****

**(Proposta do Trabalho Prático)**

**Aplicação Multithreading (WEB Server)**

Github:

<https://github.com/Africano19/SIstemasOperativos>

Hélio José (20190928) e Rúben Passarinho (20200095)

Licenciatura de Engenharia Informática

IADE – Faculdade de Design Tecnologias e Comunicação

SIstemas Operativos

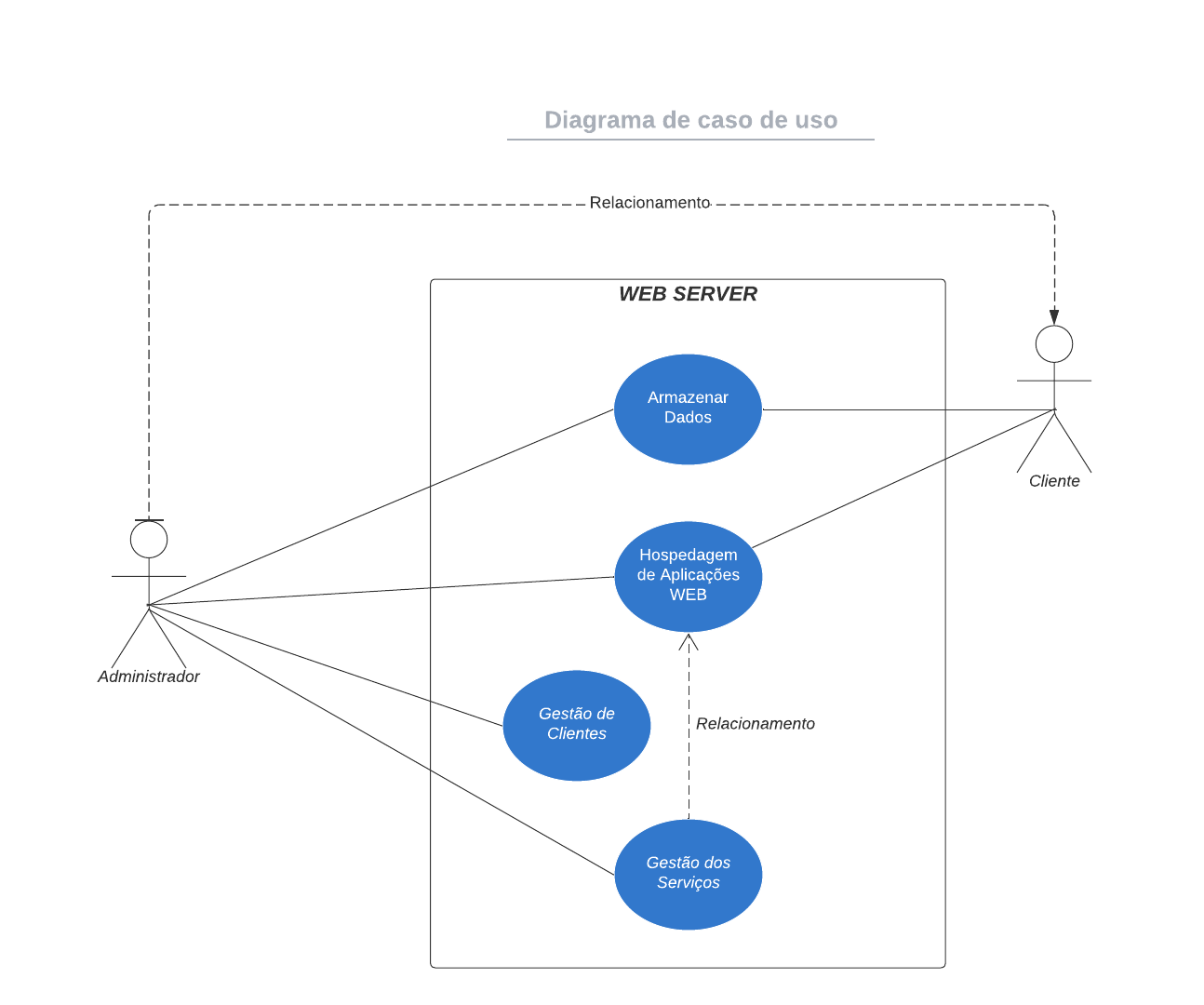
Professor Pedro Rosa

12 de maio 2023

**Descrição do problema e motivação do trabalho a realizar.**

Com a evolução computacional obtivemos um grande aumento de comunicações entre dispositivos computacionais, entre essas comunicações temos a internet juntamente com os servidores web.

O Multithreading veio no contexto de haver a necessidade de um programa ou sistema operacional suportar mais de um fio de execução ao mesmo tempo, de modo aumentar a sua performance e tempo de resposta para cada fio de execução.

**Diagrama de casos de uso**

**Solução a implementar**

Para a resolução do nosso problema iremos criar um web server simples, devido ao facto de o mesmo ser alvo de execução de várias tarefas como por exemplo o cliente pode requerer dados, enviar dados, solicitar a execução de processos e também executá-los. As tarefas dos servidores web têm o envio de dados para o cliente, acesso a base de dados, e execução de processos.

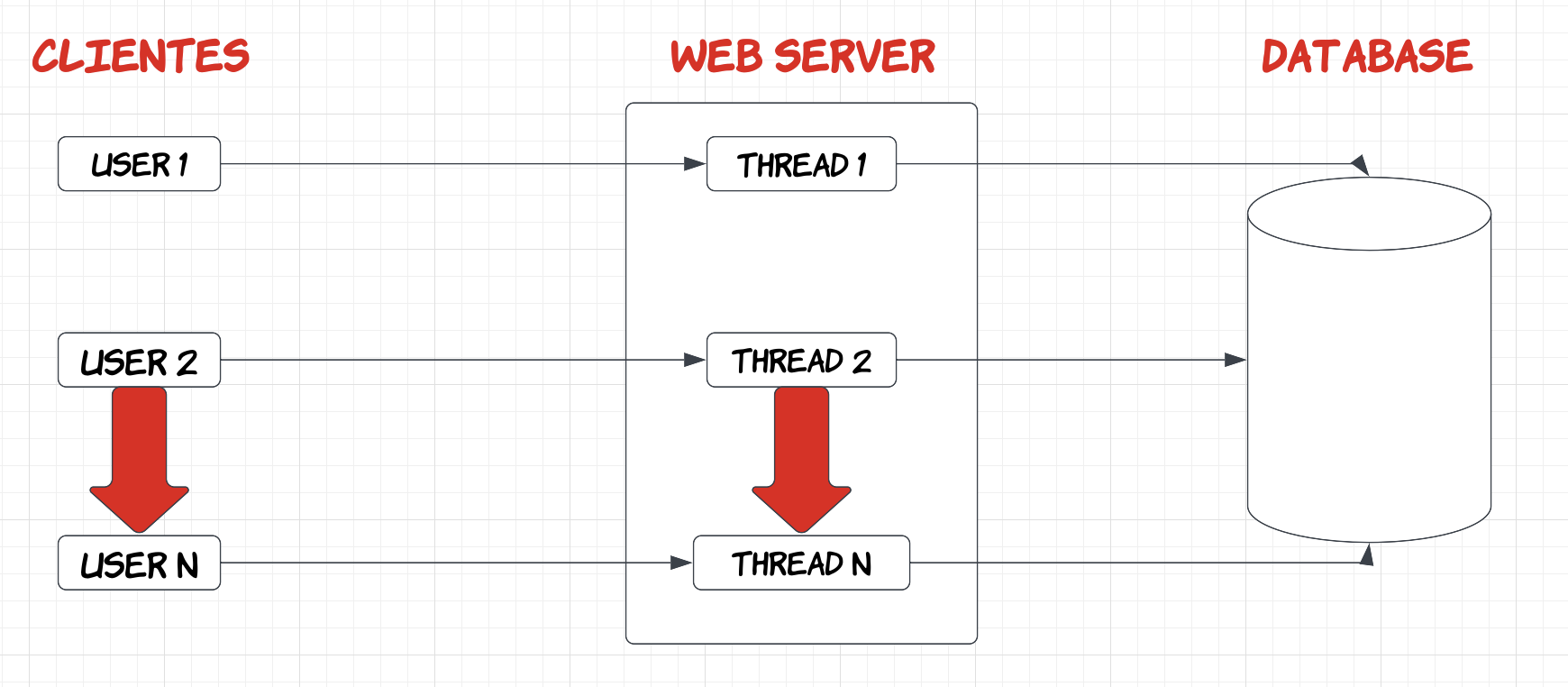
**Enquadramento nas áreas da Unidade Curricular**

Este projeto enquadra-se à cadeira de Sistemas operativos, devido ao facto do Multithreading ser uma funcionalidade bastante comum e essencial em sistemas operativos de forma a melhorar o desempenho, permitindo que múltiplas tarefas sejam executadas em paralelo/simultâneo.

**Requisitos Técnicos para o desenvolvimento do projeto**

* Sistema baseado em Multithreading;
* Prevenção da possível corrupção dos dados enquanto houver duas leituras de threads diferentes ao mesmo tempo;
* Imagens Docker para as instâncias de webserver;
* Nomad como alternativa aos Kubernets;
* Nginx (Load balancer) para equilibrar a carga entre os webservers;
* Automatização da criação e destruição de containers de webserver com base nas necessidades da aplicação;
* As informações sobre a saúde e o desempenho dos web services serão monitoradas através da ferramentas de monitoramento (Prometheus);

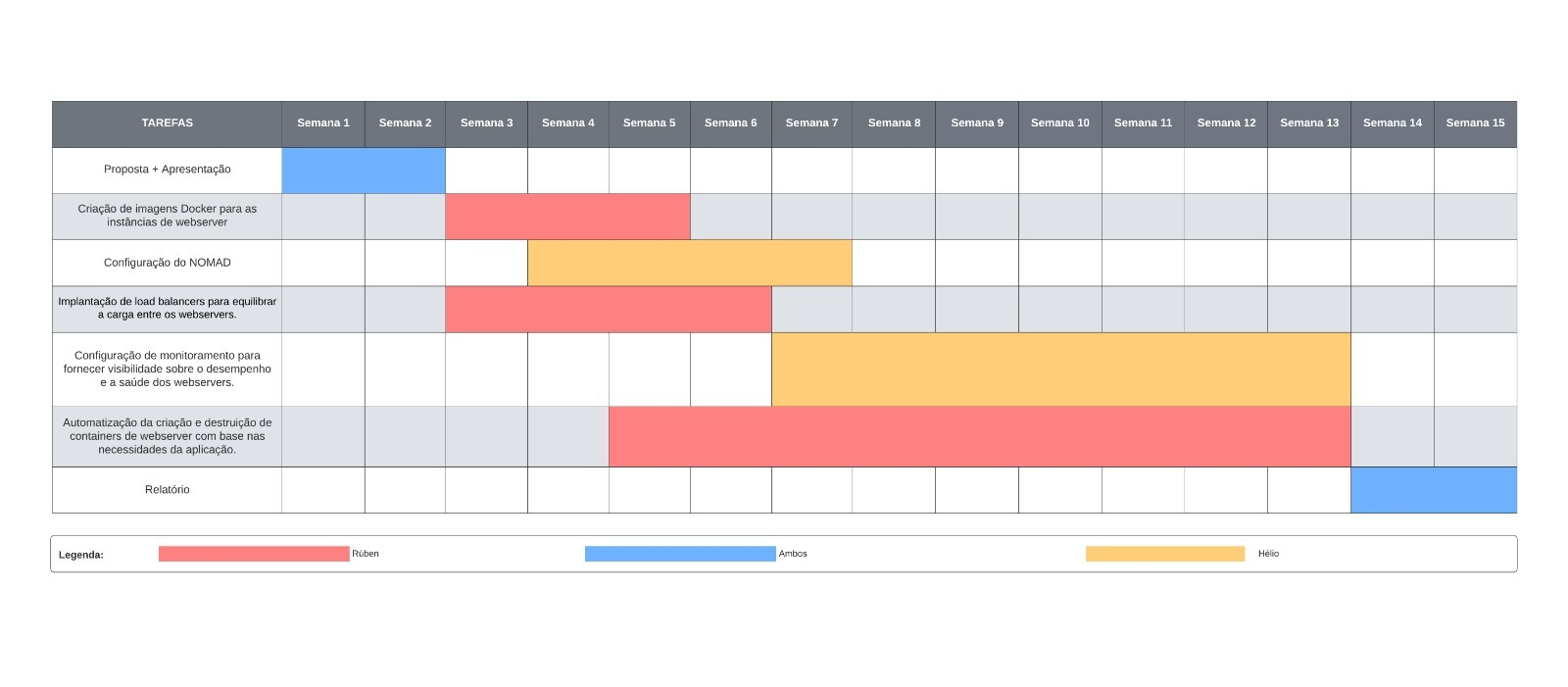
**Arquitetura da Solução**



**Tecnologias a utilizar**

* Linguagens Utilizadas: Java;
* Sistema Operativo a Base de Linux;
* Nomad;

**Planeamento e calendarização**

****

**Descrição do Projeto**

Este projeto é um servidor multithreaded criado para manipular múltiplas conexões de cliente de forma eficiente e segura. Utiliza a biblioteca de soquetes Java para criar um servidor e aceitar conexões, e a biblioteca ExecutorService para gerenciar threads.

**Funcionalidades Principais**

1. **Multithreading**: Para cada nova conexão de cliente, é criado um novo thread. Isso permite que o servidor manipule várias conexões de cliente simultaneamente.

2. **Criação de Trabalhos Nomad**: Para cada nova conexão de cliente, o servidor cria um novo trabalho na plataforma Nomad.

5. **Segurança SSL**: O servidor utiliza uma fábrica de soquetes SSL para criar um servidor seguro que aceita apenas conexões criptografadas.

**Estrutura do Código**

O código é composto por duas classes principais: MultithreadedServer e ServerThread.

- MultithreadedServer: Esta é a classe principal que contém o método `main()`. Ele cria o servidor, aceita conexões de cliente, cria e gerencia threads.

- ServerThread: Esta é uma classe interna que representa um thread de manipulação de clientes. Cada instância de ServerThread é responsável por manipular uma única conexão de cliente.

**Melhorias Realizadas**

As melhorias realizadas durante o desenvolvimento do projeto incluem:

1. **Tratamento de Exceções:** O código foi melhorado para lidar com várias exceções que podem ocorrer durante a execução, como erros de I/O e soquetes já em uso.

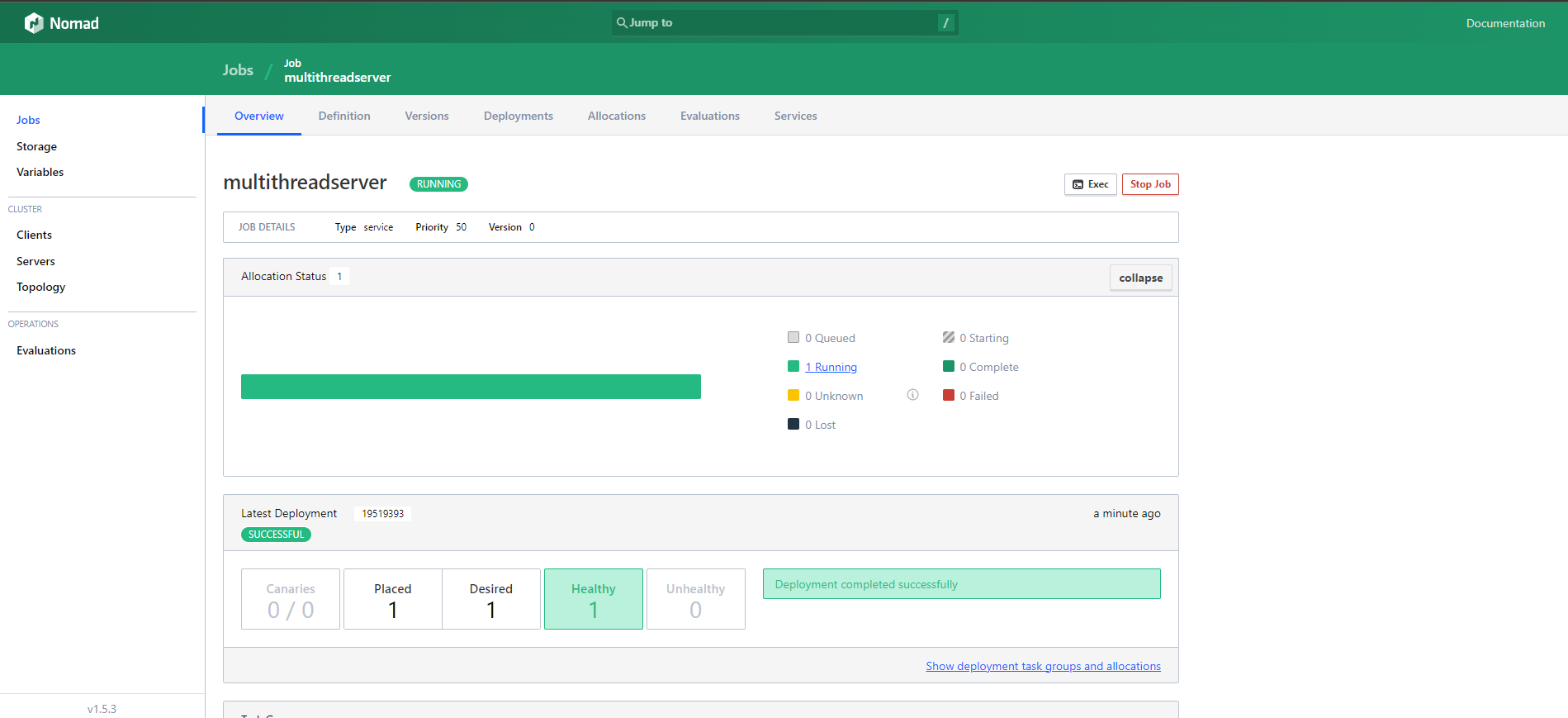
2. **Recursos de Limpeza**: O código foi melhorado para limpar recursos corretamente após o uso, incluindo fechamento de soquetes e interrupção de threads.

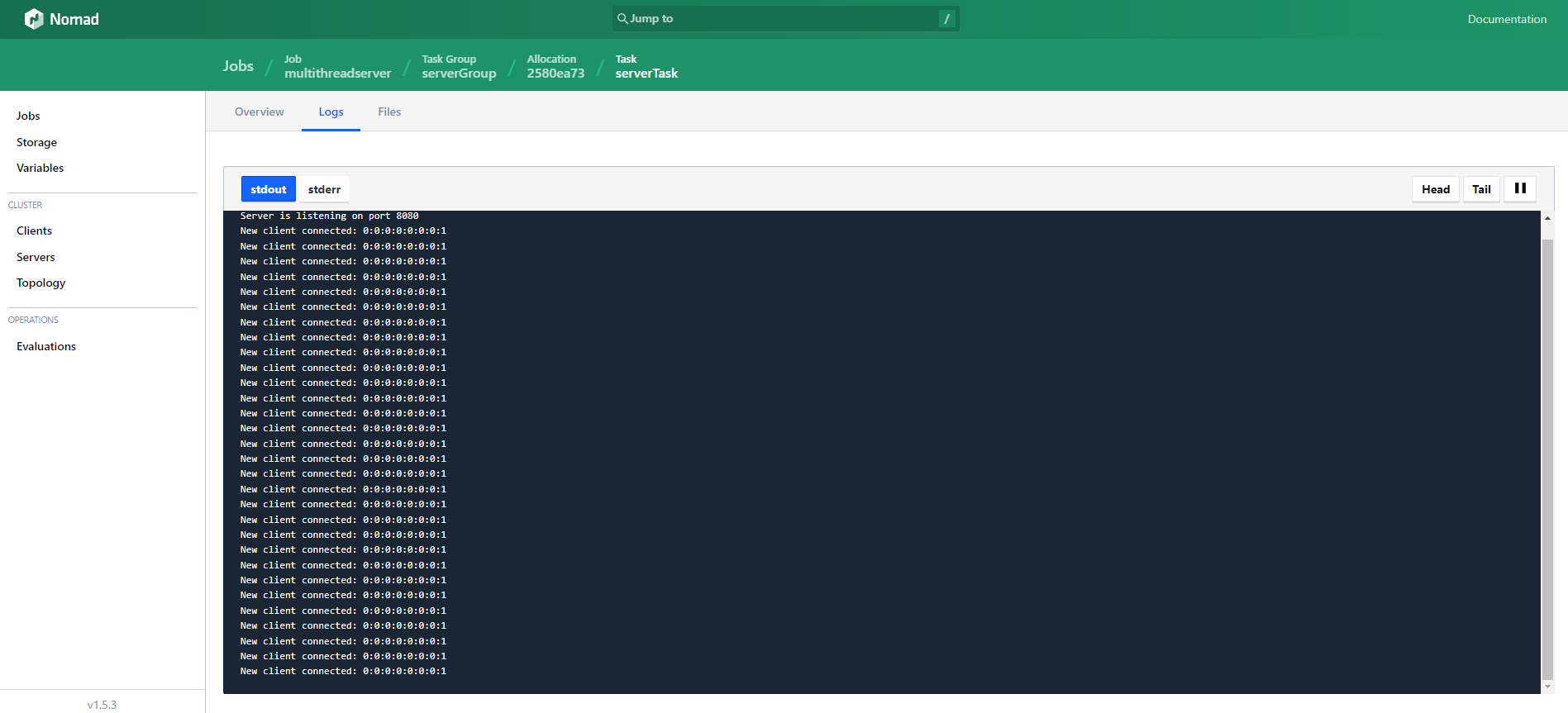
3. **Multithreading**: O código foi atualizado para usar um `ExecutorService` para gerenciar threads, em vez de criar manualmente novos threads.

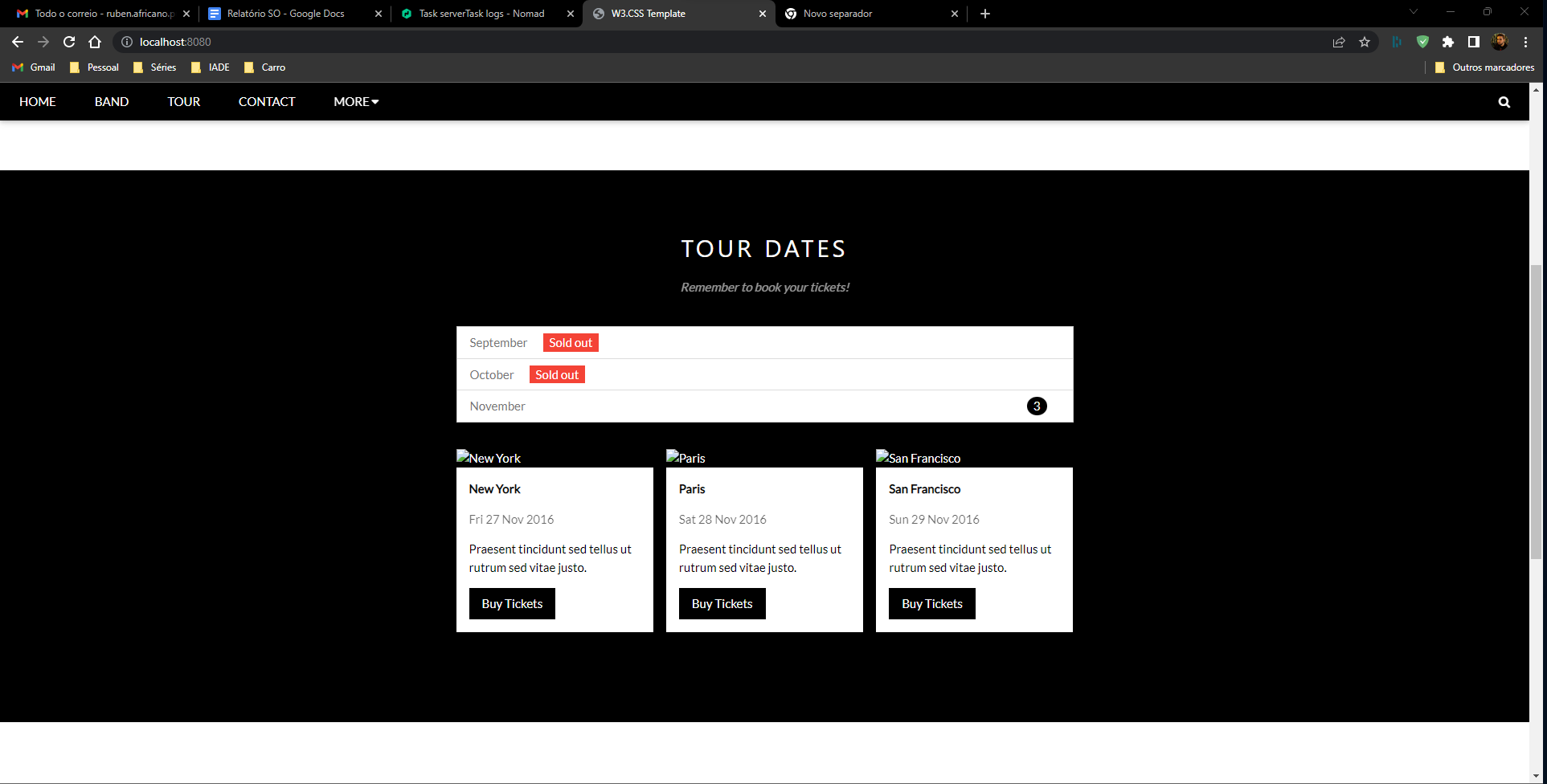
5. **Segurança**: O código foi atualizado para usar soquetes SSL, proporcionando uma conexão segura entre o servidor e os clientes.

6. **Documentação**: Comentários foram adicionados ao código para explicar o propósito e a funcionalidade de cada seção.

**Resultado:**







**Conclusão**

Este projeto serve como uma base sólida para um servidor multithreaded em Java. Ele demonstra práticas de programação seguras e eficientes, como o uso de threads, o tratamento adequado de exceções, a limpeza de recursos e a segurança SSL.

**A metodologia utilizada foi a pesquisa de papers relacionados com o tema, enriquecido com vários artigos:**

### What is multithreading?

# Paul Kirvan (Independent IT consultant/auditor)

Link: <https://www.techtarget.com/whatis/definition/multithreading>

# Web Workers: Multithreaded Programs in JavaScript

# Ido Green (Book) Link:<https://books.google.pt/books?hl=pt-PT&lr=&id=lEdt-AKB3iQC&oi=fnd&pg=PR5&dq=multithreading+simple+web+server&ots=fVM3xib66u&sig=ABMmo2lb3Akppaue6V-hgQ-FBKg&redir_esc=y#v=onepage&q&f=false>

* Nomad

Link: <https://developer.hashicorp.com/nomad/docs>

# 