Universidade Estadual do Sudoeste da Bahia - UESB
Bacharelado em Ciência da Computação

Cristiano Costa Batista

Implementação de um editor de texto colaborativo no LARA: uma abordagem com Etherpad em ambiente Kubernetes

Vitória da Conquista - BA
2025

Cristiano Costa Batista

Implementação de um editor de texto colaborativo no LARA: uma abordagem com Etherpad em ambiente Kubernetes

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Sudoeste da Bahia, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Orientadora: Prof.ª, Dr. ª Maísa Soares do Santos Lopes.

Vitória da Conquista - BA
2025

Agradecimentos

A minha mãe, minha avó materna e meu irmão, por tornarem essa realidade possível.

À minha orientadora, a Professora Dra. Maísa Soares dos Santos Lopes, pela paciência, pelas instruções, pelas conversas leves e enriquecedoras, pela confiança e por me guiar durante todo o processo.

Aos meus amigos, por todos os momentos que lembrarei com carinho e gratidão, e por terem tornado essa jornada tão leve e inesquecível.

Cada um de vocês fez parte de uma fase incrível em minha vida.

Resumo

A programação tem se consolidado como competência central nas áreas de tecnologia e campos correlatos, tanto pela sua capacidade de estruturar o pensamento lógico quanto por viabilizar soluções para problemas complexos. Contudo, o ingresso de estudantes iniciantes nesse universo costuma ser marcado por obstáculos: compreensão de abstrações, domínio de sintaxe e ferramentas e, sobretudo, limitações na colaboração e no acompanhamento da evolução do trabalho coletivo.

Sendo assim, este trabalho apresenta a implementação de um editor de texto colaborativo no ambiente virtual de aprendizagem LARA, com foco na preparação técnica para a futura integração do Etherpad — uma ferramenta de edição colaborativa em tempo real — a essa plataforma. A proposta organiza as bases de um módulo de coedição integrado ao LARA, assegurando condições de estabilidade, persistência e governança para o uso pedagógico.

Do ponto de vista educacional, o módulo projeta ganhos para iniciantes em programação: a escrita conjunta com identificação de autoria, o chat acoplado e o histórico reproduzível do documento favorecem a co-construção de soluções, a regulação do grupo e a avaliação formativa do processo. Ao reduzir barreiras operacionais e tornar visíveis as contribuições, o ambiente estimula participação, raciocínio em pares e metacognição, criando um contexto seguro para "aprender programando com os outros" e aproximando os alunos de práticas profissionais de desenvolvimento colaborativo.

Palavras-chave: aprendizagem colaborativa suportada por computador; editor de texto colaborativo; ambiente virtual de aprendizagem.

Abstract

Programming has become a core competency across technology and related fields, both for its ability to structure logical thinking and for enabling solutions to complex problems. Nevertheless, beginners entering this universe often face obstacles: grasping abstractions, mastering syntax and tools, and, above all, limitations in collaboration and in tracking the evolution of collective work.

Accordingly, this work presents the implementation of a collaborative text editor within the LARA virtual learning environment, focusing on the technical preparation for the future integration of Etherpad—a real-time collaborative editing tool—into that platform. The proposal lays the foundations of a co-editing module integrated into LARA, ensuring stability, persistence, and governance for pedagogical use.

From an educational standpoint, the module is expected to benefit novice programmers: joint writing with authorship identification, an embedded chat, and a reproducible document history support the co-construction of solutions, group self-regulation, and formative assessment of the process. By lowering operational barriers and making contributions visible, the environment fosters participation, pair reasoning, and metacognition, creating a safe context for "learning to program with others" and bringing students closer to professional practices of collaborative software development.

Keywords: computer-supported collaborative learning; collaborative text editor; virtual learning environment.

Sumário

1 Introdução

Nos últimos anos, a educação tem passado por transformações significativas, impulsionadas pelo avanço da tecnologia e pela crescente demanda por métodos de ensino mais flexíveis e eficientes. Nesse contexto, os Ambientes Virtuais de Aprendizagem (AVA) emergiram como ferramentas essenciais para facilitar o processo de ensino e aprendizagem. Um AVA consiste em uma plataforma online que permite a interação entre estudantes e professores, disponibilizando recursos educacionais, ferramentas de comunicação e atividades colaborativas. Esse ambiente virtual possibilita que a aprendizagem ocorra de maneira contínua e dinâmica, independentemente de barreiras geográficas.

O Laboratório Acadêmico em Rede de Aprendizagem (LARA) é um AVA desenvolvido com o objetivo de ser uma plataforma educacional que integra recursos tecnológicos e métodos de ensino para aprimorar o processo de aprendizagem nas disciplinas do curso de Ciência da Computação. O LARA abrange funcionalidades de laboratórios remotos, laboratórios virtuais, elementos de engenharia de software, integração com projetos de robótica e suporte a estratégias de colaboração no ensino de programação (LOPES, 2017). Em outras palavras, trata-se de um ambiente rico e multidisciplinar, voltado a oferecer experiências práticas e colaborativas de aprendizagem. Entretanto, para que essa colaboração seja efetivamente aproveitada, faz-se necessária a integração de ferramentas adequadas que suportem atividades conjuntas em tempo real.

Uma ferramenta essencial para viabilizar a aprendizagem colaborativa no LARA é o Etherpad, um editor de texto online open-source que permite a edição simultânea de documentos por múltiplos usuários. O Etherpad oferece uma interface web intuitiva e funcionalidades que facilitam a colaboração síncrona, como destaque de texto por autor (cada colaborador é identificado por uma cor e nome), chat embutido para comunicação e salvamento automático de alterações. Essas características tornam a ferramenta atrativa para atividades em grupo, possibilitando que os alunos escrevam código ou texto em conjunto e vejam as contribuições uns dos outros em tempo real. A Figura 1 ilustra a interface do Etherpad Lite, com múltiplos usuários editando simultaneamente um documento e cada inserção de texto marcada com a cor de seu respectivo autor.

Apesar das vantagens do Etherpad em termos de edição colaborativa síncrona, ele apresenta limitações importantes no contexto de programação. Em especial, a ausência de um sistema de controle de versão robusto integrado ao Etherpad gera desafios durante atividades colaborativas de desenvolvimento de software. Diferentemente de ambientes de desenvolvimento tradicionais que utilizam controle de versão (como o Git) para registrar o histórico de mudanças no código, o Etherpad provê apenas um histórico linear das edições (uma "linha do tempo") e não oferece mecanismos avançados de ramificação, mesclagem ou recuperação sofisticada de versões do conteúdo. Como resultado, em um cenário com vários alunos editando código simultaneamente, torna-se difícil acompanhar quem fez cada

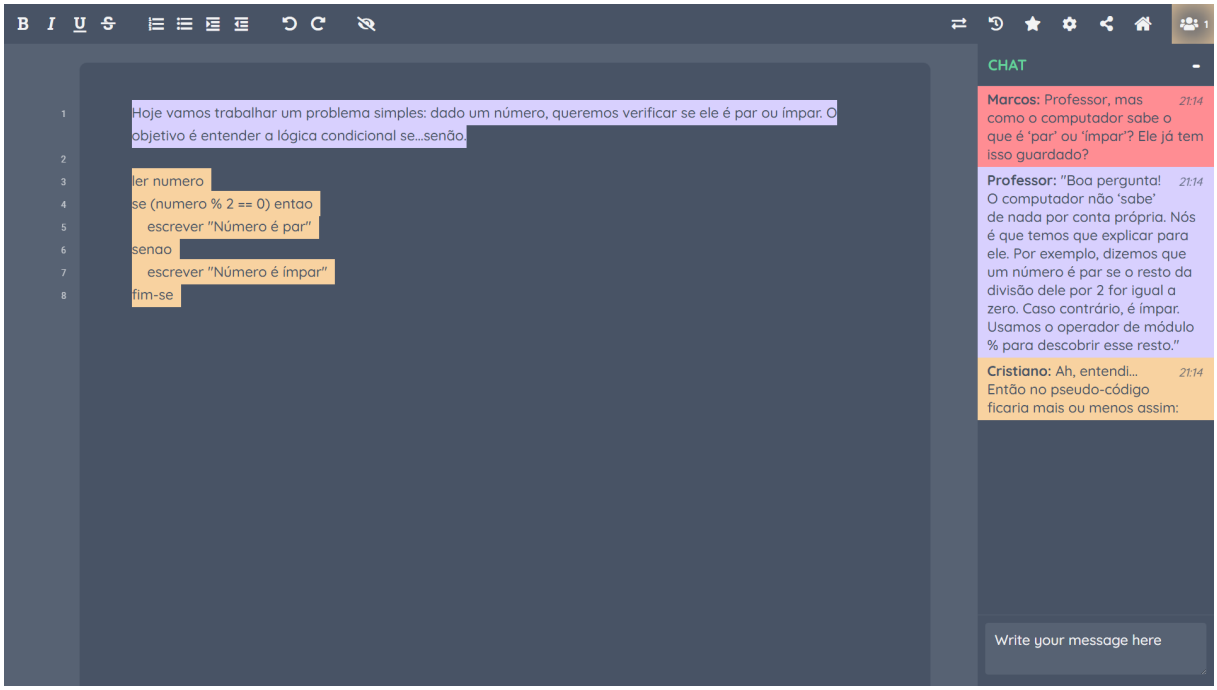

Figura 1 – Interface de um pad no Etherpad Lite.

modificação, reverter mudanças específicas ou resolver conflitos de edições concorrentes. Problemas como perda de trabalho, dificuldade de rastrear o histórico de alterações e obstáculos na correção colaborativa podem emergir devido a essa carência de versionamento estruturado.

A importância de um sistema de controle de versões em contextos colaborativos é amplamente reconhecida. Tais sistemas permitem gerenciar de forma confiável as contribuições de múltiplos desenvolvedores, mantendo a integridade do código ao longo do tempo (LOELIGER; MCCULLOUGH, 2012). No desenvolvimento de software educacional, Hochmüller e Maurer (2018) defende que o uso de controle de versão promove maior organização do trabalho em grupo e auxilia os estudantes a compreenderem a evolução de seus projetos. Sem esse recurso, os alunos podem enfrentar conflitos de código frequentes e dificuldade para consolidar diferentes partes de uma solução programada coletivamente. Esse cenário compromete a eficácia da aprendizagem colaborativa, o que justifica a necessidade de aprimorar o ambiente LARA com mecanismos adicionais de versionamento.

Diante desse contexto, formulou-se o problema de pesquisa que motivou este trabalho: como integrar o Etherpad ao LARA de forma a oferecer um ambiente de programação colaborativa em tempo real para alunos iniciantes, suprindo a falta de controle de versão e garantindo a estabilidade e persistência da aplicação? O presente trabalho adota uma abordagem com foco na implementação do Etherpad no LARA utilizando tecnologias de containerização e orquestração, de modo a garantir uma implantação estável, escalável e com persistência de dados, ao mesmo tempo em que se explora a integração

com sistemas de versionamento externos (como o Git) para suprir as lacunas identificadas.

Em síntese, este projeto visa dotar o LARA de uma ferramenta colaborativa madura para edição de código/texto, alinhada aos princípios de Aprendizagem Colaborativa Suportada por Computador (CSCL) e às necessidades do primeiro contato com programação por estudantes iniciantes. Acredita-se que a integração do Etherpad ao LARA, aliada a uma arquitetura tecnológica moderna (Docker, Kubernetes) e a mecanismos de versionamento, proporcionará um ambiente enriquecido em que os alunos possam programar coletivamente de forma segura, síncrona e eficaz. A seguir, são apresentados os objetivos que norteiam este trabalho, bem como a metodologia empregada para atingi-los.

1.1 Objetivos

1.1.1 Objetivo Geral

Implementar a ferramenta Etherpad no ambiente virtual de aprendizagem LARA, de forma a servir como plataforma de programação colaborativa para estudantes iniciantes, garantindo sua integração plena ao ambiente já existente e atendendo aos requisitos de colaboração em tempo real e persistência de dados.

1.1.2 Objetivos Específicos

a) Levantar e analisar requisitos funcionais e não funcionais, pedagógicos e técnicos do módulo de edição colaborativa a ser preparado para integração ao LARA;

b) Definir arquitetura do sistema no ambiente Kubernetes;

c) Criar um *Dockerfile* para uma versão estável e reprodutível do Etherpad;

d) Integrar o Etherpad ao LARA, garantindo persistência de dados e edição colaborativa em tempo real.

2 Fundamentação Teórica

2.1 Aprendizagem Colaborativa Suportada por Computador (CSCL)

A Aprendizagem Colaborativa Suportada por Computador, conhecida pela sigla CSCL (do inglês Computer-Supported Collaborative Learning), é um campo de estudo dedicado a investigar como a tecnologia pode apoiar e mediar processos de aprendizagem colaborativa entre estudantes. De forma geral, a CSCL promove a interação social e a cooperação entre alunos por meio de plataformas digitais, com o objetivo de melhorar a qualidade da educação (KOSCHMANN, 1996). Stahl, Koschmann e Suthers (2006) argumentam que a CSCL oferece ferramentas que permitem a troca de ideias e conhecimentos em grupos, criando um ambiente de aprendizagem mais dinâmico e interativo . Esse ambiente colaborativo incentiva os alunos a construírem conhecimento coletivamente, em contraste com modelos tradicionais centrados no aprendizado individual.

Segundo Dillenbourg (1999), a CSCL envolve o uso de computadores para apoiar grupos de aprendizes na realização de tarefas compartilhadas. Por meio de espaços virtuais, os alunos podem trabalhar conjuntamente em um mesmo problema, compartilhar informações e cocriar soluções, mesmo que estejam geograficamente distantes. A tecnologia atua como facilitadora da comunicação e coordenação do grupo, oferecendo recursos como fóruns de discussão, bate-papo (chat), videoconferência e ferramentas de coedição de documentos. Esses recursos são essenciais para viabilizar interações síncronas e assíncronas ricas em um contexto educacional colaborativo. Vale ressaltar que disponibilizar a tecnologia por si só não garante colaboração efetiva (PIURA et al., 2014); é necessário um desenho pedagógico adequado para engajar os participantes e orientar as interações de forma produtiva. Em suma, a CSCL integra princípios de educação colaborativa com recuros tecnológicos, buscando criar experiências de aprendizagem em que os estudantes aprendam com e através dos outros, mediadas por computadores.

2.2 Ambientes Virtuais de Aprendizagem (AVA)

Ambientes Virtuais de Aprendizagem (AVA) são plataformas baseadas em software que dão suporte ao processo de ensino e aprendizagem no meio digital. Também conhecidos como Learning Management Systems (LMS), esses ambientes permitem a organização de cursos e disciplinas on-line, oferecendo um conjunto integrado de funcionalidades para professores e alunos (GOMES; PIMENTEL, 2021). Os AVAs possibilitam o acesso a conteúdos educacionais de forma flexível e acessível, muitas vezes a qualquer hora e lugar, bastando uma conexão à Internet. Além disso,Williams (2022) destacam que os AVAs favorecem tanto interações assíncronas (por exemplo, fóruns de discussão, envio de tarefas,

mensagens) quanto síncronas (como webconferências e chats), ampliando as oportunidades de comunicação pedagógica.

2.2.1 Funções e características dos AVAs

De modo geral, um AVA desempenha diversos papéis no suporte ao ensino. Dentre as principais características e funcionalidades de um AVA típico, podemos citar:

- **Gerenciamento de Conteúdo:** permite ao professor disponibilizar materiais didáticos (textos, vídeos, slides) e organizar módulos ou unidades de aprendizagem de forma estruturada. Os alunos podem acessar e navegar por esses conteúdos conforme seu progresso no curso (GOMES; PIMENTEL, 2021).

- **Comunicação e Colaboração:** provê ferramentas de interação como fóruns de discussão, sistemas de mensagens internas, chats em tempo real e, em alguns casos, wikis ou editores colaborativos de texto. Essas funcionalidades estimulam a troca de ideias entre os participantes e a aprendizagem coletiva, alinhadas aos princípios da CSCL (GOMES; PIMENTEL, 2021).

- **Gestão de Atividades e Avaliações:** inclui mecanismos para criação e entrega de tarefas, quizzes ou exercícios on-line, bem como sistemas de avaliação e feedback. Os professores podem acompanhar o desempenho dos alunos e fornecer retorno individualizado dentro da própria plataforma (GOMES; PIMENTEL, 2021).

- **Acompanhamento e Registro:** registra dados sobre a participação dos alunos (acesso a materiais, postagens, notas obtidas), permitindo o monitoramento do progresso e a geração de relatórios. Esse acompanhamento ajuda a identificar dificuldades e a adaptar intervenções pedagógicas (PEREIRA, 2023).

- **Acessibilidade e Personalização:** os AVAs buscam atender a diversos perfis de usuários, oferecendo suporte a múltiplos idiomas, adaptações de interface e compatibilidade com dispositivos móveis. Também permitem certa personalização do ambiente pelo professor (por exemplo, escolha de ferramentas ativas no curso) para adequá-lo à metodologia desejada (GOMES; PIMENTEL, 2021).

  Essas funções tornam os AVAs componentes centrais na educação a distância e no ensino híbrido, pois organizam de forma sistemática o ecossistema de um curso on-line (PIURA et al., 2014). Ferramentas amplamente utilizadas, como o Moodle, Google Classroom, Canvas, entre outras, exemplificam como um AVA integra recursos educacionais variados em um único ambiente coeso. Figura 2 ilustra, de forma esquemática, os principais componentes e fluxos de interação em um AVA típico, incluindo as interfaces para professores, alunos e recursos externos.

Figura 2 – Representação esquemática dos principais componentes e funcionalidades de um Ambiente Virtual de Aprendizagem (AVA) — igualmente conhecido como Learning Management System (LMS) [1].

2.2.2 Exemplo: O ambiente LARA

Como estudo de caso de um AVA com características especiais de colaboração, destaca-se o projeto LARA – Laboratório Remoto em Ambiente Virtual de Aprendizagem. Segundo Lopes (2017), o LARA foi concebido como parte de uma iniciativa de inovação pedagógica que integra aprendizagem colaborativa, programação e laboratórios remotos de robótica em um mesmo ambiente educacional. Em essência, o LARA estende as funcionalidades tradicionais de um AVA ao incorporar ferramentas que permitem atividades práticas de programação e experimentação remota, tudo integrado à plataforma de ensino.

A arquitetura do LARA, denominada LaraPC (LARA para Programação Colaborativa), combina módulos de estudo baseados em um AVA (no caso, foi utilizado o Moodle) com módulos customizados para suporte a laboratório remoto e programação colaborativa (LOPES, 2017). O laboratório remoto viabiliza que estudantes controlem e observem, via Internet, um robô físico localizado em um laboratório, permitindo experimentações reais à distância. Paralelamente, o LARA foi concebido para apoiar atividades de codificação on-line no qual grupos de alunos podem desenvolver algoritmos em conjunto, em tempo real, integrando conceitos de CSCL à aprendizagem de programação. Para isso, o ambiente prevê a formação de grupos (*parties*) e o uso de ferramentas de coedição e análise de problemas. Ao ingressar em uma sessão, os alunos podem trabalhar colaborativamente em um mesmo artefato e manter um espaço compartilhado, favorecendo a discussão e a

resolução conjunta do desafio proposto.

A integração entre os componentes do LARA é feita de forma transparente ao usuário. A Figura 3 apresenta uma visão geral dessa arquitetura: o módulo Moodle cuida do gerenciamento de cursos (inscrições, notas, conteúdo teórico), enquanto módulos adicionais tratam da comunicação com o hardware do laboratório remoto e da coordenação da colaboração entre alunos.

Notavelmente, a proposta arquitetural descrita por Lopes (2017) prevê a incorporação de um editor colaborativo em tempo real ao LARA como serviço para edição compartilhada de texto e código dentro do ambiente. Neste trabalho, adota-se o Etherpad (ver seção 4) como candidato a cumprir esse papel e realiza-se sua preparação técnica para futura integração com o AVA. Assim, quando o aluno acessar a área de desenvolvimento do LARA, pretende-se que um pad colaborativo seja criado e disponibilizado ao grupo para editar o código em conjunto, enquanto o chat pode ser usado para análises e discussões do grupo.

Esse exemplo demonstra como um AVA pode ser aprimorado com ferramentas colaborativas especializadas, integrando aprendizado teórico, colaboração on-line e experimentação prática em um único ecossistema educacional.

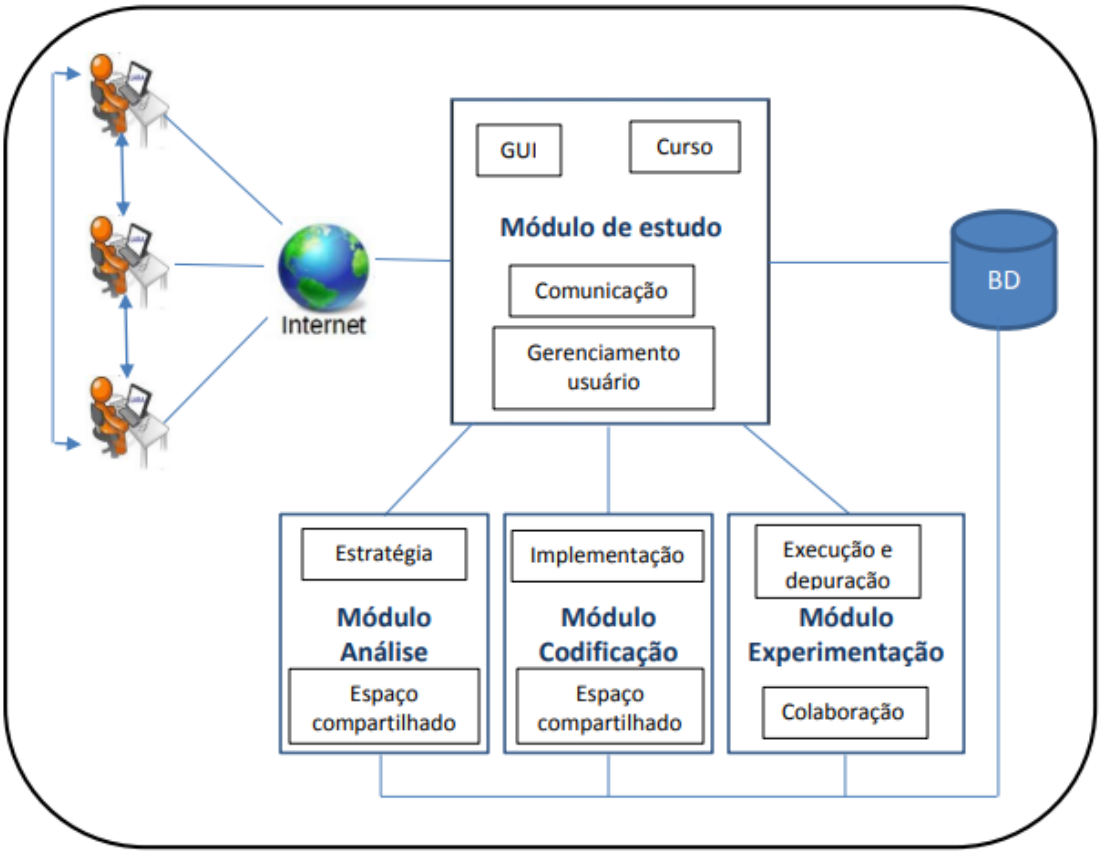

Figura 3 – Componentes da arquitetura proposta para o LARA em Lopes (2017).

2.3 Edição Colaborativa

Com a proliferação de aplicações de produtividade on-line, a edição colaborativa de documentos tornou-se uma funcionalidade cada vez mais comum e importante. De acordo com Dang e Ignat (2016), Sun (1998), a edição colaborativa refere-se ao processo em que múltiplos usuários podem criar e revisar simultaneamente um mesmo documento digital, compartilhando em tempo real as atualizações uns dos outros. Exemplos populares incluem ferramentas como Google Docs, Microsoft Office Online e editores de código multiusuário. Em contextos educacionais e de trabalho em grupo, a possibilidade de coeditar documentos permite que ideias sejam construídas coletivamente e de forma ágil, evitando envios de versões múltiplas por e-mail e consolidando as contribuições de todos em um só lugar (MCKELVEY et al., 2023).

Entretanto, permitir que várias pessoas editem simultaneamente um mesmo documento traz desafios técnicos significativos. Como apontam Wu e Pu (2009), o principal deles é como manter a consistência do documento em todas as telas dos usuários, mesmo quando edições ocorrem de forma concorrente (por exemplo, dois usuários inserindo texto em posições próximas ao mesmo tempo). Para resolver esse problema, foram desenvolvidos protocolos e algoritmos específicos que garantem a convergência do documento colaborativo – isto é, que ao final todos vejam o mesmo conteúdo, independentemente da ordem exata em que as operações de edição ocorreram em cada lugar (ELLIS; GIBBS, 1989).

Duas abordagens amplamente utilizadas para viabilizar a edição colaborativa consistente são a Transformação Operacional (OT, Operational Transformation) e os Tipos de Dados Replicados sem Conflito (CRDT, Conflict-Free Replicated Data Types) (SHAPIRO et al., 2011; SUN, 1998). Em termos gerais, ambas as abordagens asseguram que, apesar de edições concorrentes, todas as réplicas do documento cheguem a um estado final idêntico (consistência eventual). No entanto, elas o fazem de maneiras distintas:

- **Transformação Operacional (OT):** Introduzida no final da década de 1980, a OT se baseia na ideia de transformar as operações de edição recebidas de outros usuários antes de aplicá-las localmente, caso haja conflitos de posição no documento (SUN, 1998). Em OT, cada ação do usuário (inserir ou apagar texto, por exemplo) é enviada a um servidor central juntamente com o contexto em que ocorreu (por exemplo, a posição onde o texto foi inserido, baseada na versão local do documento). O servidor então retransmite essa operação para os demais clientes. Se um cliente já tiver realizado operações locais posteriores, ele transforma a operação recebida para ajustar sua posição ou conteúdo, de forma a não sobrescrever indevidamente as alterações próprias. Após as transformações necessárias, a operação é aplicada. Esse processo garante que todas as operações concorrentes sejam incorporadas de maneira consistente em cada cópia do documento. A maioria dos editores colaborativos tradicionais, incluindo o próprio Etherpad, adotam algoritmos baseados em OT para

gerenciar a concorrência (Etherpad Foundation, 2018). A vantagem da OT é que ela foi bastante explorada e otimizada ao longo dos anos, mostrando-se eficiente para um número moderado de usuários simultâneos e com baixa latência de rede. Entretanto, a OT normalmente depende de um servidor central coordenador, e a implementação correta de todos os casos extremos (edge cases) pode se tornar complexa conforme o número de tipos de operações aumenta.

- **Tipos de Dados Replicados sem Conflito (CRDT):** Os CRDTs são uma família mais recente de técnicas que permitem a replicação de dados sem conflitos em sistemas distribuídos, muitas vezes sem necessidade de um servidor central coordenando as operações (PREGUIÇA; BAQUERO; SHAPIRO, 2018). Em um CRDT para edição de texto, cada inserção ou remoção é tratada como uma operação com identificadores únicos e uma ordem determinada de forma lógica (por exemplo, utilizando estruturas de árvore ou sequências com identificadores causais). Todos os eventos de edição são disseminados para os demais participantes, e cada cliente aplica localmente todas as operações recebidas, confiando que a definição do CRDT garante a convergência automática (já que as operações são projetadas para serem comutativas e idempotentes) (SHAPIRO et al., 2011). Isso significa que, mesmo que diferentes usuários editem offline e sincronizem posteriormente, ou que as mensagens cheguem em ordens distintas a cada cliente, o resultado final será consistente sem a necessidade de resolução de conflitos complexa. Uma vantagem importante, enfatiza por Kleppmann et al. (2019), é a resiliência a falhas de rede: CRDTs funcionam bem em cenários *peer-to-peer* ou com conexões intermitentes, permitindo edição offline com sincronização posterior [2]. Por outro lado, CRDTs podem introduzir *overhead* de armazenamento ou desempenho, pois muitas vezes precisam manter metadados adicionais (por exemplo, identificadores únicos para cada caractere inserido) e algoritmos de *garbage collection* para não crescer indefinidamente (RINBERG et al., 2022). Embora ainda sejam menos comuns que OT em implementações industriais, CRDTs têm ganhado espaço em aplicações que demandam alta escalabilidade e funcionamento descentralizado.

As principais diferenças entre OT e CRDT podem ser resumidas em uma tabela comparativa. A Tabela 1 exemplifica essa comparação, contrastando aspectos como dependência de servidor, suporte a operações offline e complexidade de implementação de cada abordagem. Em linhas gerais, OT tende a ser mais simples de implementar em modelos cliente-servidor e apresenta baixo custo de comunicação, porém requer conectividade para todos os participantes e cuidados para tratar conflitos. Já CRDTs oferecem maior robustez em ambientes distribuídos heterogêneos, ao custo de estruturas mais elaboradas e, potencialmente, maior consumo de recursos.

[2] Jahns, Kevin. Are CRDTs Suitable for Shared Editing? Disponível em: <https://blog.kevinjahns.de/

Tabela 1 – Comparação simplificada entre Transformação Operacional (OT) e CRDT em editores colaborativos.

Aspecto	**Transformação Operacional (OT)**	**CRDT (Dados Replicados sem Conflito)**
Dependência de Servidor	Requer geralmente um servidor central para coordenar as operações (modelo cliente-servidor).	Pode operar de forma distribuída, *peer-to-peer*, sem necessidade de servidor central para ordem global.
Suporte Offline	Edições offline são limitadas; é necessário sincronização com o servidor para aplicar operações.	Permite edições offline; as operações são mescladas posteriormente, mantendo a consistência.
Complexidade	Algoritmos de OT precisam lidar com muitos casos de transformação; implementações maduras existem, mas exigem cuidado para assegurar convergência.	A lógica de convergência está embutida na estrutura de dados; pode demandar mais memória e processamento para manter identificadores e históricos.
Exemplos de Uso	Etherpad, Google Docs (variações de OT proprietárias), Apache Wave (Google Wave)	Automerge, Yjs (bibliotecas CRDT para textos), aplicativo PeerPad (editor P2P).

Do ponto de vista do desenvolvedor de aplicações colaborativas, a escolha entre OT e CRDT depende dos requisitos do sistema (número de usuários concorrentes, tolerância a falhas de conexão, infraestrutura disponível etc.). Em ambos os casos, o objetivo final é o mesmo: garantir a consistência do documento e a experiência em tempo real para todos os participantes. No contexto deste trabalho, é importante notar que o Etherpad utiliza uma variante de algoritmo OT em seu núcleo para permitir a edição colaborativa em tempo real (detalhado na próxima seção). Assim, compreender esses conceitos fundamenta a análise de como ferramentas como Etherpad lidam com edição concorrente e como podem ser integradas em plataformas educativas colaborativas.

2.4 Controle de versões em documentos

O conceito de controle de versões refere-se ao conjunto de técnicas e ferramentas que permitem registrar, gerenciar e recuperar diferentes estados de um documento ao longo do tempo (CHACON; STRAUB, 2014). Tradicionalmente aplicado ao desenvolvimento de software (controle de versões de código-fonte), esse conceito é igualmente relevante para documentos textuais em geral, sobretudo quando produzidos de forma colaborativa ou iterativa. A ideia central é que toda modificação realizada fique registrada, de modo que seja possível entender quem alterou o quê e quando, comparar versões e, eventualmente, reverter alterações indesejadas.

are-crdts-suitable-for-shared-editing>. Acesso em: 06 out. 2025.

Sistemas de controle de versão (também chamados Version Control Systems, VCS) surgiram para solucionar problemas de coordenação e histórico em projetos com múltiplos arquivos e autores. Um dos sistemas mais populares atualmente é o Git, criado em 2005 por Linus Torvalds, inicialmente para gerenciar o desenvolvimento do kernel do Linux (CHACON; STRAUB, 2014). O Git adotou uma arquitetura distribuída inovadora: em vez de um repositório central único, cada desenvolvedor possui localmente uma cópia completa do repositório com todo o histórico do projeto. As contribuições são integradas por meio de operações de commit (que grava um estado do código com mensagem descritiva) e merge (que combina diferentes linhas de desenvolvimento). Ferramentas como o Git permitem rastrear cada mudança no código-fonte, identificar facilmente a autoria de cada linha (comando blame), inspecionar diferenças entre quaisquer duas versões e reverter o projeto a um estado anterior se necessário (SPINELLIS, 2005).

Os benefícios do controle de versões no desenvolvimento de software são amplamente documentados. Além de facilitar a colaboração simultânea de diversos programadores sem conflitos (quando bem coordenada), o versionamento melhora a qualidade do software, pois cada alteração passa a ser um ponto de verificação e documentação da evolução do projeto (LOELIGER; MCCULLOUGH, 2012). Com um VCS robusto, é possível experimentar novas funcionalidades em ramos (branches) separados sem arriscar a estabilidade do código principal, identificando e isolando bugs introduzidos em commits específicos, e mantendo um registro histórico para auditoria e aprendizado. Williams e Kessler (2002) argumentam que mesmo em programação em duplas (pair programming), uma prática colaborativa, o uso de controle de versão é fundamental para disciplinar e registrar o trabalho da dupla.

Embora vários sistemas de versionamento existam (como SVN (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2004), Mercurial (O'SULLIVAN, 2009) e Perforce, entre outros), o Git destacou-se por sua velocidade, flexibilidade e forte suporte a ramificações. Ele tornou-se um padrão de fato, auxiliado por plataformas como GitHub e GitLab que adicionaram funcionalidades sociais e de gerenciamento de projetos ao fluxo de versionamento.

Transpondo esses conceitos para o contexto de documentos de texto e educação, vemos aplicações semelhantes. Por exemplo, wikis educacionais (como a Wikipedia ou wikis de curso) mantêm histórico de todas as edições de cada página, permitindo reverter vandalismos ou analisar a evolução do conteúdo. Ferramentas como o Microsoft Word oferecem controle de alterações (track changes) para revisão de documentos, o que não deixa de ser uma forma de versionamento linear com marcações de autoria. Em cenários de aprendizagem colaborativa, integrar um sistema de controle de versão pode aprimorar a organização e responsabilidade dos estudantes. Hochmüller e Maurer (2018) discutem que o uso de sistemas de versionamento em ambientes de aprendizagem colaborativa estimula boas práticas de coordenação e documenta o processo de construção de conhecimento de forma mais transparente. Huang, Liu e Zhang (2008) implementaram um ambiente virtual

de aprendizagem baseado em versionamento no qual os estudantes podiam colaborar em projetos de programação e a plataforma registrava todas as contribuições no estilo de um repositório de software . Os autores relataram que essa abordagem facilitou a identificação de contribuições individuais e evitou conflitos de edição, ao mesmo tempo em que os alunos aprenderam sobre ferramentas profissionais de desenvolvimento.

No caso específico do Etherpad, embora ele não use Git ou outro VCS tradicional internamente, ele de certo modo implementa um controle de versão customizado para os documentos (pads). Como vimos, cada alteração é registrada no histórico e pode ser revisitada via time slider, o que oferece algumas das vantagens do versionamento: rastreabilidade, recuperação e análise do processo de edição. No entanto, há diferenças importantes: o Etherpad trabalha em escala de caracteres e segundos (capturando cada digitação) e converge automaticamente as edições concorrentes, enquanto um sistema como Git trabalha em escala de linhas/arquivos e normalmente requer intervenção humana para decidir merges em caso de conflitos. Em aplicações futuras, pode-se imaginar integrar o melhor dos dois mundos – por exemplo, exportar periodicamente snapshots de pads Etherpad para um repositório Git, combinando edição em tempo real com auditoria e ramificação externas. De fato, já existem scripts e guias da comunidade para exportar ou sincronizar pads do Etherpad com repositórios Git, permitindo commits periódicos para fins de backup e acompanhamento estruturado [3].

Em suma, o controle de versões é um conceito transversal que aporta benefícios claros onde múltiplos agentes atuam sobre um mesmo artefato digital. Seja no desenvolvimento de código ou na coautoria de documentos, a capacidade de registrar, comparar e reverter alterações aumenta a confiança no processo colaborativo. Para este trabalho, compreender os princípios de versionamento reforça a importância de aproveitar o histórico detalhado fornecido pelo Etherpad e pensar em estratégias de análise e integração desses dados de forma similar ao que se faz com repositórios de software: extraindo métricas (quem contribuiu mais, em que momento), garantindo accountability (toda edição fica registrada) e fornecendo meios de restaurar conteúdo caso ocorra algum erro ou comportamento indesejado durante a colaboração.

[3] Automatically Storing Etherpad Pad contents in Git. Disponível em: <https://mclear.co.uk/2020/04/25/automatically-storing-etherpad-pad-contents-in-git/>. Acesso em: 06 out. 2025.

3 Metodologia

Metodologicamente, este projeto caracterizou-se como uma pesquisa aplicada de natureza tecnológica, pois teve como objetivo aplicar conhecimentos técnicos para desenvolver uma solução prática a um problema específico no contexto educacional (PIURA et al., 2014). Para alcançar o objetivo proposto, seguiu-se um conjunto de etapas estruturadas de desenvolvimento, conforme descrito a seguir.

Quanto à natureza da pesquisa, tratou-se de um trabalho com caráter prescritivo, cujo objetivo foi propor soluções específicas para problemas identificados, fornecendo recomendações e diretrizes para a implementação prática dessas soluções. Segundo Gregor (2006), pesquisas de caráter prescritivo focam em "como algo deve ser feito", visando a aplicação prática dos resultados obtidos.

O primeiro passo consistiu na realização de um levantamento bibliográfico e documental abrangente sobre os temas correlatos ao trabalho, como Aprendizagem Colaborativa Suportada por Computador (CSCL), ferramentas colaborativas de programação, controle de versão, bem como os aspectos técnicos do Etherpad e do LARA. Paralelamente, foi conduzido o levantamento de requisitos funcionais e não funcionais da aplicação (objetivo específico a)). Essa etapa incluiu entrevistas e conversas com a equipe de pesquisadores do LARA, com o objetivo de coletar informações detalhadas sobre as necessidades e expectativas em relação ao sistema. Os requisitos funcionais definiram as funcionalidades esperadas — como edição colaborativa em tempo real e integração com o AVA — enquanto os requisitos não funcionais abrangeram aspectos como desempenho, segurança, usabilidade e compatibilidade com a infraestrutura existente.

Em seguida, procedeu-se à definição e modelagem da arquitetura do sistema (b)), utilizando tecnologias de containerização e orquestração. Optou-se por uma arquitetura em que o Etherpad fosse executado em um contêiner Docker, gerenciado por um ambiente Kubernetes. Nessa fase, foram empregadas ferramentas de modelagem, como diagramas UML, para visualizar a estrutura do sistema e as interações entre seus componentes. Avaliaram-se diferentes alternativas de integração — por exemplo, incorporação via *iframe* ou por meio da API REST do Etherpad (Etherpad Foundation, 2024); persistência em banco de dados local ou em serviço externo — e definiu-se uma arquitetura modular, na qual o Etherpad opera como um serviço independente, containerizado, comunicando-se com o LARA por meio de APIs e compartilhando um banco de dados para persistência das edições. As configurações estabelecidas buscaram garantir escalabilidade (possibilidade de múltiplas instâncias atuando simultaneamente) e alta disponibilidade, assegurando que nenhum dado fosse perdido mesmo em casos de reinicialização dos contêineres (The Kubernetes Authors, 2025).

Para gerenciar a implementação do sistema, adotou-se a metodologia ágil Kanban. Essa escolha mostrou-se adequada dado que se tratou de um projeto de escopo variável

desenvolvido majoritariamente por um único membro (com orientação), exigindo flexibilidade e adaptabilidade. Segundo Anderson (2010), Kanban é "um método para gerenciar o trabalho do conhecimento com um foco na entrega contínua enquanto não sobrecarrega a equipe de desenvolvimento". O Kanban possibilitou visualizar o fluxo de tarefas (como *to do*, *doing*, *done*) e priorizá-las de forma dinâmica, evitando sobrecarga e gargalos no processo. Reuniões semanais de acompanhamento com a orientadora foram realizadas para revisar o progresso, ajustar requisitos e planejar as próximas atividades, garantindo que o desenvolvimento permanecesse alinhado aos objetivos propostos.

Após a definição da arquitetura e do planejamento das atividades, iniciou-se a implementação da solução. Desenvolveu-se um *Dockerfile* personalizado (c)) para gerar uma imagem Docker do Etherpad adequada ao ambiente do LARA — incluindo todas as dependências (Node.js, bibliotecas do Etherpad), eventuais plugins e configurações necessárias para que o Etherpad funcionasse de forma consistente em qualquer ambiente. Segundo a documentação oficial do Docker Docker Inc. (2025), um Dockerfile é um arquivo de texto que contém todos os comandos para montar automaticamente a imagem de uma aplicação, garantindo reprodutibilidade no *deploy*. Dessa forma, consolidou-se uma versão estável do Etherpad, personalizada para o LARA. Em paralelo, configurou-se o banco de dados do Etherpad: optou-se por utilizar o PostgreSQL como repositório de dados, em substituição ao banco padrão em arquivo (DirtyDB), a fim de assegurar persistência e confiabilidade nas informações dos documentos. O contêiner do banco de dados também foi preparado via Docker, facilitando sua implantação. Com a imagem pronta, a aplicação foi implantada em um cluster Kubernetes local, configurando *deployments*, serviços e volumes persistentes conforme planejado na arquitetura.

Com essas etapas, a metodologia garantiu que o desenvolvimento ocorresse de maneira organizada e alinhada aos objetivos. O resultado foi um protótipo funcional do Etherpad, configurado e otimizado para futura integração ao LARA, estando pronto para ser avaliado em cenários reais de sala de aula ou em estudos posteriores. Esse processo metodológico não apenas permitiu atingir o objetivo geral do projeto, como também forneceu diretrizes e aprendizados para futuras iniciativas de integração de ferramentas colaborativas em ambientes de aprendizagem similares.

4 Preparação do Módulo de Edição Colaborativa (Etherpad) para o AVA LARA

4.1 Etherpad: Editor colaborativo em tempo real

O Etherpad é uma ferramenta open-source de edição colaborativa de texto em tempo real, amplamente utilizada em diversos contextos por permitir que múltiplos usuários editem simultaneamente um documento através do navegador [1]. Originalmente lançado em 2008 pela startup AppJet, o Etherpad ganhou notoriedade por ser um dos primeiros editores colaborativos on-line de código aberto acessíveis ao público. Após ser adquirido e aberto pelo Google em 2009, a comunidade assumiu seu desenvolvimento. A versão atual é frequentemente chamada de Etherpad Lite, resultado de uma reimplementação quase completa do software original, agora em JavaScript server-side (Node.js) ao invés de Java/Scala. Essa reescrita tornou o Etherpad muito mais leve em termos de requisitos de servidor e mais fácil de manter e estender. De fato, o Etherpad moderno difere do "EtherPad" original em sua base técnica: enquanto o primeiro necessitava de um servidor robusto e não é mais mantido, o Etherpad Lite é altamente modular e continua em ativo desenvolvimento pela Etherpad Foundation. Hoje, o termo "Etherpad" refere-se geralmente a essa nova versão.

Do ponto de vista do usuário, o Etherpad fornece uma interface simples e eficiente. Cada documento colaborativo é chamado de Pad. A seguir, resumimos os principais recursos e características do Etherpad:

- **Edição simultânea com destaque de autores:** todas as edições aparecem em tempo real a todos os participantes do pad. Cada usuário é identificado por um nome ou pseudônimo e, opcionalmente, por uma cor de destaque, que colore o fundo do texto que ele digitou (figura 1). Isso permite rastrear visualmente quem contribuiu com cada parte do texto, facilitando a colaboração transparente. Esse mecanismo de autoria visível fundamenta a necessidade de registrar o processo de construção coletiva.

- **Histórico de versões e "Time Slider":** o Etherpad registra automaticamente cada alteração realizada no documento, criando um histórico completo de revisões em nível de caracter (keystroke-by-keystroke). Por meio da funcionalidade de Time Slider (deslizante de tempo), é possível reproduzir a evolução do documento ao longo do tempo, visualizando passo a passo tudo o que foi adicionado ou removido e em que ordem. Conforme a figura 4, essa característica não apenas auxilia na recuperação

[1] Wikipedia contributors. Etherpad — Wikipedia, The Free Encyclopedia. Disponível em: <https://en.wikipedia.org/wiki/Etherpad>. Acesso em: 06 out. 2025.

de versões anteriores do texto, mas também é útil em contextos educacionais para análise do processo de coedição pelo professor ou pesquisador. É possível ainda exportar todo o histórico de edição para fins de auditoria ou estudo, uma vez que o Etherpad armazena os conjuntos de diffs (chamados changesets) que compõem cada revisão. Como a colaboração também envolve coordenação em tempo real, o Etherpad provê um canal de comunicação dedicado.

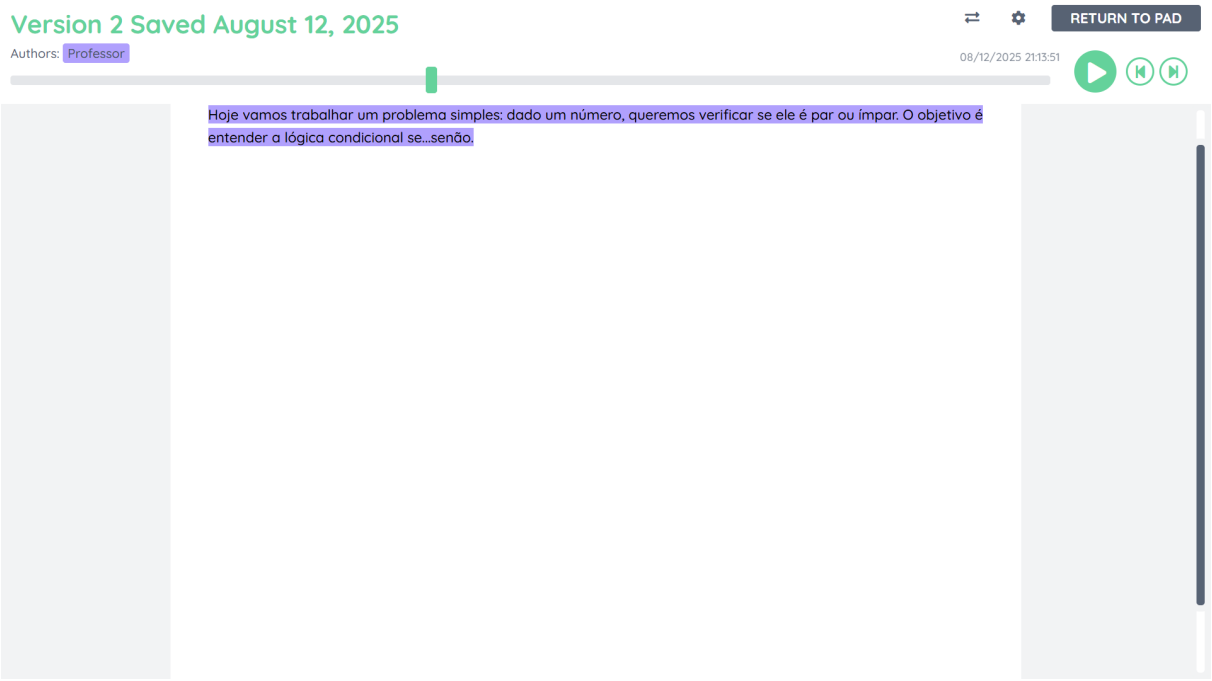

Figura 4 – Time Slider da pad exibida na figura 1

- **Chat integrado:** paralelamente à área de edição, cada pad possui uma janela de chat em tempo real. A figura 1 mostra como o chat permite que os colaboradores conversem, discutam ideias ou coordenem a edição sem alterar o conteúdo principal do documento. As mensagens de chat também são identificadas com horário e autor. Embora seja uma funcionalidade simples, mostra-se importante para a coordenação social da tarefa colaborativa, sobretudo em grupos maiores. Encerrada a coordenacao síncrona, torna-se frequente a necessidade de importar materiais e, ao final, exportar o produto do trabalho colaborativo.

- **Importação e exportação de documentos:** o Etherpad possibilita importar conteúdo inicial (por exemplo, um arquivo texto ou HTML) para dentro de um pad. Ao término da edição, o conteúdo final pode ser exportado em diversos formatos, incluindo texto simples, HTML, ODF, PDF e Microsoft Word. Dessa forma, o pad pode ser utilizado como ferramenta de produção de documentos que depois serão publicados ou reutilizados em outras plataformas, como observado na figura 5. Em cenários institucionais, todavia, a produção de conteúdo em escala requer criação e

controle automatizados de pads e permissões, recursos estes viabilizados pela API do Etherpad.

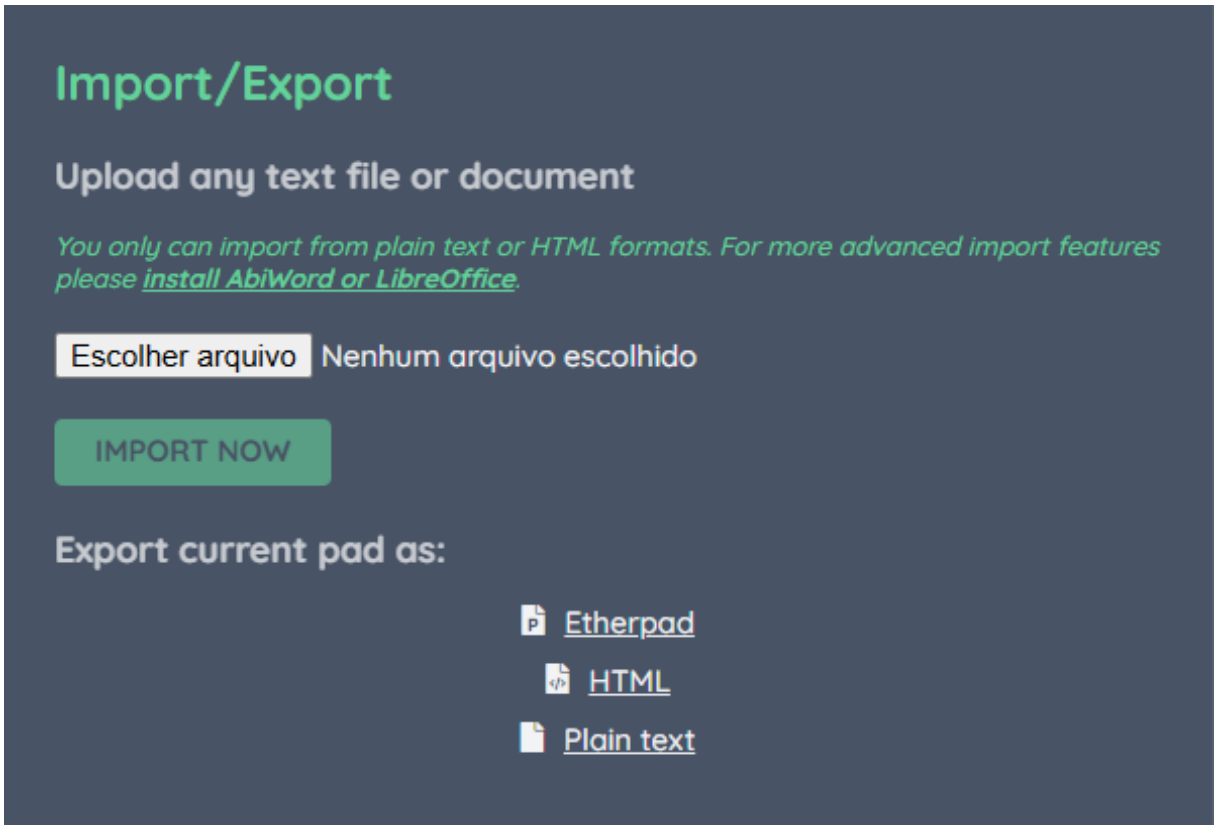

Figura 5 – Função de *import* e *export* de PADs.

- **Gerenciamento de usuários e pads (via API):** por padrão, qualquer pessoa com o link de um pad pode acessá-lo e editar (dependendo da configuração do servidor). Entretanto, para integrações mais estruturadas (como em um AVA), o Etherpad fornece uma API HTTP que permite a criação e gerenciamento de pads, grupos de pads e usuários autorizados de forma programática (Etherpad Foundation, 2024). Por exemplo, a API permite criar um pad novo via chamada REST, associá-lo a um determinado grupo (sala/turma) e até atribuir permissões de edição ou somente leitura. No caso do LARA, Lopes (2017) utilizou essa API para criar pads vinculados aos grupos formados no ambiente, garantindo que somente os membros do grupo tivessem acesso ao pad correspondente. Para além da orquestracao via API, a plataforma pode ser estendida com novas capacidades por meio de sua arquitetura modular de plugins.

- **Estrutura modular e plugins:** uma das forças do Etherpad está em sua arquitetura extensível. O servidor Etherpad é construído em Node.js e possui um sistema de plugins robusto. Há dezenas de plugins disponíveis na comunidade que adicionam funcionalidades diversas, como suporte a edição de fórmulas matemáticas (via

LaTeX), destaque de sintaxe de código fonte, exportação do histórico em formatos especiais, autenticação via Moodle, entre outros [2].. Esse framework de extensões permite adaptar o Etherpad a diferentes necessidades sem alterar o núcleo da aplicação. Conforme Han (2011) observa, mecanismos de plugin tornam o software mais flexível e evolutivo, pois novas capacidades podem ser incorporadas como módulos independentes. No contexto educacional, isso significa que é possível, por exemplo, adicionar ao Etherpad ferramentas de avaliação colaborativa, gráficos de participação ou outras funcionalidades que suportem o professor, tudo via plugins. O enriquecimento funcional, por sua vez, precisa ser sustentado por uma infraestrutura capaz de atender maiores cargas e topologias distribuídas, tema este abordado no próximo item.

- **Escalabilidade e uso distribuído:** embora o Etherpad seja frequentemente usado em implementações modestas (por exemplo, um servidor para uma universidade ou empresa), ele também pode ser escalado para suportar grande número de usuários. Projetos como o Scale Etherpad[3]. demonstram a escalabilidade do software, permitindo centenas de usuários editando simultaneamente em um mesmo pad de teste. Além disso, existe a iniciativa Etherpad Lite em cluster, que permite rodar múltiplas instâncias colaborando para atender a maior carga. Ferramentas de monitoramento específicas, como o Etherpad Scanner, foram desenvolvidas para rastrear instâncias públicas de Etherpad ao redor do mundo e coletar insights sobre seu uso. Isso evidencia que o Etherpad possui uma base de usuários significativa globalmente e tem se mantido relevante ao longo do tempo como solução de colaboração em texto[4].. Com a adoção em larga escala, emergem também preocupações de segurança e privacidade, apresentadas a seguir.

- **Segurança e privacidade:** como software aberto, o Etherpad pode ser hospedado internamente por uma instituição, oferecendo controle total sobre os dados (diferentemente de serviços SaaS de documentos colaborativos). Ele suporta conexões seguras (SSL/TLS) e autenticação de usuários quando integrado a sistemas externos. Entretanto, por padrão, um Etherpad público permite criação livre de pads sem login, o que requer atenção para não expor conteúdos confidenciais inadvertidamente. A administração de uma instância Etherpad envolve configuração de políticas de retenção de dados do pad e possíveis mecanismos de limpeza dos pads inativos.

Em resumo, o Etherpad se destaca por fornecer o essencial para edição colaborativa de forma simples e eficaz, ao mesmo tempo em que possibilita customizações avançadas conforme a necessidade. Sua estrutura interna baseia-se em tecnologias web comuns: no

[2] Etherpad plugins. Disponível em: <https://static.etherpad.org/index.html>. Acesso em: 06 out. 2025.
[3] Etherpad Scale Calculator. Disponível em: <https://scale.etherpad.org/>. Acesso em: 06 out. 2025.
[4] Etherpad Scanner. Disponível em: <https://scanner.etherpad.org/>. Acesso em: 06 out. 2025.

front-end, utiliza JavaScript para atualizar o documento em tempo real via técnicas de comet ou websockets; no back-end, a lógica Node.js mantém as sessões dos pads, aplica o algoritmo OT para mesclar edições concorrentes e interage com o banco de dados para persistir o conteúdo e o histórico. A figura 6 apresenta um diagrama simplificado da arquitetura do Etherpad, destacando os componentes principais: clientes web (navegadores) conectados ao servidor Etherpad, o módulo de controle de pads (que gerencia sessões, usuários e edição em tempo real), o subsistema de persistência (banco de dados) e as interfaces de API/plugin para extensibilidade. Essa compreensão arquitetural é fundamental para integrarmos o Etherpad de maneira eficaz em ambientes como o LARA, bem como para dimensionar seu uso e adaptar seus dados (por exemplo, extrair métricas do histórico de edições para avaliação educacional).

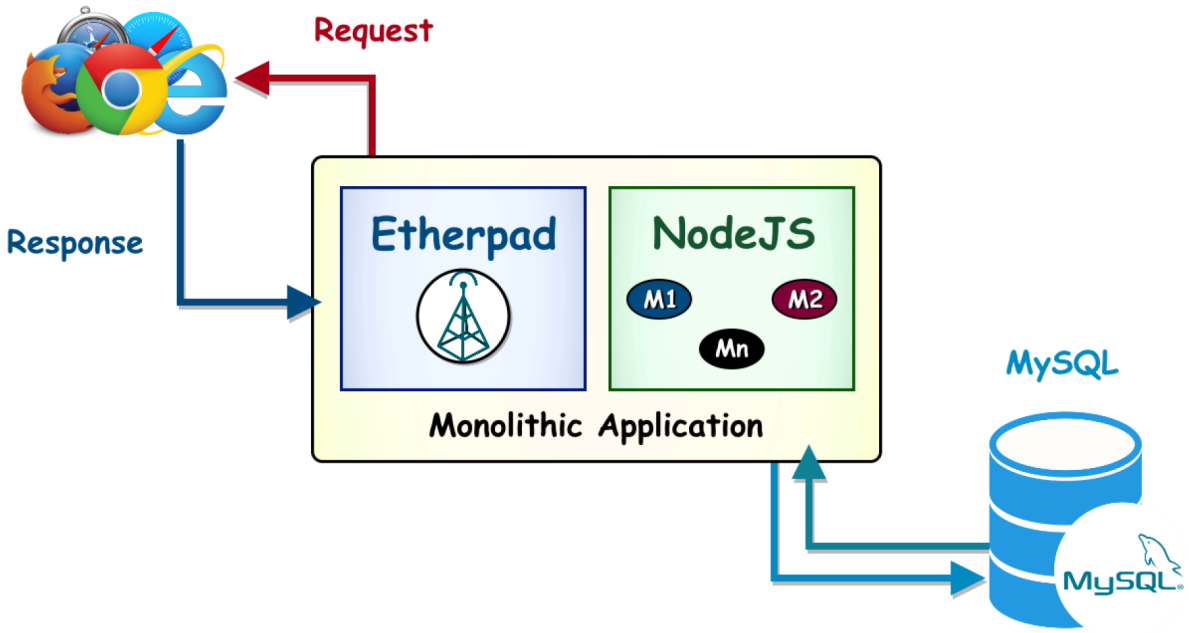

Figura 6 – Diagrama simplificado da arquitetura do Etherpad (LEON et al., 2020).

4.2 Requisitos funcionais e não funcionais

A elicitação de requisitos foi realizada de forma iterativa e colaborativa, por meio de reuniões semanais entre o aluno e a professora orientadora do projeto. Nessas sessões, discutiram-se os objetivos pedagógicos e técnicos definidos na proposta inicial e no capítulo 1, bem como detalhes das soluções tecnológicas exploradas nos capítulos 2 e 4. Com base nessas conversas orientativas informais, foi possível refinar a compreensão do escopo do projeto e identificar as funcionalidades necessárias para integrar o Etherpad ao AVA LARA. Em particular, enfatizou-se o contexto educacional do sistema – ou seja, o uso do Etherpad como ferramenta colaborativa para iniciantes em programação – de modo que os

requisitos capturados refletissem as necessidades de aprendizagem colaborativa e facilidade de uso pelo público-alvo. Estudos prévios sugerem que abordagens colaborativas podem trazer benefícios significativos no ensino de programação para novatos, aumentando o engajamento e a construção de conhecimento em grupo (BRAVO; DUQUE; GALLARDO, 2013). Assim, muitos dos requisitos definidos visam alinhar as funcionalidades do Etherpad a princípios de aprendizagem colaborativa suportada por computador (CSCL) no contexto do LARA. Ao final desse processo de análise, obteve-se um conjunto de requisitos funcionais e não funcionais que servem para guiar o desenvolvimento da solução. Esses requisitos foram cuidadosamente validados junto à orientadora para garantir alinhamento estrito com os objetivos educacionais e técnicos do projeto, assegurando que a integração proposta atenda às expectativas pedagógicas e às restrições de infraestrutura do ambiente. A seguir, apresenta-se um resumo dos principais requisitos funcionais e não funcionais levantados.

4.2.1 Requisitos funcionais

Os requisitos funcionais definem os comportamentos e funcionalidades que o sistema deverá oferecer aos usuários. Com base nos objetivos do projeto – como incentivar a colaboração em atividades de programação e proporcionar ferramentas de apoio ao aprendizado ativo – e nas capacidades do Etherpad estudadas, foram estabelecidos os seguintes requisitos funcionais principais:

- **Edição colaborativa em tempo real:** Permitir que múltiplos usuários editem simultaneamente uma mesma pad dentro do AVA, com atualizações em tempo real visíveis a todos os participantes. Cada participante deve ser identificado no pad pelo seu nome ou apelido, e as edições devem ser destacadas com cores para indicar a autoria.

- **Comunicação integrada por chat:** Disponibilizar um canal de comunicação textual em paralelo à área de edição colaborativa. Esse chat integrado ao pad permitirá que os alunos conversem e coordenem suas ações durante a atividade, sem interromper o conteúdo principal do documento. Essa funcionalidade de comunicação síncrona é importante para a coordenação social da tarefa colaborativa, auxiliando na troca de ideias e na solução conjunta de problemas enquanto programam em equipe.

- **Gerenciamento de pads vinculado às turmas/grupos do AVA:** Integrar o Etherpad ao gerenciamento de usuários e grupos do ambiente LARA. Pads devem ser criados e associados automaticamente a grupos de alunos ou turmas definidos no AVA, de forma que cada grupo tenha seu espaço colaborativo exclusivo. Dessa maneira, garante-se que somente os membros autorizados de um determinado grupo possam

acessar e editar o pad correspondente, reforçando a privacidade e a organização por turmas (LOPES, 2017).

- **Histórico de versões e monitoração pelo docente:** Prover meios para que o professor possa acompanhar a evolução do trabalho colaborativo de cada grupo. O sistema deve manter um histórico detalhado de todas as alterações em cada pad e permitir sua inspeção através de uma interface adequada – por exemplo, o recurso de reprodução temporal do Etherpad (Time Slider) pode ser utilizado para visualizar passo a passo as edições realizadas. Com isso, o docente ou pesquisador consegue analisar as contribuições individuais de cada aluno e o processo de construção coletiva da solução ao longo do tempo, o que atende a objetivos de avaliação formativa e pesquisa educacional.
- **Exportação e reutilização do conteúdo produzido:** Permitir que o conteúdo colaborativo desenvolvido nos pads seja exportado em formatos padrões (como PDF, texto simples ou HTML), viabilizando seu aproveitamento fora do ambiente Etherpad. Essa funcionalidade facilita, por exemplo, que um grupo submeta o resultado final de sua atividade para avaliação ou que o material produzido seja publicado em outras plataformas acadêmicas, sem perda de formatação básica.
- **Facilidades para o professor criar e gerenciar atividades colaborativas:** Oferecer ao professor mecanismos simples para configurar atividades de edição colaborativa. Por exemplo, o docente deve conseguir criar um pad novo para uma determinada tarefa ou grupo a partir da interface do AVA (usando a própria API do etherpad, como visto em Etherpad Foundation (2024)) ou da própria interface de administrador da aplicação, sem necessidade de procedimentos manuais complexos. O professor pode ter permissões especiais, como tornar um pad somente leitura após o término da atividade ou restaurar uma versão anterior do documento, contribuindo para o gerenciamento pedagógico das atividades.

4.2.2 Requisitos não funcionais

Além das funcionalidades, foram definidos requisitos não funcionais para garantir que a solução atenda a critérios de qualidade, desempenho e compatibilidade esperados em um ambiente educacional online. Tais requisitos asseguram que a integração do Etherpad ocorra de maneira eficaz no contexto do AVA LARA, mantendo a ferramenta confiável, segura e utilizável. Seguem os principais requisitos não funcionais identificados:

- **Desempenho e escalabilidade:** O sistema deve suportar múltiplos usuários editando simultaneamente sem degradação perceptível de desempenho (atrasos ou travamentos significativos). Mesmo em atividades com turmas numerosas, a edição colaborativa precisa permanecer responsiva e em tempo real. A solução projetada

deve ser escalável, possibilitando o aumento de recursos do servidor ou a adição de instâncias adicionais do Etherpad para acomodar picos de demanda, se necessário (LEON et al., 2020).

- **Usabilidade e acessibilidade:** A interface do editor colaborativo deve ser intuitiva e de fácil uso, considerando o público-alvo de iniciantes em programação que pode não ter familiaridade com ferramentas complexas. Deve-se prezar por um design limpo e integrado visualmente ao restante do AVA, reduzindo a curva de aprendizado e evitando distrações desnecessárias. Também é importante que a ferramenta seja acessível através dos principais navegadores web e dispositivos, sem requerer instalação de software adicional pelos alunos, possibilitando o acesso remoto de qualquer lugar com facilidade.

- **Manutenibilidade e implantação:** A solução integrada (Etherpad + LARA) deve ser de fácil manutenção a longo prazo. Isso inclui a capacidade de atualizar o Etherpad ou seus plugins sem impactar negativamente o AVA, bem como facilidade para aplicar ajustes de configuração. A implantação do Etherpad deve utilizar tecnologias compatíveis com a infraestrutura existente – por exemplo, contêineres Docker e orquestração Kubernetes – de modo a simplificar o gerenciamento do serviço (p. ex., reinicialização, escalonamento, backup de dados). Essa preocupação garante que a equipe técnica da instituição possa administrar o recurso colaborativo sem esforço excessivo, mantendo alta disponibilidade do sistema.

- **Extensibilidade e adaptabilidade:** Deve ser possível estender ou adaptar as funcionalidades da ferramenta no futuro, acompanhando eventuais novas demandas pedagógicas ou tecnológicas. Graças à arquitetura modular do Etherpad, que suporta plugins, a integração realizada deve permitir a adição de novos plugins ou desenvolvimento de funcionalidades personalizadas sem a necessidade de alterar o núcleo do sistema. Por exemplo, caso deseje-se no futuro incorporar métricas de participação dos alunos ou integrar o editor com algum repositório de códigos, essas extensões poderão ser realizadas de forma incremental, preservando a estabilidade do ambiente principal.

- **Compatibilidade e integração transparente:** O Etherpad deve funcionar de forma transparente dentro do ecossistema do AVA LARA. Isso significa que a experiência do usuário ao utilizar o editor colaborativo deve ser fluida e consistente, como se fosse parte integrante do próprio AVA. Tecnologicamente, a integração deve fazer uso de interfaces e padrões adequados (por exemplo, incorporando o pad via iframe ou usando chamadas à API do Etherpad) para sincronizar dados de usuários e grupos em tempo real. Ademais, é fundamental que futuras integrações com outras plataformas (por exemplo, um LMS externo ou ferramenta de análise de dados

educacionais) não sejam impedidas – em outras palavras, a solução deve aderir a padrões abertos de interoperabilidade, evitando amarrações proprietárias.

4.3 Tecnologias utilizadas

A implementação do Etherpad no AVA LARA envolveu uma série de tecnologias de suporte, que foram fundamentais tanto para o desenvolvimento quanto para a implantação da solução em um ambiente real. Dentre essas tecnologias destacam-se: conteinerização com Docker, orquestração de serviços com Kubernetes (utilizando-se o Minikube durante o desenvolvimento), PostgreSQL para a persistência e armazenamento de dados, a plataforma de execução Node.js e o uso de plugins de software para extensibilidade. Nesta seção, descrevemos cada uma delas e seu papel no contexto do projeto, oferecendo a fundamentação técnica para as escolhas realizadas.

4.3.1 Conteinerização com Docker

O Docker é uma plataforma amplamente utilizada para criação e gerenciamento de contêineres de software. Um contêiner pode ser entendido como um pacote padronizado que encapsula uma aplicação e todas as suas dependências (bibliotecas, configurações, etc.) em um ambiente isolado, porém leve, que pode ser executado de forma consistente em qualquer sistema que tenha o Docker instalado (Docker Inc., 2025). Diferentemente de máquinas virtuais tradicionais, os contêineres compartilham o núcleo do sistema operacional do host, o que os torna muito mais eficientes em termos de desempenho e uso de recursos.

No contexto da implementação do Etherpad, o Docker trouxe diversas vantagens. Primeiramente, permitiu empacotar o Etherpad e seus componentes (por exemplo, servidor Node.js, bibliotecas necessárias, configurações do Etherpad e do banco de dados) em uma imagem reproduzível. Isso assegura que o ambiente de produção seja idêntico ao ambiente de desenvolvimento e teste, eliminando problemas causados por diferenças de configuração ou versões de dependências. Além disso, usando o Docker pudemos facilmente utilizar uma imagem pré-configurada do Etherpad disponível no Docker Hub, acelerando o setup inicial. A modularidade do Docker também simplificou a gestão de serviços complementares: por exemplo, executar o banco de dados PostgreSQL em um contêiner separado, o Etherpad em outro, e eventualmente outros componentes do LARA, todos isolados porém podendo se comunicar em uma mesma rede virtual interna.

Outra vantagem crucial é o isolamento proporcionado: o Etherpad rodando em um contêiner Docker não interfere nas outras aplicações do servidor host, e vice-versa. Isso aumenta a segurança e confiabilidade da solução. Em caso de necessidade de escalabilidade, o Docker facilita subir instâncias adicionais do Etherpad (contêineres replicados) atrás de um balanceador de carga, embora para o escopo deste trabalho uma instância única tenha atendido adequadamente.

Em resumo, o Docker serviu como base para uma implantação portável e escalável do Etherpad. Ele se alinha a práticas modernas de DevOps, permitindo integrar a implantação do Etherpad no LARA a pipelines de deploy contínuo, testes automatizados de integração em contêiner, etc. A figura 7 ilustra o encapsulamento do Etherpad e do banco de dados em contêineres Docker separados, ressaltando como eles se comunicam através de redes definidas pelo Docker, e como o ambiente externo acessa o serviço Etherpad através de uma porta exposta do contêiner.

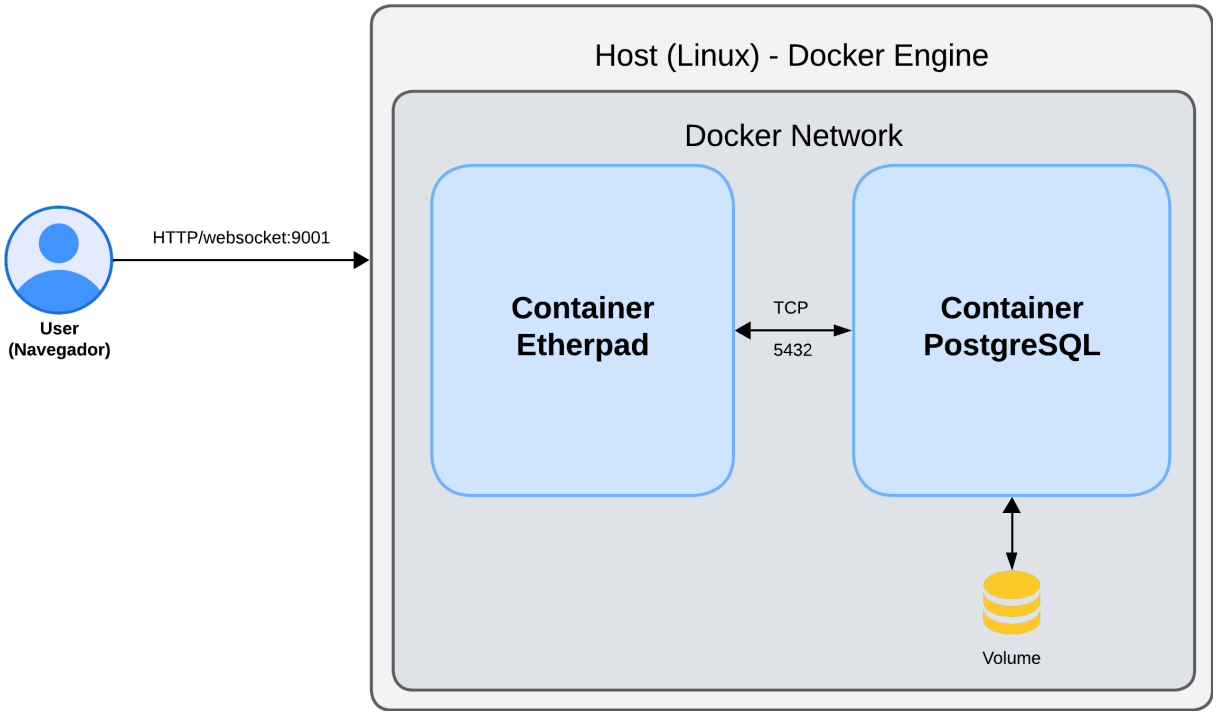

Figura 7 – Conteinerização do Etherpad e do PostgreSQL em Docker.

4.3.2 Orquestração com Kubernetes e Minikube

Conforme a quantidade de contêineres e serviços cresce, torna-se importante utilizar ferramentas de orquestração de contêineres. O Kubernetes é a plataforma líder nesse domínio, fornecendo um conjunto de recursos para implantar, escalar e gerenciar aplicações conteinerizadas em cluster de máquinas (The Kubernetes Authors, 2025). Com o Kubernetes, é possível descrever em arquivos de configuração (manifestos) como a aplicação deve rodar – quantas instâncias, quais recursos de hardware alocar, como os contêineres interagem em rede, políticas de reinício em caso de falhas, balanceamento de carga, etc. Em essência, o Kubernetes abstrai a infraestrutura subjacente e garante que a aplicação esteja sempre nas condições desejadas (self-healing, elasticidade, atualização gradual de versões, etc.).

Para efeitos de desenvolvimento e testes locais, utilizamos o Minikube, que é uma distribuição leve do Kubernetes projetada para rodar em um único nó (uma única

máquina, como o laptop do desenvolvedor) (KUBERNETES, 2025). O Minikube simula todo o sistema Kubernetes (nó mestre e nó de trabalho) em uma VM ou diretamente no host, permitindo experimentar os benefícios da orquestração sem necessidade de um cluster real na nuvem. Isso facilitou testar cenários como: reinicialização automática do Etherpad em caso de falha (usando deployments do Kubernetes com política de restart), montagem de volumes persistentes para o banco de dados (garantindo que os dados do PostgreSQL não se perdessem ao atualizar o contêiner), e exposição do serviço Etherpad através de um Service.

Uma vez validada a orquestração no Minikube, os mesmos manifestos de configuração podem ser aplicados em um cluster Kubernetes real (em nuvem ou on-premise), garantindo consistência entre ambientes. Isso evidencia um dos maiores benefícios do Kubernetes: a portabilidade e escalabilidade transparente. Conforme Burns et al. (2022), caso o uso do Etherpad no LARA se expandisse para muitos usuários simultâneos, seria simples ajustar no Kubernetes o número de réplicas do pod Etherpad, ou configurar auto-scaling baseado em CPU/memória. Da mesma forma, se quisermos atualizar a versão do Etherpad, o Kubernetes permite realizar rollout gradual (atualizando contêineres um a um para a nova versão, sem perca de tempo perceptível, se configurado adequadamente).

Em síntese, o Kubernetes providencia um nível de automação e resiliência que complementa o uso do Docker. No escopo acadêmico deste trabalho, seu uso demonstrou as possibilidades de implantação robusta e serviu como aprendizado de tecnologias de ponta em implantação de aplicações.

A Figura 8 apresenta a implantação sob Kubernetes de forma integrada: o acesso do usuário chega a um Service (NodePort), que encaminha as requisições para os pods do Etherpad. Esses pods são mantidos por um Deployment, o qual controla um ou mais ReplicaSets responsáveis por garantir o número desejado de réplicas. Em paralelo, o PostgreSQL é orquestrado por um StatefulSet, cujos pods estão associados a volumes persistentes (PV) provisionados via PVC, assegurando a durabilidade dos dados. Os pods do PostgreSQL são consumidos pelos pods do Etherpad via um service interno (ClusterIP).

4.3.3 Node.js como plataforma de execução

O Node.js é um ambiente de execução de código JavaScript no lado do servidor, construído sobre o motor V8 do Google Chrome. A tecnologia tornou-se muito popular por seu modelo de I/O não bloqueante e orientado a eventos[5], que permite alto desempenho em aplicações web em tempo real. O Etherpad foi reescrito utilizando Node.js justamente para se beneficiar dessas características e da vasta comunidade de desenvolvedores JavaScript.

Do ponto de vista deste trabalho, é importante compreender que o Etherpad em si é uma aplicação Node.js. Isso significa que para executá-lo precisamos de uma versão

[5] NodeSource Team. How Node.js Works: A Comprehensive Guide in 2025. Disponível em: <https://nodesource.com/blog/how-nodejs-works/>. Acesso em: 06 out. 2025.

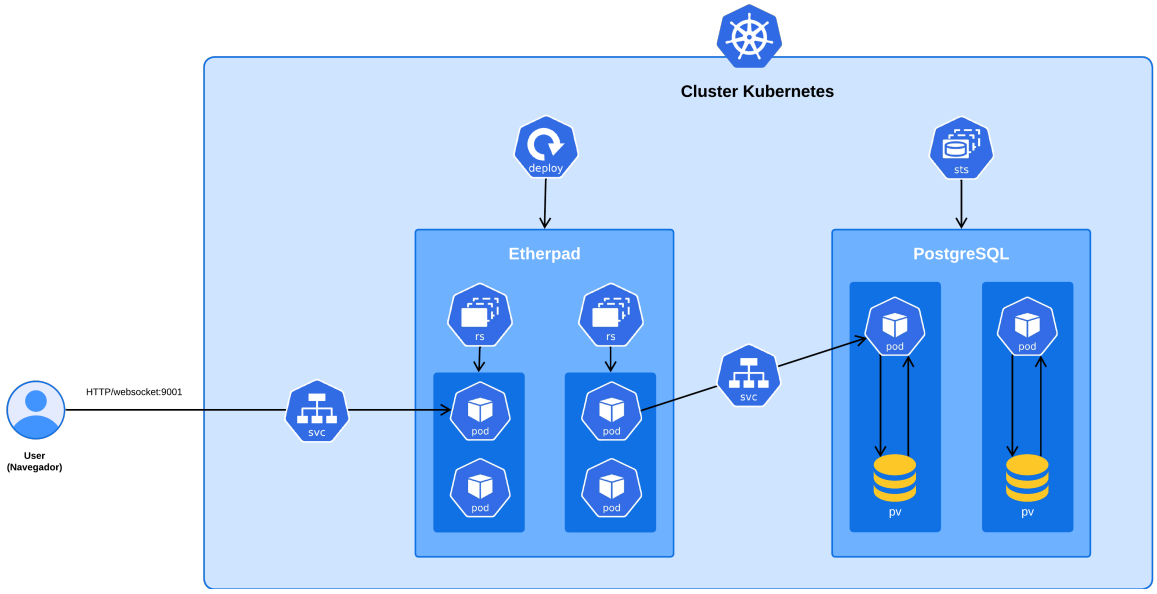

Figura 8 – Implantação no Kubernetes: o acesso do usuário chega ao Service, que roteia para os pods do Etherpad (Deployment → ReplicaSet → Pods); o PostgreSQL é gerenciado por um StatefulSet, com volumes persistentes (PVC/PV) e exposto apenas internamente via Service.

compatível do Node (18.18.2 ou superior [6]) e de um conjunto de pacotes obtidos via npm (o gerenciador de pacotes do Node). A escolha do Node.js traz algumas implicações técnicas:

- A escalabilidade vertical do Etherpad (isto é, quantos usuários pode atender em uma única instância) depende em parte do single-thread event loop do Node. O Node lida muito bem com múltiplas conexões concorrentes em I/O, mas operações CPU-intensivas podem bloquear a event loop. No Etherpad, a maior parte do trabalho é I/O (receber edições, repassar a outros clientes, escrever no banco), então o Node se adequa bem. Porém, para aproveitar múltiplos núcleos de CPU, seria necessário executar várias instâncias (processos) Node em paralelo – algo que se consegue facilmente com contêineres ou usando o módulo cluster do Node se desejado.

- O Node.js facilitou também o desenvolvimento de integrações customizadas. Por exemplo, se quiséssemos criar um plugin específico do Etherpad para o LARA, poderíamos escrevê-lo em JavaScript, aproveitando APIs do próprio Etherpad e milhares de bibliotecas disponíveis no ecossistema Node, além do próprio guia oficial presente em seu github. Isso reduz a curva de implementação de novas funcionalidades.

- Em termos de implantação, rodar um aplicativo Node implica gerenciar pacotes via npm. Com o Docker, isso foi encapsulado, mas vale citar que inicialmente configuramos e testamos o Etherpad diretamente no Node local, instalando suas

[6] etherpad-lite — Etherpad: A modern really-real-time collaborative document editor (GitHub repository). Disponível em: <https://github.com/ether/etherpad-lite>. Acesso em: 06 out. 2025.

dependências. Esse processo envolveu por exemplo rodar npm install no diretório do Etherpad e ajustar parâmetros no arquivo de configuração settings.json (como credenciais do PostgreSQL, porta de execução, chave de API etc.).

A escolha do Node.js para o Etherpad reflete uma tendência mais ampla de aplicações web em tempo real, e nos mostrou na prática os benefícios de uma arquitetura orientada a eventos para suportar colaboração síncrona. Durante o projeto, situações de debug envolveram analisar logs do servidor Node, entender callbacks referentes às operações do Etherpad, o que complementou nosso conhecimento sobre programação assíncrona. Para efeitos de fundamentação, podemos dizer que o Node.js comprova ser uma plataforma madura e confiável, adotada não só pelo Etherpad mas também por inúmeras outras aplicações críticas, possuindo vasta documentação e comunidade ativa.

Para concluir, as tecnologias de suporte descritas acima formam a espinha dorsal que possibilitou não apenas fazer o Etherpad funcionar no contexto do LARA, mas fazê-lo de maneira modular, escalável e mantível.

4.4 Plano de integração ao LARA

Esta subseção descreve como o Etherpad será exposto dentro do LARA, cobrindo autenticação, uso da API, provisionamento (criação) de *pads* por turma/grupo e procedimentos de encerramento e exportação. O plano baseia-se na API HTTP oficial do Etherpad e em seu modelo de dados (grupos, autores, sessões e *pads*) documentados pela Etherpad Foundation (2018), além do mapeamento pedagógico e de grupos descrito em Lopes (2017).

4.4.1 Entidades e mapeamentos

No LARA, propõe-se o seguinte mapeamento lógico (conceitos do LARA → objetos nativos do Etherpad):

- **Turma/Grupo no LARA** → *group* (ID interno do Etherpad).
- **Aluno(a) autenticado(a) no LARA** → *author* (perfil global; nome e cor).
- **Atividade colaborativa** → *pad* pertencente a um *group* (conteúdo e histórico).
- **Sessão de acesso** → *session* (vínculo temporal author–group, controlado por cookie).

Esse arranjo segue a estrutura "$group \rightarrow pad\,(groupID\$padName)$; $author$ + *session*" descrita na documentação oficial do Etherpad.

4.4.2 Autenticação e autorização

O servidor Etherpad autentica chamadas de API por meio de uma *API key* (arquivo `APIKEY.txt`) e autoriza usuários finais via *sessions*. No fluxo integrado:

1. O LARA autentica o(a) aluno(a) (SSO já existente do AVA).
2. O backend do LARA chama a API do Etherpad com a *API key* (servidor–servidor).
3. Para acesso do navegador ao *pad* de *group*, o LARA cria uma *session* e define um cookie `sessionID` (domínio do Etherpad) com tempo de expiração (*validUntil*) adequado ao período da atividade.

4.4.3 Provisionamento automático de *pads*

No início de uma atividade colaborativa, o LARA orquestra o seguinte fluxo idempotente (nomes de métodos conforme API HTTP oficial):

1. **Garantir o grupo da turma** (`createGroupIfNotExistsFor`): usa um identificador estável da turma para obter/criar *groupID*.
2. **Garantir o(a) autor(a)** (`createAuthorIfNotExistsFor`): associa o usuário do LARA a um *authorID* (com nome exibido).
3. **Criar/garantir a sessão** (`createSession`): cria *sessionID* (com *validUntil*) para o par *author–group*; definir cookie `sessionID` no domínio do Etherpad.
4. **Criar o *pad* da atividade no grupo** (`createGroupPad`): padID lógico do tipo `<groupID$atividade_AAAAMMDD>`, com texto inicial (`setText`) e configurações (p.ex., `setPublicStatus`, `setPassword` opcional).
5. **Incorporar no LARA** via *iframe* apontando para `/p/<padID>`; o cookie de *session* autoriza o acesso.

Operações de leitura/manutenção incluem `listPads` (por grupo), `getText`/`getHTML`, `getRevisionsCount`, `listAuthorsOfPad` e `getReadOnlyID` (Etherpad Foundation, 2018).

4.4.4 Encerramento da atividade e exportação

Ao final do período de edição:

1. **Congelamento**: tornar o *pad* somente leitura (duas alternativas):
   - Converter para *readonly* com `getReadOnlyID` e publicar o link apenas de leitura aos alunos; ou

- Desabilitar edição no pad original com `setPublicStatus(false)` e, se necessário, aplicar `setPassword` apenas para docentes.

2. **Exportação**: arquivar o produto em múltiplos formatos (endereços REST nativos como `/p/<padID>/export/pdf`, `.../txt`, `.../html`). Registrar também metadados (revisões e autores) via `getRevisionsCount` e `listAuthorsOfPad`.

3. **Encerramento de sessões**: remover/expirar *sessions* dos alunos (`deleteSession` ou aguardar *validUntil*).

4. **Conservação/limpeza**: opcionalmente, transferir o conteúdo final para repositório institucional (ex.: Git/arquivos da disciplina) e aplicar política de retenção (`deletePad` somente quando previsto pela governança de dados).

4.4.5 Considerações de segurança e boas práticas

- **Isolamento por grupo:** somente *pads* de *group* são acessíveis a quem possui *session* válida para esse *group*.
- **Mínimo de privilégios:** backend do LARA detém a *API key*; o navegador do aluno nunca expõe credenciais de API.
- **Registro e auditoria:** o histórico (changesets) e o chat, quando pertinente, subsidiam avaliação formativa; aplicar política de retenção compatível com a LGPD institucional.
- **Boas práticas de produção:** publicar o Etherpad atrás de *Ingress* com TLS; restringir *CORS*/*iframe* ao domínio do LARA; armazenar a *API key* como *Secret* (Kubernetes).

Em síntese, o plano estabelece um ciclo: (*i*) autenticar no LARA; (*ii*) provisionar *group/author/session/pad* via API; (*iii*) incorporar o *pad* no fluxo da atividade; (*iv*) congelar e exportar ao término; (*v*) assegurar segurança e boas práticas de produção. As operações e estruturas aqui descritas aderem às interfaces oficiais do Etherpad e ao desenho pedagógico do LARA, garantindo integração controlada e auditável.

4.5 Persistência de dados

Para armazenar de forma confiável o conteúdo dos pads e todo o seu histórico de modificações, o Etherpad emprega uma estratégia de persistência flexível, baseada em bancos de dados, mediada por uma biblioteca chamada UeberDB[7]. Diferentemente

[7] UeberDB2: Abstract your databases. Disponível em: <https://github.com/ether/ueberDB>. Acesso em: 06 out. 2025.

de aplicações tradicionais que utilizam um esquema relacional bem definido, o Etherpad adota um modelo orientado a pares chave-valor para guardar seus dados. Essa abordagem se alinha com a natureza do problema: documentos de texto com evoluções contínuas e incrementais são mais naturalmente representados como entradas de log ou estruturas não normalizadas, onde cada pad pode ter um conjunto variável de revisões, autores, mensagens de chat etc.

4.5.1 DirtyDB e UeberDB: opções de armazenamento

Por padrão, em uma instalação simples, o Etherpad vem configurado com o DirtyDB, um banco de dados local baseado em arquivo plano (um arquivo JSON — geralmente salvo como dirty.db)[8]. O DirtyDB é útil para fins de teste e desenvolvimento por demandar zero configuração – ele grava as entradas no próprio servidor de aplicação. Contudo, não é indicado para uso em produção, pois não é robusto contra falhas e concorrência pesada.

Para ambientes de produção, o Etherpad utiliza a camada do UeberDB, que funciona como um adaptador universal de banco de dados. O UeberDB abstrai o acesso de dados transformando qualquer SGBD subjacente em um simples repositório chave-valor. Com ele, é possível configurar o Etherpad para usar uma variedade de SGDBs conforme a preferência do administrador: *MySQL/MariaDB*, *PostgreSQL*, *SQLite*, *MongoDB*, *Redis*, dentre outros (há suporte incluso para mais de uma dezena de opções). Essa flexibilidade é valiosa para integrar o Etherpad à infraestrutura já existente de uma instituição. No caso deste trabalho, optou-se pelo PostgreSQL como base de dados para o Etherpad, aproveitando a confiabilidade e desempenho desse sistema gerenciador relacional.

O UeberDB, além de portabilizar o código para diversos bancos, implementa internamente mecanismos de cache e buffer de escrita que otimizam o desempenho do Etherpad. Em operação normal, as alterações em um pad são mantidas primeiramente em memória e gravadas periodicamente no banco em lote, ao invés de realizar escritas a cada digitação. Conforme Hellerstein, Stonebraker e Hamilton (2007), esse buffer de escritas reduz a sobrecarga transacional e melhora a escalabilidade, especialmente sob alta taxa de edição. Por outro lado, a consistência é mantida pelo fato de o servidor Etherpad centralizar as operações em tempo real e serializá-las no log de mudanças do pad. Ou seja, mesmo que múltiplos servidores de banco possam estar atendendo (por exemplo, em um cluster de banco), a lógica do Etherpad garante ordem aos eventos antes de enviá-los ao armazenamento.

Em resumo, DirtyDB oferece praticidade imediata para uso local, enquanto o UeberDB habilita o Etherpad a "falar" com diversos bancos de dados de forma uniforme. A escolha por PostgreSQL, por exemplo, traz benefícios como persistência confiável e

[8] node-dirty: GitHub repository. Disponível em: <https://github.com/felixge/node-dirty>. Acesso em: 06 out. 2025.

possibilidade de uso de ferramentas SQL para consultar os dados se necessário (embora, conforme discutiremos, os dados estejam em formato de chave-valor, podendo exigir manipulações específicas para interpretá-los externamente).

4.5.2 Estrutura dos dados persistidos

A maneira como as informações de um pad são armazenadas no banco através do UeberDB é fundamental para entender tanto a capacidade de recuperação do histórico quanto as limitações para análises diretas. Em linhas gerais, o Etherpad utiliza um espaço de chave-valor único (como uma grande tabela com duas colunas: `key`, `value`) onde todas as entradas relacionadas a pads, autores, sessões, etc., são diferenciadas pelo prefixo da chave. Conforme Etherpad Foundation (2018), algumas das principais chaves utilizadas pelo Etherpad incluem:

- **Pad content:** Cada pad ativo possui uma chave principal no formato pad:<padID> cujo valor é um objeto (ou texto) representando o conteúdo atual do documento naquele pad. Esse valor é atualizado conforme novas edições são aplicadas e reflete sempre a versão mais recente do texto consolidado .
- **Histórico de revisões:** O histórico detalhado de mudanças de um pad é armazenado como uma sequência de changesets. Cada changeset registra as diferenças introduzidas em uma determinada revisão. As chaves seguem o formato pad:<padID>:revs:<N> para a revisão de número N. O valor associado costuma incluir metadados como a composição do changeset (inserções/remoções), o autor da mudança e um carimbo de tempo. A revisão 0 geralmente é o estado inicial do pad (possivelmente vazio ou com texto importado).
- **Autores e atribuições:** Os usuários (autores) que editam pads podem ser registrados em chaves como globalAuthor:<authorID>, contendo informações como nome exibido e pad(s) que estão associados a ele. Além disso, as ligações entre pads e autores em cada revisão podem aparecer em chaves do tipo pad:<padID>:authors ou dentro dos próprios objetos de revisão.
- **Chat:** Mensagens de chat de um pad são persistidas separadamente, em chaves como pad:<padID>:chat:<msgID>, contendo conteúdo da mensagem, autor e timestamp.
- **Grupos e sessões:** Quando o Etherpad é usado em modo integrado a um sistema de autenticação (via API), existem chaves para grupos (group:<groupID>) que podem agrupar vários pads, e chaves de sessão (session:<sessionID>) que ligam usuários autenticados a pads/grupos por um intervalo de tempo.

Essa estrutura distribuída em múltiplas chaves reflete a filosofia de log de eventos. Para reconstruir o conteúdo de um pad em determinado momento, o Etherpad pode aplicar

a sequência de changesets até aquela revisão, partindo do estado vazio ou inicial. Entretanto, para otimização, o Etherpad também mantém snapshots periódicos do conteúdo completo em certas revisões (por exemplo, a cada 100 revisões) como forma de checkpoint, evitando ter que reprocessar todas as mudanças desde o início para apresentar o histórico no time slider. Com isso, a chave principal do pad (pad:<padID>) pode ser vista como o snapshot mais recente, enquanto as revisões incrementais detalham as diferenças.

Um ponto importante é que, pelo fato dos dados estarem normalizados em forma de chave-valor textuais, conforme a figura 9, realizar consultas complexas diretamente no banco (como buscar uma palavra em todos os pads, ou gerar estatísticas de edição por usuário) torna-se não trivial. Ferramentas externas ou plugins do Etherpad são utilizados para extrair esses tipos de informações. Por exemplo, a busca textual global requer iterar por todas as chaves de pad e inspecionar o conteúdo, já que não há tabelas relacionais clássicas para indexar palavras. Existem plugins de busca e administração que percorrem os pads da instância (por exemplo, ep search, ep adminpads2 e "Historical Search") . Em 2012–2013, a Mozilla colaborou ativamente com a comunidade Etherpad — por exemplo, sediando um meetup norte-americano e realizando uma auditoria de segurança que culminou no release 1.2.9 —, mas a funcionalidade de busca full-text não foi incorporada ao núcleo do projeto[9]. Assim, na versão padrão, o projeto prioriza a simplicidade do modelo de armazenamento e delega funcionalidades avançadas (como busca global) a plugins e sistemas externos.


```
{
  "atext":{
    "text":"Hoje vamos trabalhar um problema simples: dado um número, queremos verificar se ele é par ou :
    "attribs":"*1+4a*0|7+36*0+8|1+1"
  },
  "pool":{
    "numToAttrib":{
      "0":[
        "author",
        "a.xAgFUjPkZr0BpK9W"
      ],
      "1":[
        "author",
        "a.twifitkaYSu3fnYg"
      ]
    },
    "nextNum":2
  },
  "head":5,
  "chatHead":2,
  "publicStatus":false,
  "savedRevisions":[

  ]
}
```


Figura 9 – Consulta SQL feita no banco de dados da aplicação referente à PAD da figura 1.

[9] Etherpad Team. Releasing 1.2.9. Disponível em: <https://blog.etherpad.org/2013/03/17/releasing-1-2-9>. Acesso em: 06 out. 2025.

No contexto da implementação do Etherpad no LARA, conhecer a estrutura do banco de dados auxilia na integração dos sistemas. Por exemplo, para sincronizar a criação de uma nova atividade no AVA com a criação de um pad Etherpad correspondente, utilizamos a API (que internamente cria as devidas entradas no banco do Etherpad). Além disso, caso seja necessário correlacionar dados do Etherpad (por exemplo, tempo de edição, quantidade de contribuições de cada aluno) com dados do Moodle ou do laboratório remoto, deve-se realizar extrações via API ou consultas específicas combinando os bancos. A tabela 2 sugere parcialmente como os dados do Etherpad se relacionam conceitualmente, embora no nível físico estejam todos dentro do mesmo repositório chave-valor.

Tabela 2 – Estrutura de dados do Etherpad. Todos os itens são persistidos como pares chave–valor pela camada UeberDB.

Chave (padrão)	**Entidade**	**Campos/valor principais**	**Observações/Relacionamentos**
groups	Índice global de grupos	Objeto JSON com pares groupID: 1	Lista de todos os grupos existentes.
pad:PADID	Pad (conteúdo atual)	atext; pool; head; chatHead; public; passwordHash	Contém as informações consolidadas do pad; head aponta a última revisão; chatHead aponta a última mensagem do chat.
pad:PADID:revs:REVNUM	Revisão do pad (changeset)	meta.author; meta.timestamp; changeset	Log de mudanças por revisão. Permite reconstruir o estado aplicando os changesets na ordem.
pad:PADID:chat:CHATNUM	Entrada de chat do pad	text; userId; time	Mensagens de chat associadas ao pad, numeradas sequencialmente.
pad2readonly:PADID	Mapeamento pad → readonlyID	readonlyID (valor simples)	Converte o identificador do pad para um identificador somente leitura.
readonly2pad:READONLYID	Mapeamento readonlyID → pad	PADID (valor simples)	Operação inversa de pad2readonly.
token2author:TOKENID	Mapeamento de token de cliente → autor	AUTHORID (valor simples)	Associa um token a um autor global.
globalAuthor:AUTHORID	Autor (perfil global)	name; colorID	Nome e cor exibidos para esse autor.
mapper2group:MAPPER	Mapeamento externo → groupID	GROUPID (valor simples)	Conecta um identificador externo (aplicação) a um grupo interno.

Continuação na próxima página

Chave (padrão)	Entidade	Campos/valor principais	Observações/Relacionamentos
mapper2author:MAPPER	Mapeamento externo → authorID	AUTHORID (valor simples)	Conecta um identificador externo (aplicação) a um autor interno.
group:GROUPID	Grupo de pads	pads (objeto com nomes dos pads; valores 1)	Agrupa vários pads sob o mesmo groupID.
session:SESSIONID	Sessão autor–grupo	groupID; authorID; validUntil	Vínculo temporal entre um autor e um grupo.
author2sessions:AUTHORID	Índice de sessões por autor	sessionsIDs (objeto com ids; valores 1)	Lista de sessões do autor.
group2sessions:GROUPID	Índice de sessões por grupo	sessionsIDs (objeto com ids; valores 1)	Lista de sessões do grupo.

Em suma, o Etherpad sacrifica a estruturação rígida do modelo relacional em favor de um modelo flexível orientado a eventos, otimizado para a rápida persistência e recuperação sequencial do histórico de edições colaborativas. Essa escolha de design se mostra adequada para a finalidade do software, embora imponha desafios adicionais quando integrado com outras aplicações que esperam dados relacionais. No projeto presente, contornamos isso usando as interfaces oficiais do Etherpad em vez de acessar diretamente seu banco, garantindo assim consistência e simplicidade na implementação.

4.6 Plugins

Conforme mencionado, a extensibilidade por meio de plugins foi uma característica relevante tanto do Etherpad quanto de outras ferramentas utilizadas. De maneira geral, um plugin (ou plug-in, do termo em inglês) é um componente de software que adiciona funcionalidades a uma aplicação principal, sem necessidade de modificar o código-fonte dessa aplicação (HAN, 2011). Os plugins seguem um contrato ou API fornecida pelo sistema hospedeiro, através do qual podem interagir com eventos, modificar comportamentos ou fornecer novas interfaces no programa principal.

No Etherpad, a arquitetura de plugins permite que desenvolvedores criem pacotes independentes (publicados inclusive via npm) que, quando instalados, ampliam as capacidades do editor. Por exemplo, o ep_spellcheck adiciona verificação ortográfica; o ep_table_of_contents cria um sumário automático em tempo real a partir dos cabeçalhos do texto; e o ep_images_extended permite inserir imagens “inline”, além de flutuá-las e redimensioná-las. Esses são apenas alguns dentre muitos. Durante a implementação no LARA, avaliamos a utilização de certos plugins para melhor adaptar o Etherpad ao contexto educacional. Embora não tenhamos desenvolvido um plugin próprio do zero, a possibilidade existe caso requisitos específicos surjam, graças ao arcabouço fornecido.

4.6.1 Plugins utilizados

A seguir, descrevem-se os plugins selecionados para a instância do Etherpad neste projeto, justificando sua adoção no contexto do LARA e indicando a funcionalidade principal de cada um.

- **`ep_author_hover`:** adiciona nas configurações do pad a opção “Show Author on Hover”, exibindo o nome do autor ao passar o cursor sobre trechos do texto. Esse recurso reforça a visibilidade da

autoria durante atividades colaborativas, contribuindo para avaliação formativa e atribuição de crédito individual.

- **`ep_webrtc`:** provê comunicação de áudio e vídeo (e compartilhamento de tela) baseada em WebRTC entre os usuários de um mesmo pad. Os fluxos são *peer-to-peer*, o que favorece grupos pequenos (por exemplo, duplas ou quartetos) em atividades síncronas, aproximando o editor colaborativo de um estúdio de trabalho remoto.
- **`ep_align`:** permite alinhar parágrafos (esquerda, centro, direita e justificado). Em atividades de produção textual (relatos, relatórios ou documentação de código), a organização tipográfica melhora a legibilidade e a apresentação do material entregue.
- **`ep_comments_page`:** habilita comentários e anotações vinculados a trechos do texto, exibidos em barra lateral. Esse recurso é particularmente útil para *feedback* do docente e revisão por pares, sem alterar o corpo principal do documento.
- **`ep_embedded_hyperlinks2`:** facilita a inserção de hiperlinks "embutidos" no texto, permitindo referenciar documentação externa (ex.: tarefas do LARA, tutoriais, issues de repositórios) diretamente no pad.
- **`ep_font_color`:** acrescenta um seletor de cor para o texto. No ensino introdutório, cores podem ser usadas como estratégia de mediação (por exemplo, destacar trechos que exigem revisão ou diferenciar papéis durante a coedição).
- **`ep_headings2`:** adiciona suporte a cabeçalhos (H1, H2, ...), com compatibilidade de importação/exportação. Ajuda a estruturar documentos mais longos (relatórios, guias, roteiros de aula) e a criar sumários automáticos quando combinado com outros plugins.
- **`ep_markdown`:** oferece um modo de visualização em Markdown ("Show as Markdown"). Útil para exportar rascunhos técnicos ou preparar material que será versionado/compartilhado fora do Etherpad; note-se que o plugin não transforma a digitação em Markdown em formatação ao vivo — trata-se de uma *visualização*.

A seleção priorizou plugins mantidos pela comunidade do Etherpad e/ou pela Etherpad Foundation, com documentação pública e suporte nas versões atuais. A instalação e a gestão dos plugins podem ser feitas via interface administrativa (/admin/plugins) ou diretamente por configuração (por exemplo, via Docker build args ou settings.json), conforme a documentação oficial.

Em termos conceituais, a adoção de plugins está alinhada ao princípio de design de alta coesão e baixo acoplamento: funcionalidades extras podem ser adicionadas sem infligir risco ao núcleo estável da aplicação. Han (2011) ressalta que entender o mecanismo de plugins (carregamento dinâmico de módulos, gerenciamento de dependências e possíveis impactos em desempenho) é importante para prever como um sistema evoluirá. No nosso caso, confirmamos na prática que adicionar plugins ao Etherpad foi simples (bastou incluir seus nomes na configuração e reiniciar o serviço, já que o Etherpad carrega plugins listados no settings.json, ou podemos apenas acessar a rota /admin/plugins do software e adicionar o plugin desejado). No entanto, também observamos que é preciso cautela na escolha: alguns plugins de terceiros não estavam totalmente atualizados para a última versão do Etherpad, causando incompatibilidades. Assim, optamos por um conjunto mínimo e estável de plugins durante a implantação final, priorizando confiabilidade.

4.7 Arquitetura

A implantação do Etherpad no contexto do LARA foi realizada sobre uma arquitetura baseada em contêineres orquestrados pelo Kubernetes, integrando a aplicação Etherpad a um banco de dados PostgreSQL. Essa arquitetura segue boas práticas de separação de componentes: o Etherpad, sendo um serviço sem estado (*stateless*), executa independentemente do armazenamento de dados, enquanto o PostgreSQL (*stateful*) assume a responsabilidade pelo armazenamento persistente das informações dos pads. Na figura 10, é possível ver as diferenças entre esses tipos de serviços dentro de um ambiente cluster. Cada componente é definido e gerenciado por objetos nativos do Kubernetes, conforme descrito a seguir.

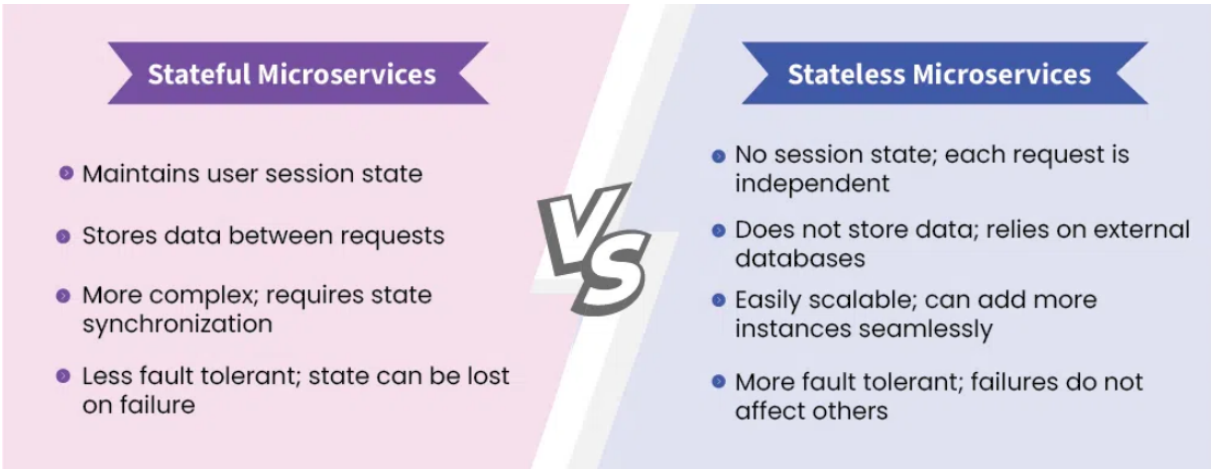

Figura 10 – Diferença entre serviços *stateless* e *stateful*. [10]

O Etherpad foi implantado como um pod gerenciado por um objeto Deployment. Um Deployment no Kubernetes gerencia um conjunto de réplicas de um contêiner, garantindo alta disponibilidade e atualizações controladas; conforme Camisso, Jetha e Juell (2020), esse tipo de controlador é indicado para aplicações que não mantêm estado na camada de aplicação. No caso do LARA, o Deployment assegura que haja sempre uma instância do Etherpad em execução, recriando o pod automaticamente em caso de falha. Optou-se por utilizar apenas 1 réplica do Etherpad nesta implantação, mas o uso de um Deployment permitiria escalar horizontalmente o serviço com facilidade caso a demanda de usuários aumente (bastando ajustar o número de réplicas).

O contêiner do Etherpad utiliza uma imagem Docker personalizada, `crisao/etherpad-lite:1.0`, criada especificamente para este projeto. Trata-se de uma imagem improvisada: prevê-se migrar futuramente para uma imagem oficial mantida pela equipe do LARA em um repositório próprio, garantindo maior confiabilidade a longo prazo. O pod do Etherpad expõe a porta 9001 (porta padrão do Etherpad) e obtém suas variáveis de configuração via objetos ConfigMap e Secret, conforme detalhado adiante.

Para o banco de dados PostgreSQL, que é um componente com estado (*stateful*), utilizou-se um objeto StatefulSet. O StatefulSet é projetado para gerenciar aplicações que necessitam de armazenamento persistente ou identidade de rede estável entre reinicializações (The Kubernetes Authors, 2025). Diferentemente de um Deployment, em que os pods são intercambiáveis, um StatefulSet garante que cada réplica tenha um identificador fixo e retenha a associação com seu volume de dados – em outras palavras, cada pod mantém sua identidade e estado próprios mesmo que seja reiniciado ou movido de nó.

Na arquitetura implementada configurou-se o StatefulSet do PostgreSQL com réplica única, resultando em um único pod de banco (denominado `postgres-db-0`). Esse pod utiliza a imagem oficial `postgres:14-alpine` e foi parametrizado com variáveis de ambiente que definem o banco de dados a ser usado (`etherpad_db`) e as credenciais de acesso: por exemplo, o usuário do banco (`etherpad_user`) e a

respectiva senha. Esta última não é exposta diretamente no manifesto; em vez disso, o StatefulSet referencia um objeto Secret que armazena a senha do banco, de modo que a variável de ambiente `POSTGRES_PASSWORD` seja populada de forma segura. Adicionalmente, o StatefulSet inclui um *VolumeClaimTemplate* que provisiona automaticamente um *PersistentVolumeClaim* (PVC) de 1Gi para o armazenamento dos dados do PostgreSQL. Esse volume persistente garante que os dados do banco sejam preservados mesmo que o contêiner seja reiniciado ou agendado em outro nó do cluster, permitindo inclusive a realização de backups do volume independentemente do ciclo de vida do pod do banco.

A comunicação e exposição dos serviços Etherpad e PostgreSQL dentro do cluster foram configuradas por meio de objetos Service. Para o Etherpad, foi criado um Service do tipo `NodePort`, que expõe a aplicação em uma porta fixa em cada nó do cluster. Assim, usuários externos podem acessar a interface web do Etherpad através do endereço IP de um dos nós (ou do IP do Minikube, no ambiente de desenvolvimento) e da porta NodePort designada, que encaminha as requisições para a porta 9001 do contêiner Etherpad. Essa escolha dispensa a necessidade de um balanceador de carga externo e foi suficiente para testes locais; em cenários produtivos, outras abordagens como *LoadBalancer* ou *Ingress* poderiam ser consideradas, mas o NodePort cumpriu o papel de fornecer acesso externo de forma simples.

Já o PostgreSQL foi exposto apenas internamente, por meio de um Service do tipo `ClusterIP` (padrão do Kubernetes). Esse Service fornece um endpoint estável dentro do cluster (no mesmo namespace) para que o Etherpad se conecte ao banco, mas não abre nenhuma porta externamente, mantendo o SGBD isolado de acessos fora do cluster. Além disso, definiu-se esse Service como *headless* (atributo `clusterIP: None`), habilitando a descoberta via DNS dos pods do StatefulSet. Na prática, como há apenas uma réplica do banco, o nome DNS `postgres-db.etherpad.svc.cluster.local` resolve diretamente para o IP do pod `postgres-db-0`, permitindo que o Etherpad o acesse pelo hostname configurado. Essa configuração reforça a segurança da solução, pois o banco de dados não fica acessível fora do cluster, ao mesmo tempo em que facilita a referência e comunicação com o serviço de banco internamente.

A configuração da aplicação Etherpad em si (credenciais de banco, parâmetros de operação do servidor etc.) foi externalizada em um objeto ConfigMap. O ConfigMap denominado `etherpad-config` contém pares chave-valor correspondentes às variáveis utilizadas pelo Etherpad. Incluem-se chaves como `DB_TYPE`, `DB_HOST`, `DB_PORT`, `DB_USER` e `DB_NAME`, especificando o banco de dados PostgreSQL a ser utilizado, e também opções da aplicação como `TITLE` (título da instância Etherpad), `DEFAULT_PAD_TEXT` (texto inicial padrão dos pads), `TRUST_PROXY` (habilitação de *proxy reverso*) e `ADMIN_PASSWORD` (senha do usuário administrador do Etherpad). Esses valores são carregados no contêiner do Etherpad através da diretiva `envFrom` no Deployment (que referencia o ConfigMap), complementada por variáveis de ambiente individuais que obtêm as credenciais sensíveis a partir de Secrets (por exemplo, a variável `DB_PASS` do Etherpad é suprida pelo Secret do PostgreSQL). Com essa abordagem, segue-se a boa prática de separar as configurações da imagem da aplicação, possibilitando ajustes posteriores sem necessidade de reconstruir ou alterar o contêiner.

Para armazenar informações confidenciais, como senhas, a solução fez uso de objetos Secret. O Kubernetes Secret chamado `postgres-secret` guarda, de forma codificada (Base64), a senha do banco de dados PostgreSQL utilizada pelo Etherpad (chave `POSTGRES_PASSWORD`). De maneira similar, planeja-se mover a senha de administrador do Etherpad para um Secret dedicado – no ConfigMap há inclusive um comentário indicativo dessa melhoria –, embora na implementação atual ela permaneça no ConfigMap para simplificar a implantação inicial. O uso de Secrets evita expor credenciais em texto plano nos manifestos, integrando-as de forma segura aos pods no momento da execução.

A Figura 11 apresenta a arquitetura lógica preparada para o LARA: usuários acessam o Etherpad por meio de um *Service* do tipo `NodePort`; a aplicação é executada como Deployment (gerenciador de pods); a persistência ocorre no PostgreSQL exposto internamente por *Service* `ClusterIP/headless` e acoplado a um PVC (que referencia um PV *StorageClass*). Também se evidencia a injeção de configuração

por ConfigMap (variáveis não sensíveis) e de credenciais por Secret (valores codificados em Base64).

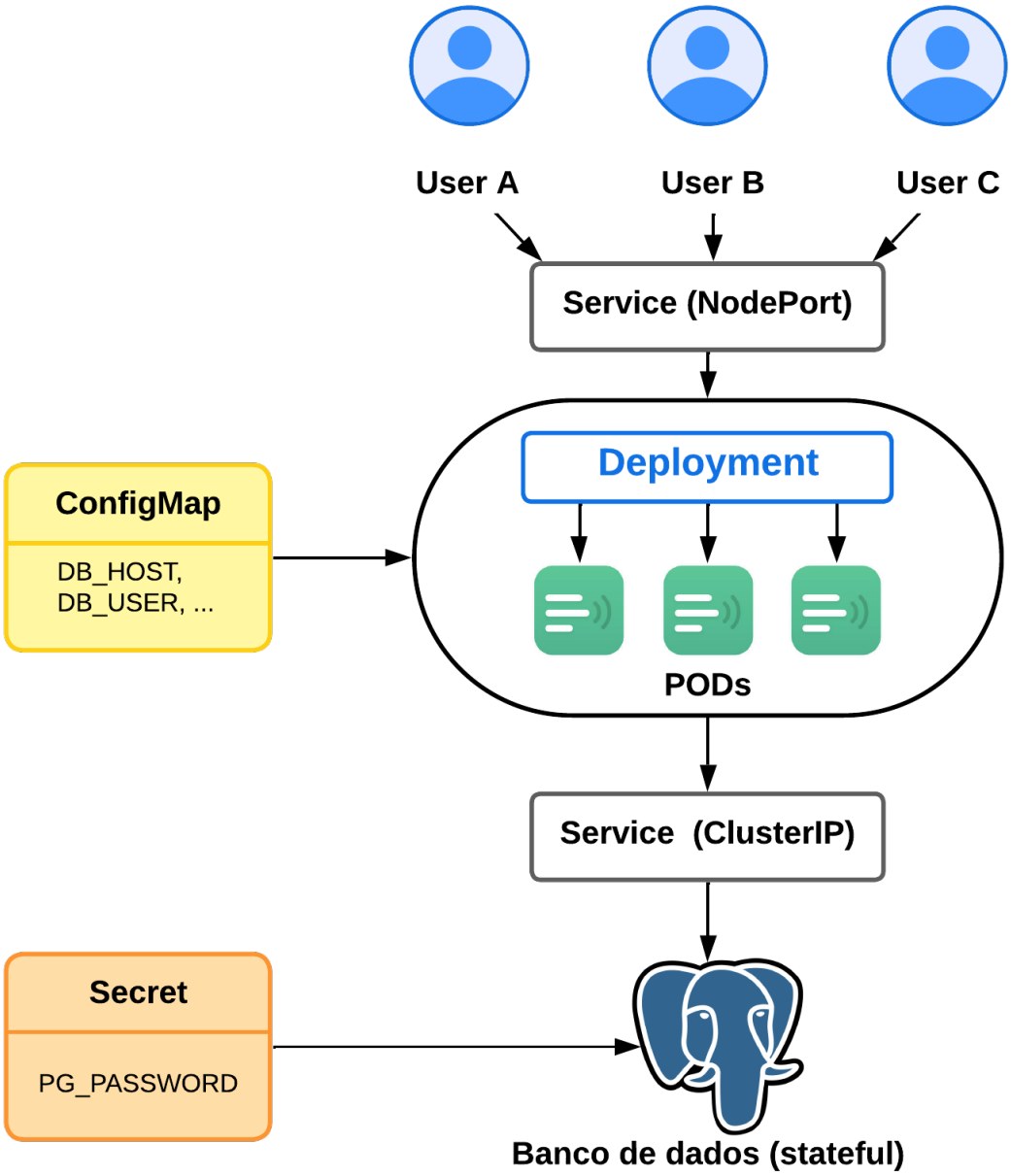

Figura 11 – Arquitetura lógica.

Já a figura 12 sintetiza os recursos efetivamente criados no namespace etherpad. Observa-se a presença do deployment.apps/etherpad-deployment (com 1 réplica e pod/etherpad-deployment-... em Running), do statefulset.apps/postgres-db (com pod/postgres-db-0), bem como dos serviços etherpad-svc do tipo NodePort (mapeando 9001:31110/TCP) e postgres-db do tipo ClusterIP/headless (porta 5432/TCP). Além destes, também há o configmap/etherpad-config, responsável por armazenar as variáveis não sensíveis que irão alimentar a aplicação, e o secret/postgres-secret que irá alimentar o banco de dados com as credenciais sensíveis que não podem ser armazenadas no configmap.

Esses elementos estão alinhados à arquitetura lógica da Figura 11 e corroboram a separação entre componente *stateless* (aplicação) e *stateful* (dados).

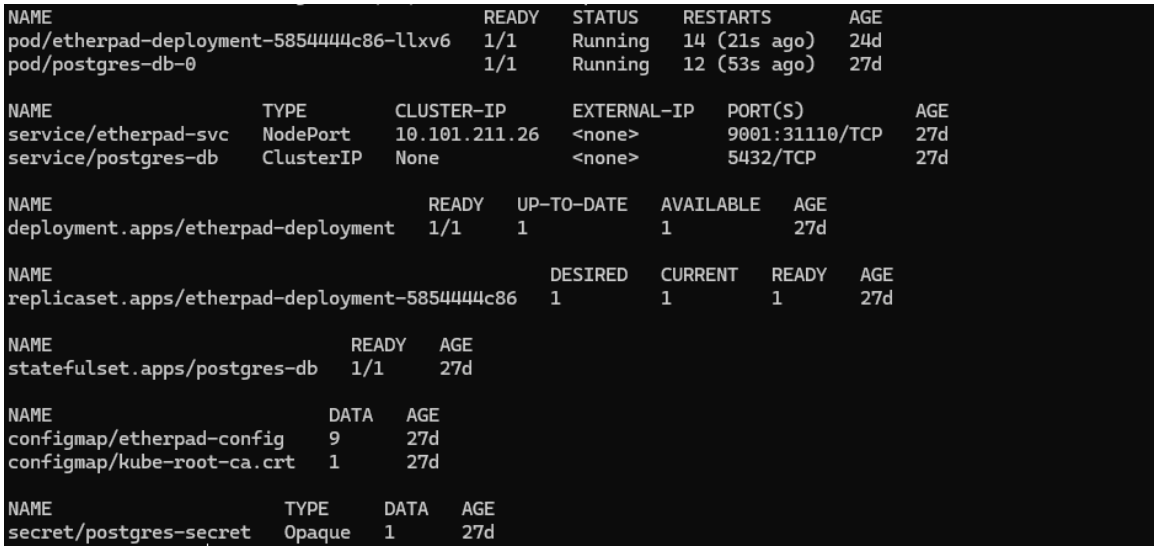

Figura 12 – Workloads aplicados no kubernetes através de manifestos yamls.

Adicionalmente, todos os manifestos Kubernetes definidos para essa implantação serão disponibilizados no repositório oficial do LARA. Isso inclui os arquivos YAML do Deployment do Etherpad, do StatefulSet do PostgreSQL, dos Services, do ConfigMap de configuração e do Secret – este último com os campos de credenciais deliberadamente vazios no repositório, por motivo de segurança. Em um ambiente de produção, valores reais para essas credenciais devem ser fornecidos de forma segura (por exemplo, via variáveis de ambiente no pipeline de implantação ou por ferramentas de gerenciamento de segredos), mantendo as senhas protegidas fora do controle de versão.

Finalmente, a separação entre os componentes sem estado e com estado traz diversos benefícios em termos de reinicialização, backup e escalabilidade. O Etherpad, por ser *stateless*, pode ser reiniciado ou replicado à vontade, sem perda de dados — caso o pod da aplicação caia, o Deployment irá recriá-lo automaticamente; se houver aumento de carga de usuários, pode-se elevar a contagem de réplicas do Etherpad sem grandes complicações. Por sua vez, o PostgreSQL concentra todo o estado da aplicação e permanece protegido por armazenamento persistente, de modo que reinicializações do seu pod não acarretam perda de informações. Além disso, os procedimentos de backup podem se focar unicamente no banco de dados (por exemplo, realizando *dumps* SQL regulares ou instantâneos do volume de dados), simplificando a manutenção da persistência. A escalabilidade horizontal do banco de dados, se necessária, requer abordagens adicionais (p.ex., replicação ou *sharding*); porém, graças à arquitetura adotada, a escalabilidade inicial do sistema pode ser obtida na camada de aplicação, simplesmente adicionando instâncias do Etherpad que compartilhem o mesmo banco. Essa divisão de responsabilidades está alinhada às práticas recomendadas em arquiteturas cloud-nativas e de microsserviços, contribuindo para a robustez e facilitando a manutenção da solução implantada no LARA.

4.7.1 Síntese de atendimento aos requisitos

Para concluir, apresenta-se uma matriz de rastreabilidade que relaciona os requisitos elicitados (seção 4.2) com os elementos da solução implementada ao longo do Capítulo 4. Os quadros a seguir indicam, para cada requisito, os artefatos/tecnologias que o suportam, onde isso aparece no texto (evidência) e o status nesta versão do projeto (*Atendido*, *Parcial* ou *Preparado* para a futura integração ao LARA).

Tabela 3 – Quadro — Rastreabilidade dos *requisitos funcionais* na solução

Requisito funcional	**Implementação / artefatos**	**Evidência (seções/figuras)**	**Status**
Edição colaborativa em tempo real	Etherpad, destaque de autor, sincronização via WebSocket/HTTP.	Seç. 4.1; Fig. 1 e 4	Atendido
Chat integrado ao pad	Chat nativo por pad (janela lateral).	Seç. 4.1; Fig. 1	Atendido
Gerenciamento de pads por grupos/turmas	Uso da API HTTP do Etherpad (grupos, sessões, pads); mapeamentos *pad/group/session*.	Seç. 4.1 (API); Seç. 4.4.2 (estrutura de dados)	Preparado
Histórico de versões e *Time Slider* para monitoramento docente	Registro de *changesets*, reprodução temporal (*Time Slider*).	Seç. 4.1; Fig. 4	Atendido
Exportação do conteúdo (PDF, HTML, ODT, DOCX, TXT)	Mecanismos nativos de import/export do Etherpad.	Seç. 4.1; Fig. 5	Atendido
Facilidades para o docente criar/gerir atividades colaborativas	Página de Admin do Etherpad (/admin), instalação de plugins; criação programática via API.	Seç. 4.1 (API, plugins); Seç. 4.5	Parcial

Tabela 4 – Quadro — Rastreabilidade dos *requisitos não funcionais* na solução

Requisito não funcional	**Implementação / artefatos**	**Evidência (seções/figuras)**	**Status**
Desempenho e escalabilidade	Implantação em contêiner; *Deployment* Kubernetes (réplicas escaláveis); *Service* para balancear tráfego; possibilidade de *HPA*.	Seç. 4.3.1–4.3.2 e 4.6; Fig. 8, 11 e 12	Preparado
Usabilidade e acessibilidade	UI simples do Etherpad (web, sem instalação); identificação por cores; plugins opcionais (ex.: headings, markdown).	Seç. 4.1; Seç. 4.5	Atendido
Manutenibilidade e implantação reprodutível	Imagem Docker dedicada; configurações externas via *ConfigMap*; credenciais via *Secret*; manifestos versionados.	Seç. 4.3.1 e 4.6; Fig. 7, 11 e 12	Atendido
Persistência e durabilidade dos dados	PostgreSQL sob *StatefulSet* com PVC/PV; camada UeberDB; *Service* interno (ClusterIP).	Seç. 4.3.2 e 4.4; Seç. 4.6; Fig. 8, 11 e 12	Atendido
Extensibilidade e adaptabilidade	Arquitetura modular de plugins do Etherpad; seleção de plugins educacionais.	Seç. 4.1 e 4.5	Atendido
Compatibilidade e integração transparente ao AVA	Acesso via web/iframe; uso de API para grupos/sessões; preparação de DNS/Service e variáveis de ambiente.	Seç. 4.1 (API), 4.3–4.6; Fig. 8	Preparado
Segurança básica e confidencialidade	Isolamento por contêiner; banco não exposto externamente (Service ClusterIP); credenciais em *Secret*.	Seç. 4.6; Fig. 11 e 12	Parcial

5 Conclusão

Este trabalho partiu do desafio de preparar um editor colaborativo em tempo real para o contexto de um AVA acadêmico, alinhando princípios de CSCL às práticas contemporâneas de engenharia de software. A solução adotada tomou o Etherpad como núcleo de coedição e estruturou sua implantação em contêineres, com orquestração via Kubernetes e persistência em PostgreSQL, visando reprodutibilidade, robustez e escalabilidade.

Todas essas fundamentações teóricas e técnicas embasam as decisões tomadas na implementação e fornecem contexto para compreender os resultados e desafios discutidos nos capítulos deste trabalho. Em essência, integrar o Etherpad no LARA não se resume a juntar duas ferramentas, mas sim a conciliar paradigmas de aprendizagem colaborativa com soluções de engenharia de software modernas, o que exigiu domínio dos conceitos aqui expostos.

Em síntese, os objetivos (a)–(c) foram cumpridos: (a) levantamento e análise de requisitos, com elicitação, organização e rastreabilidade que orientaram as decisões; (b) definição e aplicação de uma arquitetura Kubernetes, com separação de componentes stateless/stateful e uso de Deployment (Etherpad), StatefulSet (PostgreSQL) e Services com possibilidade de escalonamento; (c) criação de um Dockerfile/imagem estável e reprodutível, parametrizada via ConfigMap/Secret, favorecendo reprodutibilidade e manutenibilidade; já o objetivo (d) — integração do Etherpad ao LARA com persistência e coedição em tempo real — permanece pendente: a instância do Etherpad e os mecanismos de integração (API de grupos/sessões/pads), gestão por turmas e exportação estão prontos, restando a incorporação plena ao front-end do LARA.

Como próximos passos operacionais, prevê-se (i) a integração de interface no LARA, incorporando os pads nas páginas de atividade, vinculando grupos/turmas aos respectivos pads e controlando permissões de leitura/edição com exposição do histórico para avaliação formativa; (ii) autenticação e autorização federadas via SSO do LARA, com criação e gestão transparente de authors e sessions no Etherpad; (iii) provisionamento e ciclo de vida automatizados, com criação de pads na abertura da atividade, políticas de retenção, rotinas de backup e cleanup no encerramento; e (iv) operacionalização para produção com Ingress (TLS e DNS institucional), monitoramento e logs instrumentados, possibilidade de HPA, pipeline CI/CD e evolução da persistência com backup/restore e, quando aplicável, alta disponibilidade do banco.

Em síntese, o presente projeto entrega uma base técnica sólida e replicável para a adoção de coedição no LARA, articulando requisitos pedagógicos e de infraestrutura. Ao documentar a preparação da integração e disponibilizar artefatos de implantação, o trabalho reduz o esforço de adoção futura e abre caminho para experimentação em sala de aula e pesquisas subsequentes em CSCL com suporte de ferramentas abertas e escaláveis.

Referências

ANDERSON, D. J. *Kanban: Successful Evolutionary Change for Your Technology Business*. [S.l.]: Blue Hole Press, 2010.

BRAVO, C.; DUQUE, R.; GALLARDO, J. A groupware system to support collaborative programming: Design and experiences. *Journal of Systems and Software*, v. 86, n. 7, p. 1759–1771, 2013.

BURNS, B. et al. *Kubernetes: Up and Running: Dive into the Future of Infrastructure*. 3. ed. Sebastopol, CA: O'Reilly Media, 2022. Acesso em: 22 ago. 2025. Disponível em: <https://www.oreilly.com/library/view/kubernetes-up-and/9781098110192/>.

CAMISSO, J.; JETHA, H.; JUELL, K. *Kubernetes for Full-Stack Developers*. New York, NY: DigitalOcean, 2020. Acesso em: 11 set. 2025. Disponível em: <https://assets.digitalocean.com/books/kubernetes-for-full-stack-developers.pdf>.

CHACON, S.; STRAUB, B. *Pro Git*. 2. ed. New York: Apress, 2014.

COLLINS-SUSSMAN, B.; FITZPATRICK, B. W.; PILATO, C. M. *Version Control with Subversion*. 2. ed. Sebastopol: O'Reilly Media, 2004.

DANG, Q.-V.; IGNAT, C.-L. Performance of real-time collaborative editors at large scale: User perspective. In: *IFIP International Conference on Open and Big Data*. IFIP, 2016. p. 248–259. Acesso em: 17 ago. 2025. Disponível em: <https://www.researchgate.net/publication/304457145_Performance_of_real-time_collaborative_editors_at_large_scale_User_perspective>.

DILLENBOURG, P. What do you mean by collaborative learning? In: DILLENBOURG, P. (Ed.). *Collaborative learning: Cognitive and computational approaches*. Oxford: Elsevier, 1999. p. 1–19.

Docker Inc. *Docker Documentation*. 2025. <https://docs.docker.com/>. Acesso em: 11 fev. 2025.

ELLIS, C. A.; GIBBS, S. J. Concurrency control in groupware systems. In: *Proceedings of the 1989 ACM SIGMOD Conference*. ACM, 1989. p. 399–407. Disponível em: <https://dl.acm.org/doi/10.1145/67544.66963>.

Etherpad Foundation. *EasySync Technical Manual*. 2018. <https://etherpad.org/doc/v2.4.2/#_technical_notes>. Acesso em: 17 ago. 2025.

Etherpad Foundation. *Etherpad HTTP API Documentation*. 2024. <https://etherpad.org/doc/v2.2.7/#_http_api>. Acesso em: 11 fev. 2025.

GOMES, A. S.; PIMENTEL, E. P. Ambientes virtuais de aprendizagem para uma educação mediada por tecnologias digitais. In: PIMENTEL, M.; SAMPAIO, F. F.; SANTOS, E. (Ed.). *Informática na Educação: ambientes de aprendizagem, objetos de aprendizagem e empreendedorismo*. Porto Alegre, RS, Brazil: Sociedade Brasileira de Computação, 2021, (Série Informática na Educação, v. 5). Disponível em: <https://ceie.sbc.org.br/livrodidatico/ava>.

GREGOR, S. The nature of theory in information systems. *MIS Quarterly*, v. 30, n. 3, p. 611–642, 2006.

HAN, J. Understanding software plugins. *IEEE Software*, v. 28, n. 6, p. 14–16, 2011.

HELLERSTEIN, J. M.; STONEBRAKER, M.; HAMILTON, J. Architecture of a database system. *Foundations and Trends in Databases*, Now Publishers, v. 1, n. 2, p. 141–259, 2007. Disponível em: <https://db.cs.berkeley.edu/papers/fntdb07-architecture.pdf>.

HOCHMüLLER, E.; MAURER, H. Using version control systems in collaborative learning environments: An educational perspective. *Educational Technology & Society*, v. 21, n. 4, p. 23–34, 2018.

HUANG, J.; LIU, Q.; ZHANG, Y. Development and application of a collaborative virtual learning environment based on version control. *Computers & Education*, Elsevier, v. 50, n. 1, p. 1–22, 2008.

KLEPPMANN, M. et al. *Local-First Software: You Own Your Data, in spite of the Cloud*. 2019. <https://martin.kleppmann.com/papers/local-first.pdf>. Acesso em: 17 ago. 2025.

KOSCHMANN, T. (Ed.). *CSCL: Theory and practice of an emerging paradigm*. Mahwah, NJ: Lawrence Erlbaum Associates, 1996.

KUBERNETES. *Welcome! (minikube - Kubernetes)*. 2025. <https://minikube.sigs.k8s.io/>. Acesso em: 22 ago. 2025.

LEON, F. T. et al. A container orchestration development that optimizes the etherpad collaborative editing tool through a novel management system. *Electronics*, v. 9, p. 828, 05 2020.

LOELIGER, J.; MCCULLOUGH, M. *Version Control with Git: Powerful tools and techniques for collaborative software development*. 2nd. ed. Sebastopol, CA: O'Reilly Media, 2012.

LOPES, M. S. dos S. *Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica*. Tese (Doctoral Thesis) — Universidade Federal da Bahia, Salvador, BA, 2017.

MCKELVEY, K. R. et al. *Upwelling: Combinando colaboração em tempo real com controle de versão para escritores*. 2023. <https://www.inkandswitch.com/upwelling/>. Acesso em: 17 ago. 2025.

O'SULLIVAN, B. *Mercurial: The Definitive Guide*. 1. ed. Sebastopol: O'Reilly Media, 2009.

PEREIRA, N. B. de S. *Estudo do estado da arte acerca da eficácia da acessibilidade dentro dos Ambientes Virtuais de Aprendizagem*. 2023. Trabalho de Conclusão de Curso (Graduação em Pedagogia) – Universidade de Brasília.

PIURA, P. et al. Pesquisa aplicada tecnológica: um estudo sobre a inovação em ambientes educacionais. *Revista Espacios*, v. 35, n. 9, 2014. Acesso em: 27 jun. 2024. Disponível em: <https://www.revistaespacios.com/a14v35n09/14350913.html>.

PREGUIÇA, N.; BAQUERO, C.; SHAPIRO, M. *Conflict-free Replicated Data Types (CRDTs): An Overview*. 2018. <https://pages.lip6.fr/Marc.Shapiro/papers/CRDTs-Springer2018-authorversion.pdf>. Acesso em: 17 ago. 2025.

RINBERG, A. et al. Dson: Json crdt using delta-mutations for document stores. *Proceedings of the VLDB Endowment*, v. 15, n. 5, p. 1053–1065, 2022. Disponível em: <https://www.vldb.org/pvldb/vol15/p1053-rinberg.pdf>.

SHAPIRO, M. et al. Conflict-free replicated data types. In: *Stabilization, Safety, and Security of Distributed Systems (SSS 2011)*. Springer, 2011. (LNCS, v. 6976), p. 386–400. Acesso em: 17 ago. 2025. Disponível em: <https://gsd.di.uminho.pt/members/cbm/members/cbm/ps/sss2011.pdf>.

SPINELLIS, D. Version control systems. *IEEE Software*, IEEE, v. 22, n. 5, p. 108–109, 2005.

STAHL, G.; KOSCHMANN, T.; SUTHERS, D. Computer-supported collaborative learning: An historical perspective. In: SAWYER, R. K. (Ed.). *Cambridge Handbook of the Learning Sciences*. Cambridge: Cambridge University Press, 2006. p. 409–426.

SUN, C. Operational transformation in real-time group editors: Issues, algorithms, and achievements. In: *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98)*. Seattle, WA: ACM, 1998. p. 59–68. Disponível em: <https://dl.acm.org/doi/10.1145/289444.289469>.

The Kubernetes Authors. *Kubernetes Documentation*. 2025. <https://kubernetes.io/docs/>. Acesso em: 11 fev. 2025.

WILLIAMS, L.; KESSLER, R. *Pair Programming Illuminated*. [S.l.]: Addison-Wesley, 2002.

WILLIAMS, R. T. An academic review of virtual learning environments. *ICRRD Quality Index Research Journal*, v. 3, n. 2, p. 143–145, 2022.

WU, Q.; PU, C. *Consistency in Real-time Collaborative Editing Systems Based on Partial Persistent Sequences*. [S.l.], 2009. Acesso em: 17 ago. 2025. Disponível em: <https://repository.gatech.edu/server/api/core/bitstreams/b75c1b6b-158a-4fc4-9414-0a05ebe39dc5/content>.