version – from the client just created. Body and return is at developer discretion.

Requisition Example:

```
Unset
{
  "name" : "Master Yoda",
  "dob" : "1970-01-01",
  "fight_skills" : ["BJJ", "KungFu", "Judo"]
}
```

- GET /warrior/[:id] – return warrior created with corresponding id;

```
Unset
{
    "id" : "f7379ae8-8f9b-4cd5-8221-51efe19e721b",
    "name" : "Master Yoda",
    "dob" : "1970-01-01",
    "fight_skills" : ["BJJ", "KungFu", "Judo"]
}
```

Return 200 - Ok or 404 - Not Found.

- GET /warrior?t=[:search term] – search warrior attributes

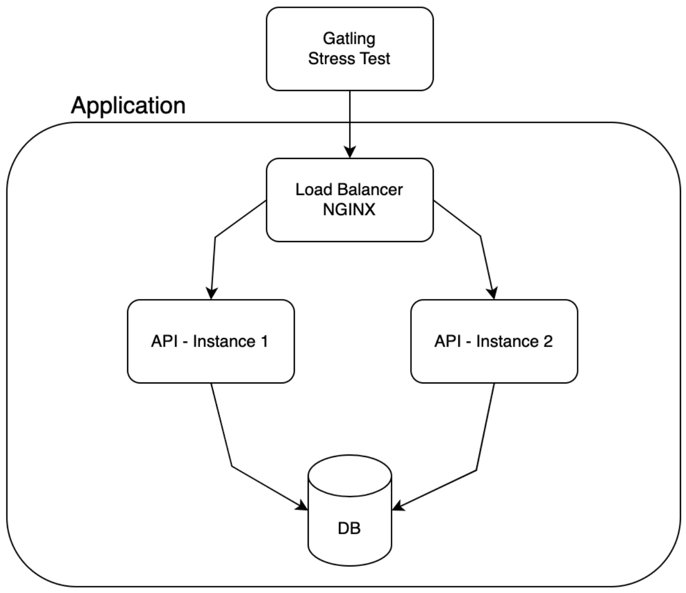
```
Unset
[{
    "id" : "f7379ae8-8f9b-4cd5-8221-51efe19e721b",
    "name" : "Master Yoda",
    "dob" : "1970-01-01",
    "fight_skills" : ["BJJ", "KungFu", "Judo"]
},
{
    "id" : "g1245be9-f82s-6fe9-2281-64abc29d891c",
    "name" : "Vader",
    "nascimento" : "1989-09-09",
    "fight_skills" : ["Sambo", "Capoeira"]
}]
```

Always return 200, even with an empty list. Results don't need to be paginated. It should return the first 50 entries. If no search term passed, the route must return 400 - Bad Request.

- GET /counting-warriors – count registered warriors;

Route just to verify proper test operations, doesn't need to have any performance concern, and it'll be not the subject of evaluation.

Overall Architecture and Constraints



- **CPU Constraints: 1.5 CPUs**
- **Memory Constraints: 3.0 GB**

References

- Gatling & Gatling Repo
- Nginx
- Docker
- MongoDB / PostgreSQL / MariaDB / MySQL / DuckDB / Neo4j
- REST API

Stack and Technologies Used

- Docker
- C++, Rust, Python, Java, Go, Ruby, PHP, Scala, Clojure, OCaml, Node.js
- Nginx
- Gatling

Deliverables

- **README**
- **Docker-compose:**
  - **Full build**
  - **Memory/CPU constraints**
- **Source code**
  - **API**

    - **Database configuration**
    - **Dependencies (package.json, Gemfile, etc.)**
- **(optional) configuration file for nginx (or choosen web server)**

Proposed Schedule

Sprint	**Version**	**Deadline**	**Deliverables & Activities**
Initial Project Presentation	**N.A.**	04/02 (Tue)	Project Presentation Q&A
First Version	**1.0**	04/23 (Tue)	API with full spec: 1. Source 2. Build 3. Instructions API will be tested, and Groups will receive the first Gatlin report, along with the test files and payload.
Second Iteration	**2.0**	04/30 (Tue)	Changelog, listing changes from **v1.0** to the current one. Groups will run tests on **v2.0**, and show results.
Third Iteration	**3.0**	05/07 (Tue)	Changelog, listing changes from **v2.0** to the current one. Groups will run tests on **v3.0**, and show results.
Project Submission and Presentation	**Final Version**	05/14 (Tue)	Final project version, with final test results. Groups can prepare material to support project presentation.

Outcome

By the end of the "API Under Stress" lab, students will not only have hands-on experience designing and optimizing backend systems for stress but will also understand the importance of proactive monitoring and agile response to issues.

- **Scalability and Performance**: How well does the system scale with increasing load? What is the maximum load the system can handle without significant degradation?
- **Resilience and Reliability**: How well does the system recover from failures? Are there any single points of failure?
- **Optimization and Improvements**: After identifying bottlenecks, how effective were the optimizations made?
- **Documentation and Reporting:** Is the documentation clear, comprehensive, and useful for future reference? Is the report insightful and well-organized?
- **Presentation Skills**: Clarity of communication, presentation structure, and ability to answer technical questions.