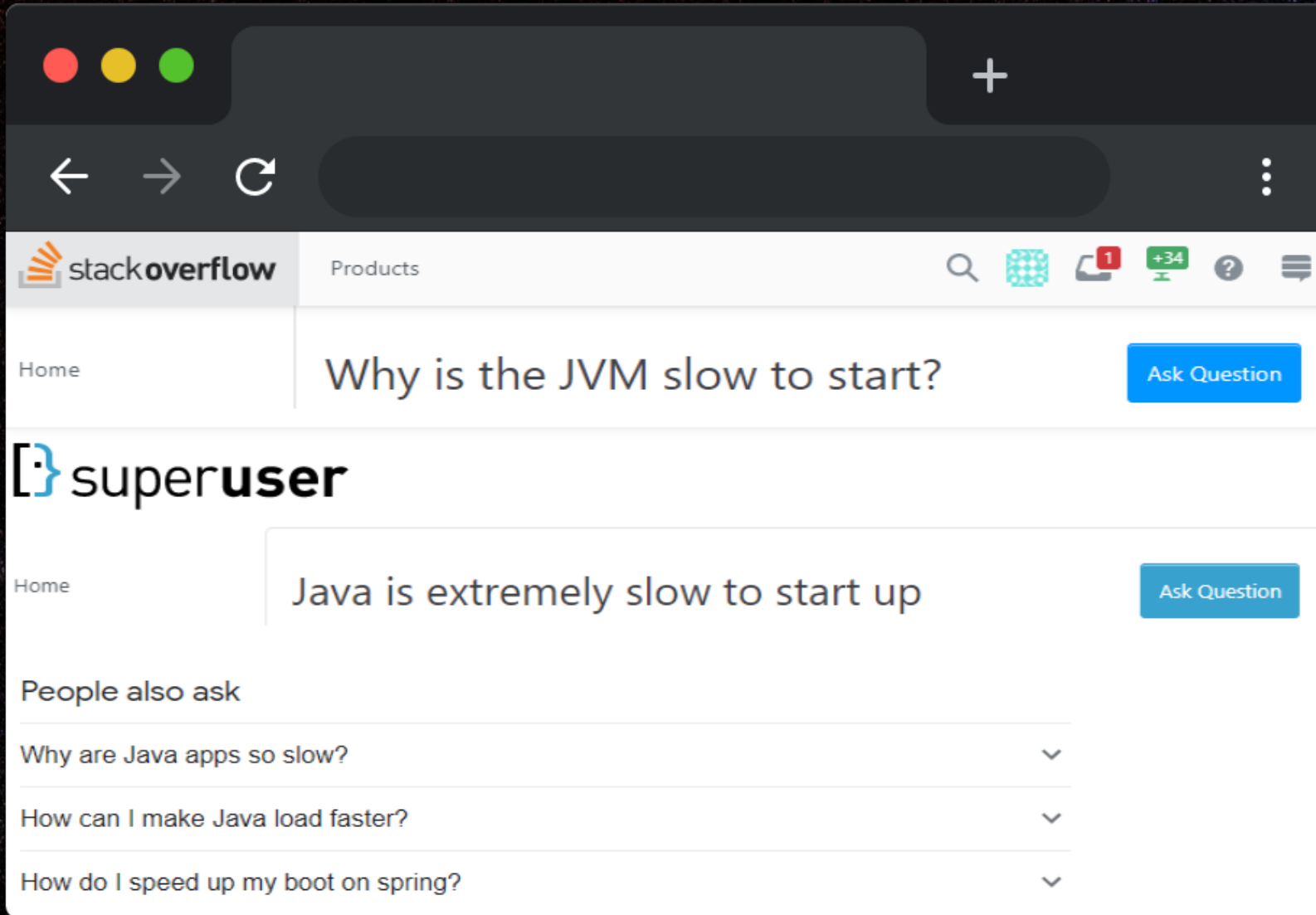


About

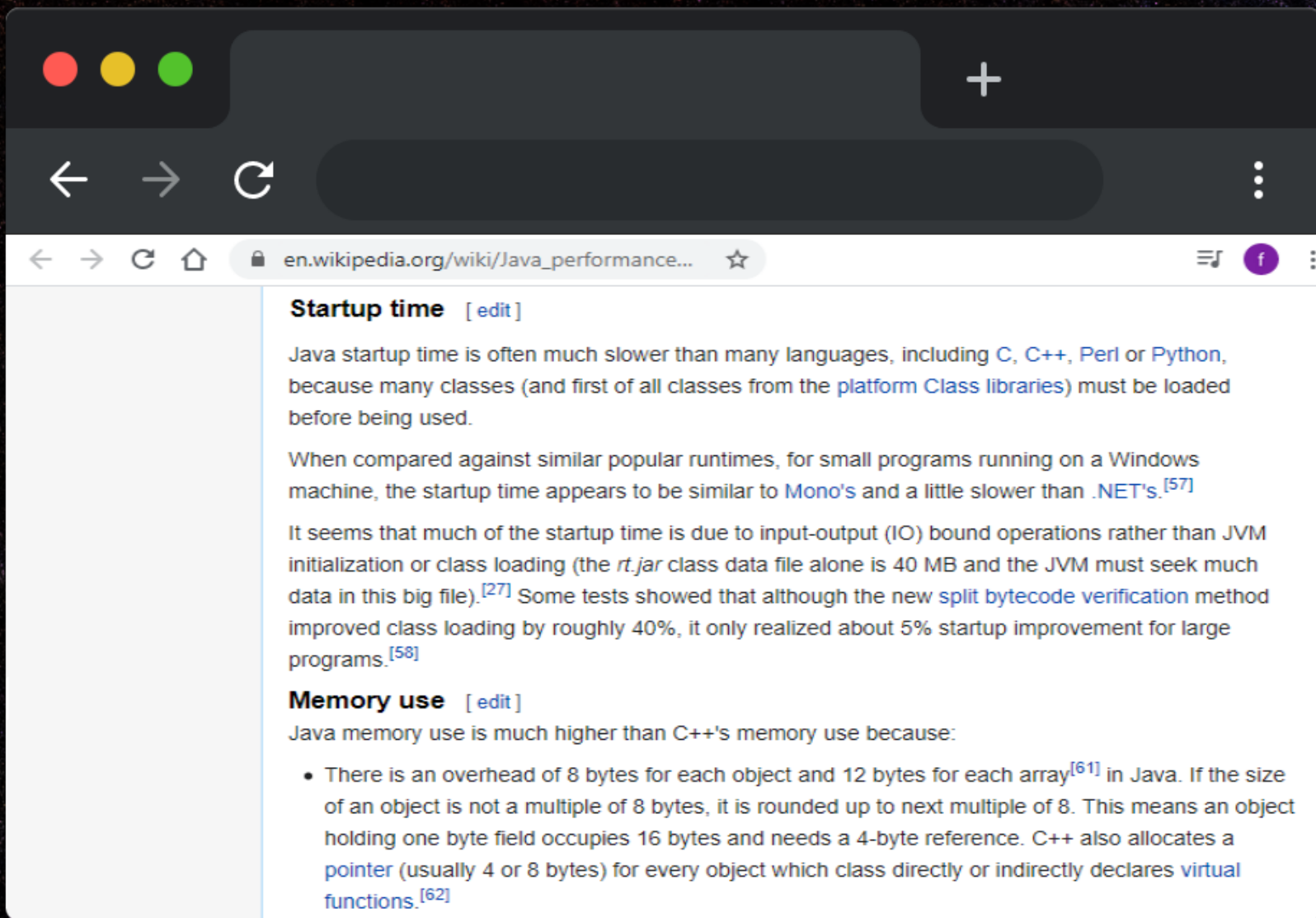
GraalVM™

adesso

The question is



What we can do?



Startup time [\[edit \]](#)

Java startup time is often much slower than many languages, including [C](#), [C++](#), [Perl](#) or [Python](#), because many classes (and first of all classes from the [platform Class libraries](#)) must be loaded before being used.

When compared against similar popular runtimes, for small programs running on a Windows machine, the startup time appears to be similar to [Mono's](#) and a little slower than [.NET's](#).^[57]

It seems that much of the startup time is due to input-output (IO) bound operations rather than JVM initialization or class loading (the *rt.jar* class data file alone is 40 MB and the JVM must seek much data in this big file).^[27] Some tests showed that although the new [split bytecode verification](#) method improved class loading by roughly 40%, it only realized about 5% startup improvement for large programs.^[58]

Memory use [\[edit \]](#)

Java memory use is much higher than C++'s memory use because:

- There is an overhead of 8 bytes for each object and 12 bytes for each array^[61] in Java. If the size of an object is not a multiple of 8 bytes, it is rounded up to next multiple of 8. This means an object holding one byte field occupies 16 bytes and needs a 4-byte reference. C++ also allocates a [pointer](#) (usually 4 or 8 bytes) for every object which class directly or indirectly declares [virtual functions](#).^[62]

So what is the solution?

Wait.
So you just save it,
And your code is running?
And it's Java?!



I know, right?
SUPERSONIC JAVA, FTW!



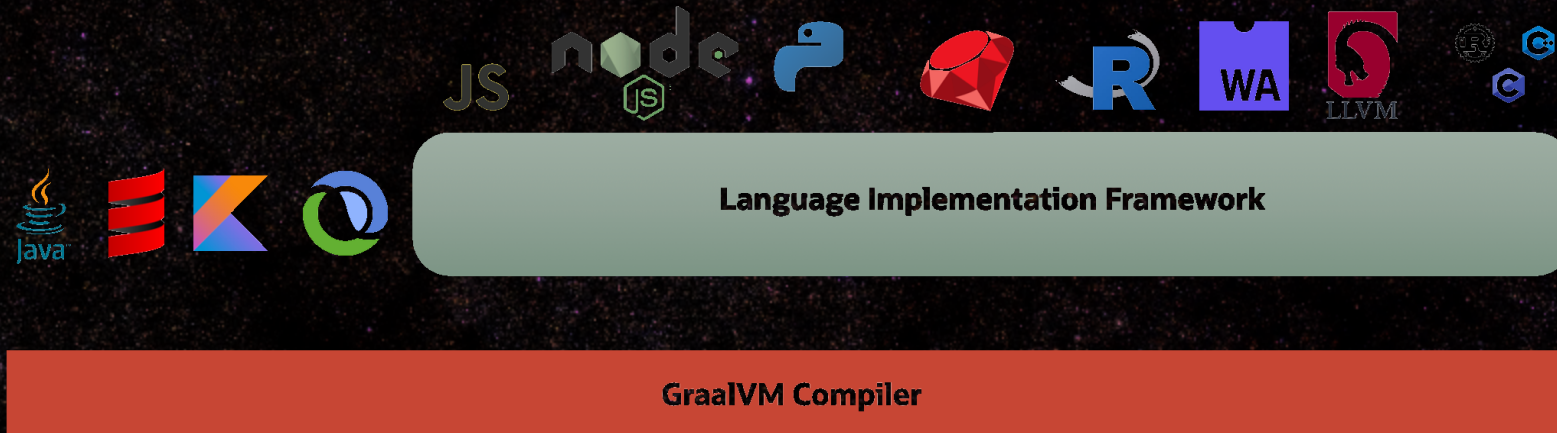
What is GraalVM ?

GraalVM is a **high-performance** JDK distribution designed to **accelerate the execution of applications** written in **Java** and other JVM languages along with support for **JavaScript**, **Ruby**, **Python** and ***a number of other popular languages***. **GraalVM**'s polyglot capabilities make it possible to mix multiple programming languages in a single application while ***eliminating foreign language call costs***.

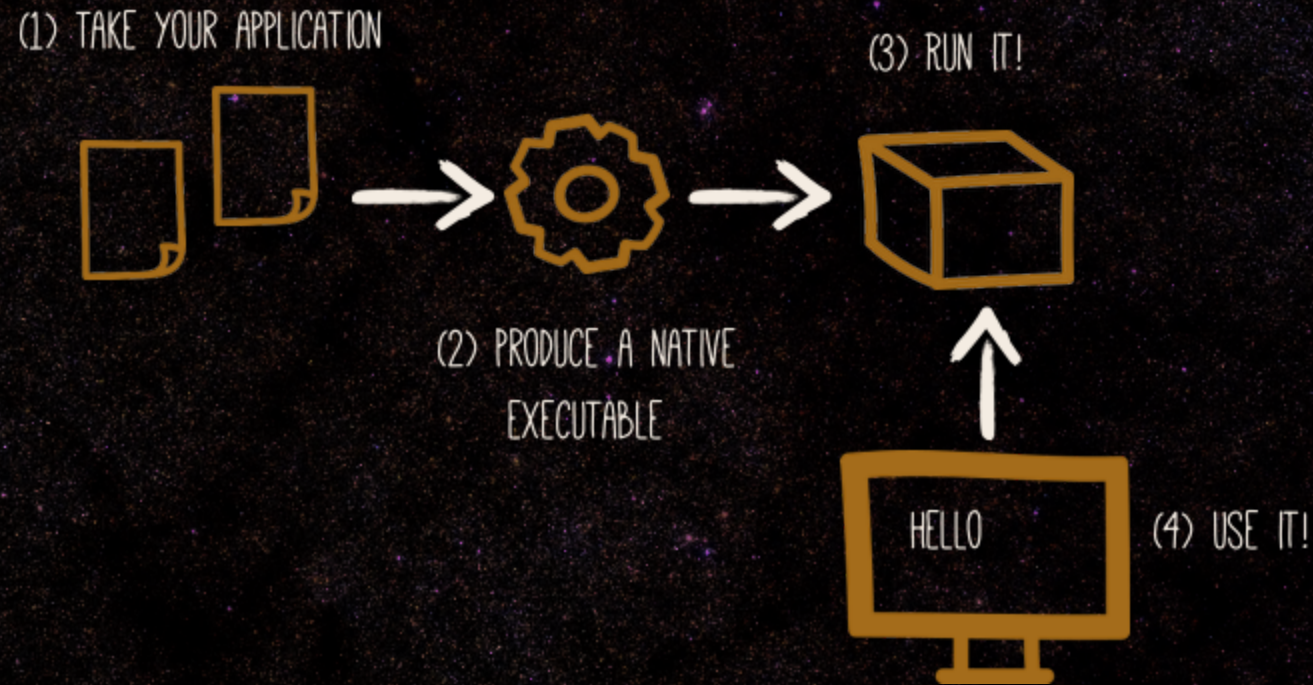
What can GraalVM give me?

- Mix multiple programming languages in a single application
- Native executables
- Amazingly fast boot time
- Incredibly low RSS memory (not just heap size!)
- Instant (relatively) scale up and high density memory utilization in container orchestration platforms like Kubernetes.

GraalVM Architecture



Native Image



The **native executable** for our application will contain the **application code**, **required libraries**, **Java APIs**, and **a reduced version of a VM**. The smaller VM base improves the **startup time** of the application and produces a **minimal disk footprint**.

Truffle language implementation framework

The Truffle language implementation framework (henceforth “Truffle”) is an open source library for building tools and programming languages implementations as interpreters for self-modifying Abstract Syntax Trees. Together with the open source GraalVM compiler,

Truffle represents a significant step forward in programming language implementation technology in the current era of dynamic languages.

Supported Frameworks and Tech

- Spring Framework (Experimental)
- Quarkus
- Play Framework
- Camel
- Prometheus
- JavaFX

...

Disadvantages

- The main downside of this approach is the platform-dependent native code.

That means you need to compile source code for linux/windows etc.

Meanwhile ??

SystemAdmin01: **server_api.so doesn't work on my server !**

Customer\$\$\$__: **customer_api.exe doesn't work on my pc !**

DevGuy42_____: **api.jar works on my workstation !**

F1rat_____: **In some cases, problems may arise from the libraries or build scripts used. In other words, it is necessary to pay attention to tests.**

SystemAdmin01: **himm, server_api.so doest work on my server !**

.....

DevGuy42 writing ...

Tests : Sure

```
package org.acme.quickstart;

import io.quarkus.test.junit.NativeImageTest;

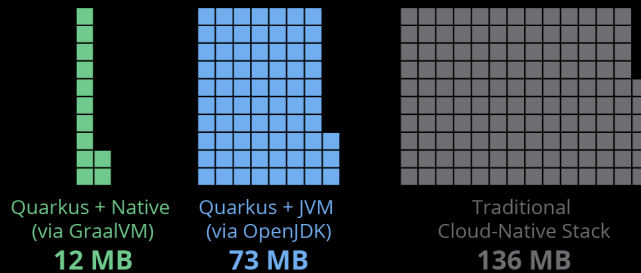
@NativeImageTest
public class NativeGreetingResourceIT extends GreetingResourceTest {
    // Run the same tests
}
```


Performance

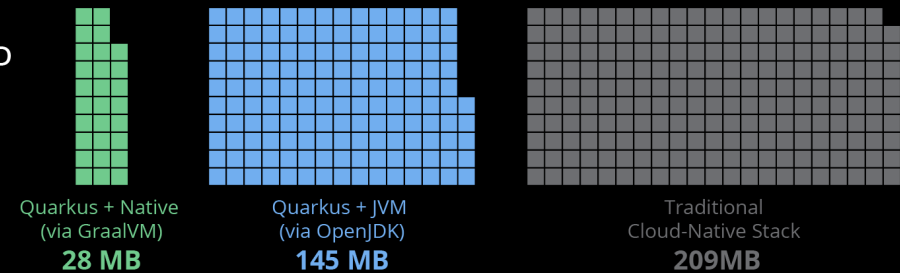
Memory (RSS) in Megabytes*

*Tested on a single-core machine

REST



REST
+ CRUD



BOOT + First Response Time

REST

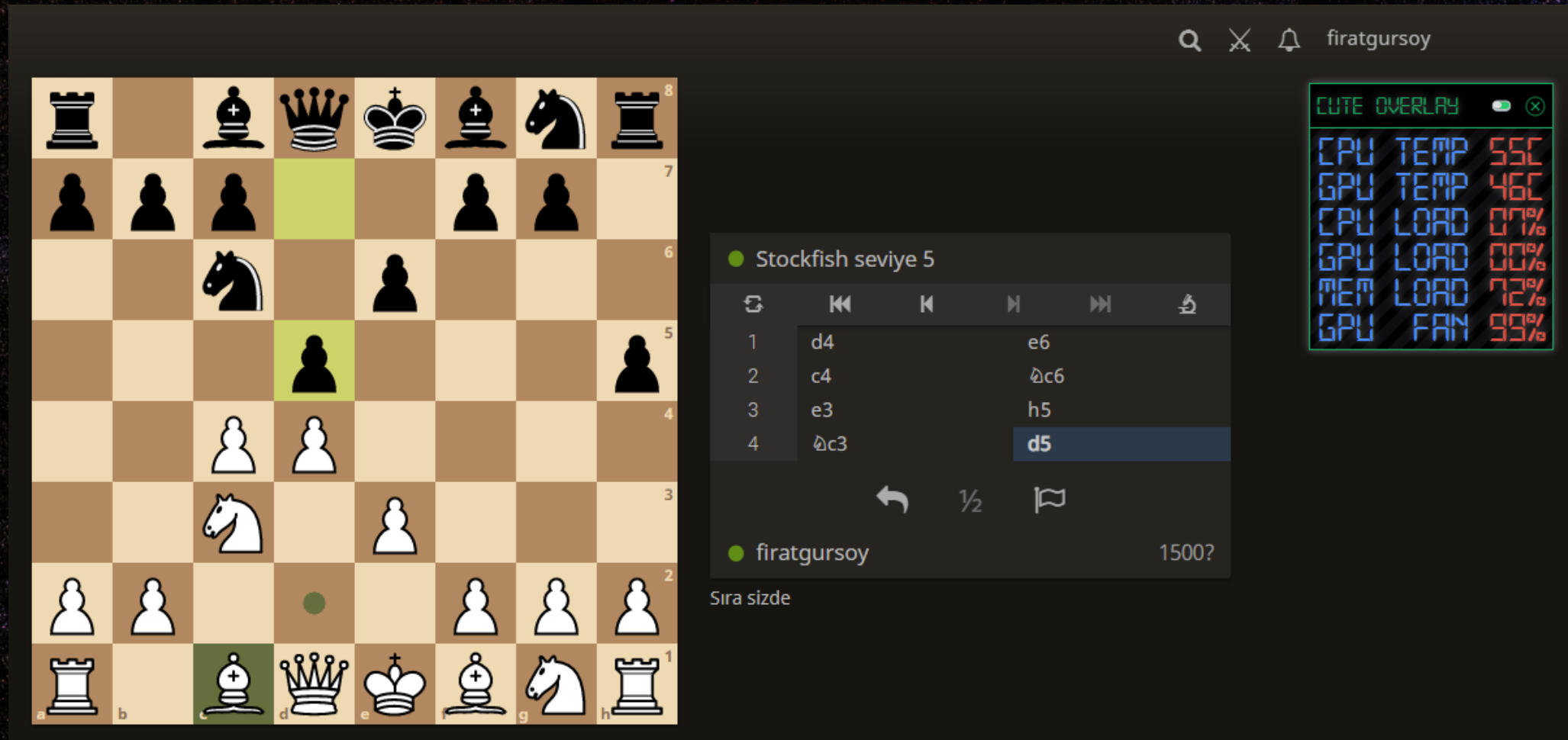


REST
+ CRUD



Quarkus Test

After this experience i made a demo application for my requirement.
It's really reducing startup and memory footprint dramatically. :)



Spacial thanks to

- Oracle
- Quarkus
- Spring
- JavaFX / GluonHQ
- Google and Wiki

