Gerenciamento de Sessões e Persistência de Dados em Terminal Web para o Ensino de Computação

Gabriel Muller Fischer[1], **Pedro Ivo Kuhn**[1],
Rafael Burlamaqui Amaral[1], **Gerson Geraldo H. Cavalheiro**[1]

[1]Universidade Federal de Pelotas (UFPel)
R. Gomes Carneiro, 01 – Balsa, Pelotas – RS, 96010-610

{gmfischer, pikuhn, rafael.amaral, gerson.cavalheiro}@inf.ufpel.edu.br

Resumo. *Este trabalho propõe uma arquitetura distribuída de terminais de computação de alto desempenho integrados ao Moodle via LTI 1.3. A solução utiliza a orquestração do Kubernetes para o provisionamento dinâmico de ambientes conteinerizados, garantindo distribuição de carga e isolamento de recursos. Implementou-se um sistema de gerenciamento de sessões com interrupção por inatividade para otimização do cluster e uma camada de persistência de dados via MinIO. A proposta busca demonstrar que a abstração da infraestrutura assegura escalabilidade e eficiência no uso do hardware institucional, reduzindo custos operacionais e a complexidade de configuração em comparação a laboratórios físicos tradicionais.*

1. Introdução

A manutenção de laboratórios de informática em instituições de ensino superior enfrenta desafios multidimensionais que abrangem desde o elevado custo de aquisição de hardware até a complexidade logística de gerenciamento de software. Além disso, em disciplinas práticas de computação, o tempo de aula é frequentemente desperdiçado com a configuração de ambientes locais e resolução de conflitos de dependências nos dispositivos pessoais dos alunos.

Neste cenário, os Ambientes Virtuais de Aprendizagem (AVAs), como o Moodle [Moodle HQ 2025], consolidaram-se como o centro da gestão acadêmica. Contudo, suas funcionalidades nativas não suprem a necessidade de experimentação prática em ambientes de alto desempenho. A integração de ferramentas externas via protocolo *Learning Tools Interoperability* [Moodle HQ 2026] (LTI) 1.3 surge como uma solução para estender as capacidades do Moodle, permitindo a entrega de ferramentas especializadas através de uma arquitetura desacoplada e segura.

Este trabalho apresenta a evolução de um Terminal Web[Kuhn 2025] integrado ao Moodle, baseado em uma infraestrutura elástica orquestrada por Kubernetes [Cloud Native Computing Foundation 2025], que visa democratizar o acesso ao hardware institucional. A solução permite que o docente defina o ambiente de software através de imagens Docker, garantindo o isolamento de recursos e a distribuição de carga entre os nós do cluster. As contribuições recentes focam na sustentabilidade desta infraestrutura por meio de um sistema de gerenciamento de sessões com *timeout* e na garantia da continuidade do aprendizado através de uma camada de persistência de dados do usuário utilizando o sistema de armazenamento do MinIO [MinIO, Inc. 2026].

2. Trabalhos Relacionados

A integração de ferramentas de programação em Ambientes Virtuais de Aprendizagem (AVAs) é um tema consolidado, com soluções como o Virtual Programming Lab [del Pino 2026] (VPL) e o CodeRunner [Lobb 2026]. O VPL oferece edição e execução em servidores dedicados, enquanto o CodeRunner foca na avaliação automática via testes unitários. Entretanto, essas soluções apresentam limitações de escalabilidade e flexibilidade de ambiente. Por serem plugins nativos, a execução ocorre geralmente em *sandboxes* estáticas, restringindo o professor às linguagens e bibliotecas pré-instaladas no servidor do plugin.

Diferente dessas abordagens, a solução proposta utiliza o padrão LTI 1.3 para promover um desacoplamento total entre o AVA e o ambiente de execução. Enquanto o VPL gerencia recursos de forma isolada e efêmera, este trabalho utiliza a orquestração do Kubernetes para o provisionamento dinâmico de instâncias. Isso permite que a virtualização baseada em containers ofereça uma densidade de usuários significativamente superior a laboratórios baseados em máquinas virtuais tradicionais, devido ao compartilhamento do kernel e menor consumo de overhead por instância.

O grande diferencial reside na combinação de liberdade de configuração via Docker [Docker, Inc. 2025] com a persistência de estado via MinIO. Ao contrário das soluções citadas, onde os arquivos costumam ficar vinculados à atividade do Moodle, o uso do MinIO permite que o estudante gerencie seu próprio volume de dados, garantindo a continuidade do trabalho entre diferentes sessões e dispositivos, independentemente da efemeridade dos pods do cluster.

3. Arquitetura e Implementação

A infraestrutura utiliza o Kubernetes para a orquestração do ciclo de vida das instâncias. Cada acesso via Moodle dispara o provisionamento de um Pod efêmero, conforme ilustrado na Figura 1, garantindo o isolamento lógico e de recursos (CPU/RAM) via *namespaces* e *cgroups*. Esta abordagem move a carga computacional para o cluster, permitindo que o aluno realize tarefas complexas de qualquer dispositivo com navegador, seja utilizando a infraestrutura física da faculdade ou seu computador pessoal fora dela.

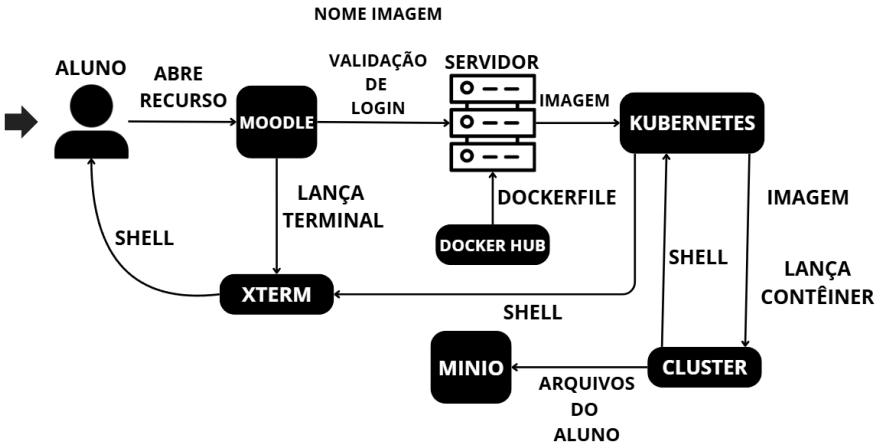

Figura 1. Diagrama da comunicação entre usuário e o *Terminal Web*

A comunicação Moodle-Terminal ocorre via protocolo LTI 1.3, estabelecendo um canal de confiança baseado em chaves assimétricas. Esta camada automatiza a autenticação e transmite os parâmetros de configuração, como a imagem Docker específica definida pelo docente para a disciplina, assegurando a padronização do ambiente de execução.

Para a persistência de estado, integrou-se o MinIO como sistema de armazenamento de objetos, cujo fluxo de permanência de dados é detalhado na Figura 2. Ao iniciar a sessão, o sistema realiza o mapeamento do volume do usuário, preservando o progresso entre sessões. O sistema impõe uma quota de 200 MB e oferece uma interface externa para gerenciamento e exportação de arquivos, garantindo que os dados do aluno permaneçam disponíveis mesmo após o encerramento do contêiner.

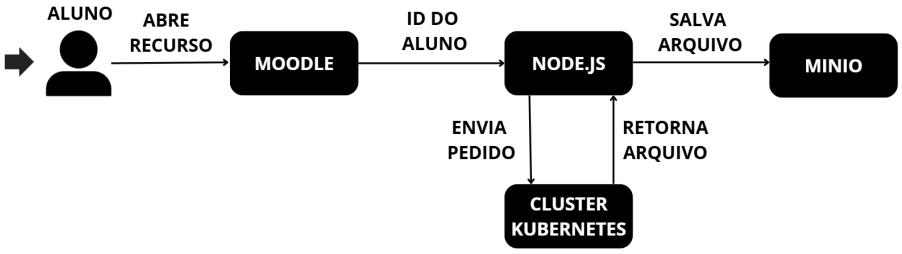

Figura 2. Diagrama do sistema de permanência com *MinIO*

O controle de recursos é otimizado por uma política de timeout de 120 minutos, alinhada à duração média das atividades práticas. O mecanismo encerra sessões ociosas para liberar recursos no escalonador do Kubernetes, mas permite prorrogações sucessivas de 30 minutos mediante interação ativa do usuário.

Avaliação e Desempenho: A arquitetura foi validada com 26 alunos simultâneos em um cluster de 10 nós (Intel i5, 16GB RAM), totalizando 40 núcleos lógicos. Para otimizar a densidade, as instâncias utilizaram reservas (requests) de 0,1 vCPU e limites (limits) de 0,5 vCPU. Esta configuração elástica garantiu baixo consumo ocioso e alto desempenho sob demanda. O tempo médio de provisionamento foi de 5 segundos para imagens em cache, sendo estendido apenas no download inicial de imagens inéditas. Embora o MinIO e a gestão de sessões estivessem em fase de implementação durante os testes, os resultados motivaram a consolidação destas funções para a sustentabilidade do ambiente.

4. Discussão e Implicações da Solução

A centralização no Kubernetes permite que computadores modestos atuem como terminais de alto desempenho, movendo a carga computacional para a infraestrutura institucional. Essa abstração possibilita o consumo elástico de recursos [Bernstein 2014], distribuindo a capacidade do cluster conforme a demanda das turmas. Adicionalmente, a política de timeout evita o desperdício de hardware, garantindo rotatividade eficiente e preservando o estado da sessão contra instabilidades de rede.

Do ponto de vista pedagógico, a abstração do ambiente via Docker possibilitou que o início das atividades práticas ocorra de forma imediata, uma vez que o professor pré-configura as dependências e ferramentas na imagem da disciplina, eliminando a necessidade de intervenções manuais em cada terminal físico do laboratório.

Quanto à persistência, o MinIO resolve a efemeridade dos contêineres, permitindo que o estudante mantenha seu fluxo de trabalho de forma transparente entre sessões. Diferente de soluções que vinculam arquivos à atividade, o sistema proposto garante a independência dos dados do usuário, facilitando o acesso remoto e a integridade da pasta root. Atualmente, o projeto encontra-se em fase de testes para validar se a cota de 200 MB por usuário é suficiente para a demanda acadêmica das disciplinas, assim como para mensurar o impacto da carga de trabalho no armazenamento centralizado.

5. CONCLUSÃO

Este trabalho busca demonstrar a viabilidade de integrar um Terminal Web ao Moodle, unindo a escalabilidade do Kubernetes à persistência do MinIO. A solução resolve gargalos de configuração e desperdício de tempo em laboratórios físicos, oferecendo uma infraestrutura flexível para docentes e discentes.

Do ponto de vista institucional, a solução promove a centralização e o compartilhamento de recursos de alto desempenho no cluster, permitindo que os laboratórios físicos utilizem estações de trabalho com hardware mais modesto. Essa abordagem otimiza o investimento governamental ao evitar a aquisição de múltiplos laboratórios com hardware potente e exclusivo, concentrando o poder de processamento em um equipamento compartilhado de alta densidade. Tal modelo reduz drasticamente a complexidade de manutenção local, estende o ciclo de vida do parque tecnológico existente e assegura que a capacidade computacional seja distribuída de forma equânime entre diferentes turmas e disciplinas.

Testes iniciais realizados com uma turma real validaram a estabilidade do acesso via Moodle e o desempenho do terminal em atividades práticas. Como passos futuros, o projeto focará na validação do sistema de timeout e da persistência via MinIO em cenários de uso real, além da análise detalhada do impacto dessas funcionalidades no consumo de recursos do cluster sob alta carga.

Referências

Bernstein, D. (2014). Containers and cloud: From LxC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Cloud Native Computing Foundation (2025). Kubernetes: Production-grade container orchestration. Acessado em: 28 ago. 2025.

del Pino, J. C. R. (2026). Virtual programming lab (vpl) for moodle. Acessado em: 30 mar. 2026.

Docker, Inc. (2025). Docker: Accelerated, containerized application development. Acessado em: 28 ago. 2025.

Kuhn, P. I. (2025). Terminal web. Acessado em: 21 ago. 2025.

Lobb, R. (2026). Coderunner: A tool for automatic assessment of programming code. Acessado em: 30 mar. 2026.

MinIO, Inc. (2026). High performance object storage. Acessado em: 24 fev. 2026.

Moodle HQ (2025). Open-source learning platform. Acessado em: 28 ago. 2025.

Moodle HQ (2026). Lti external tools: Publish as lti tool. Acessado em: 30 mar. 2026.