UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

GUSTAVO PAIVA

DESENVOLVIMENTO DE UMA HIPERMÍDIA ADAPTATIVA PARA O ESTUDO DA LÍNGUA LATINA

CAXIAS DO SUL

2026

GUSTAVO PAIVA

DESENVOLVIMENTO DE UMA HIPERMÍDIA ADAPTATIVA PARA O ESTUDO DA LÍNGUA LATINA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Elisa Boff

CAXIAS DO SUL

2026

GUSTAVO PAIVA

DESENVOLVIMENTO DE UMA HIPERMÍDIA ADAPTATIVA PARA O ESTUDO DA LÍNGUA LATINA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado(a) em 30/06/2026

BANCA EXAMINADORA

Profa. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

Prof. Dra. Iraci Cristina Da Silveira De Carli
Universidade de Caxias do Sul - UCS

Prof. Dr. Marcos Eduardo Casa
Universidade de Caxias do Sul - UCS

“A coruja de Minerva abre suas asas somente ao cair da noite”

G.W.F. Hegel

RESUMO

O ensino da língua latina é historicamente marcado por uma dicotomia entre o método focado em Gramática-Tradução e os métodos de Leitura/Direto, sendo ambas abordagens lineares no formato *one-size-fits-all*. Este trabalho tem como objetivo central propor a concepção e modelagem de um Sistema Hipermídia Adaptativo capaz de reconciliar essas duas vertentes pedagógicas, personalizando o percurso de aprendizado. Para isso, a solução *Lingua Latina adaptativa* do TCC foi projetada utilizando a metodologia de engenharia de software UWE, que estende a UML para o design de aplicações hipermídia. O sistema foi detalhado em suas três fases de modelagem: o Design Conceitual, que define o Modelo de Domínio (com conceitos gramaticais, textos e provas), o Modelo de Usuário (rastreando o progresso do aluno) e o Modelo de Adaptação (baseado em regras de pré-requisito); o Design de Navegação (definindo os contextos e links); e o Design de Apresentação (modelando a interface). Para materializar a especificação de modelagem, desenvolveu-se um protótipo funcional utilizando o framework Next.js, integrado ao banco de dados relacional PostgreSQL e hospedado na plataforma Vercel. O protótipo implementa a lógica do grafo de dependências adaptativas no editor visual React Flow, empregando o algoritmo de Tarjan para garantir a aciclicidade das lições e evitar deadlocks pedagógicos. A validação do sistema foi realizada por meio de uma Avaliação Heurística de Usabilidade baseada nos dez princípios de Nielsen e por um ensaio empírico de relato de experiência de usuário com um profissional de ensino de línguas. Os resultados atestam a qualidade ergonômica da plataforma e a adequação do percurso adaptativo para o aluno.

Palavras-chave: Hipermídia Adaptativa, Ensino de Latim, Modelagem UML, Engenharia de Software para Web, UWE, Modelo de Usuário.

ABSTRACT

The teaching of the Latin language is historically marked by a dichotomy between the Grammar-Translation method and the Reading/Direct methods, with both traditional approaches following a linear, "one-size-fits-all" model. This work's central objective is to propose the design and modeling of an Adaptive Hypermedia System capable of reconciling these two pedagogical approaches by personalizing the learning path. To this end, the *Lingua Latina Adaptativa* solution was designed using the UWE software engineering methodology, which extends UML for hypermedia application design. The system was detailed in its three modeling phases: Conceptual Design, which defines the Domain Model (with grammatical concepts, texts, and exams), the User Model (tracking student progress), and the Adaptation Model (based on prerequisite rules); Navigational Design (defining contexts and links); and Presentational Design (modeling the interface). To materialize the modeling specification, a functional prototype was developed using the Next.js framework, integrated with a PostgreSQL database, and hosted on the Vercel platform. The prototype implements the adaptive dependency graph logic on a React Flow visual editor, employing Tarjan's algorithm to ensure acyclicity and prevent pedagogical deadlocks. The system's validation was carried out through a Usability Heuristic Evaluation based on Nielsen's ten principles and an empirical user experience trial with a language teaching professional. The results attest to the platform's ergonomic quality and the suitability of the adaptive learning path for the student.

Keywords: Adaptive Hypermedia, Latin Language Teaching, UML Modeling, Web Engineering, UWE, User Model.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AHAM	Adaptive Hypermedia Application Model
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
OCL	Object Constraint Language
SHA	Sistema de Hipermídia adaptativa
UML	Unified Modeling Language
UWE	UML-based Web Engineering
WWW	World Wide Web
PaaS	Platform as a Service

SUMÁRIO

1 INTRODUÇÃO

A língua latina, embora frequentemente classificada como uma língua "morta", permanece como o pilar fundamental da cultura ocidental e a raiz de um vasto léxico global. Para os falantes de línguas românicas, como o português, estudar latim não é apenas um exercício acadêmico, mas o ato de "pisar a ponte mental"(LOURENÇO, 2021) que leva da nossa própria língua à estrutura que a originou.

Contudo, o ensino do latim enfrenta um desafio pedagógico secular, marcado por uma profunda dicotomia metodológica. De um lado, situa-se o método tradicional de Gramática-Tradução, cristalizado em obras como a de Napoleão Mendes de Almeida, que defende o domínio exaustivo da morfologia e sintaxe como ferramenta para "desenvolver a inteligência"e o "espírito de análise"(ALMEIDA, 2011). De outro, encontram-se os métodos modernos de Leitura ou Direto (como os de (JONES; SIDWELL, 2012) e (ORBERG, 2011)), que surgiram da necessidade prática de focar na "habilidade de leitura de textos originais"(LEITE; BARBOSA *et al.*, 2014) no "espaço mínimo"(LEITE; BARBOSA *et al.*, 2014) que a disciplina ocupa nos currículos atuais.

Ambas as abordagens, em suas implementações tradicionais em sala de aula ou em livro didático, sofrem de uma limitação fundamental: a rigidez. Elas impõem um percurso linear, "um-tamanho-serve-para-todos"("one-size-fits-all") (BRUSILOVSKY, 2001), incapaz de se ajustar aos diferentes ritmos, objetivos e dificuldades de cada aluno. Um estudante pode dominar as declinações rapidamente mas ter dificuldade com a sintaxe dos verbos, enquanto outro pode ter o problema inverso; ambos, no entanto, são forçados a seguir a mesma página e os mesmos exercícios.

É neste cenário que este trabalho propõe a aplicação de uma solução de engenharia de software: os Sistemas Hipermídia Adaptativos (SHA).

A Hipermídia Adaptativa (HA) é uma alternativa à abordagem estática (BRUSILOVSKY, 2001). Um SHA é um sistema que "constrói um modelo das metas, preferências e conhecimento de cada usuário individual"(o Modelo de Usuário) e utiliza esse modelo para "adaptar vários aspectos visíveis do sistema"às necessidades desse usuário (BRUSILOVSKY, 1996; BRUSILOVSKY, 2001).

Este trabalho argumenta que a arquitetura de um SHA é a solução ideal para reconciliar a tensão pedagógica do ensino de Latim. Um sistema devidamente modelado pode:

1. **Respeitar o rigor gramatical:** Através do **Suporte à Navegação Adaptativa** (técnica de *ocultação* ou *anotação*) (BRUSILOVSKY, 1996), o sistema pode ocultar textos e exercícios cujos pré-requisitos gramaticais (definidos no Modelo de Domínio) o aluno ainda não domina.

2. **Promover a leitura contextual:** Através da **Apresentação Adaptativa** (técnica de *texto condicional* ou *stretchtext*) (BRUSILOVSKY, 1996), o sistema pode exibir um texto clássico e fornecer explicações gramaticais ou vocabulário adicional apenas para os conceitos que o Modelo de Usuário identifica como "não dominados"por aquele aluno específico.

O desafio, portanto, não é apenas pedagógico, mas de engenharia de software: como projetar e estruturar um sistema tão complexo? Abordagens genéricas de engenharia de software "não são suficientes para suportar as particularidades do processo de desenvolvimento de Sistemas de Hipermídia Adaptativa"(KOCH; WIRSING, 2001).

1.1 PROBLEMA DE PESQUISA E OBJETIVOS

O debate pedagógico identificado — a rigidez das abordagens existentes diante da diversidade de ritmos e objetivos dos alunos — aponta para uma lacuna concreta que a Hipermídia Adaptativa pode preencher. Cabe, portanto, investigar de que forma um sistema adaptativo pode ser modelado para atender simultaneamente às exigências de ambos os métodos.

Este trabalho parte da seguinte questão de pesquisa: "Como um Sistema Hipermídia Adaptativo pode ser modelado para reconciliar as abordagens pedagógicas de Gramática-Tradução e Leitura, personalizando o percurso de ensino da Língua Latina às necessidades individuais de cada aluno?"

Para responder a esta questão, adota-se a metodologia UWE (*UML-based Web Engineering*), associada ao Modelo de Munique, como instrumento de design (KOCH; MANDEL, 1999; KOCH; WIRSING, 2001). Esta escolha justifica-se por sua capacidade de integrar, desde a concepção, a semântica do domínio com os mecanismos de adaptação baseados no Modelo de Usuário.

O objetivo geral deste trabalho é desenvolver a modelagem de software para um Sistema Hipermídia Adaptativo voltado ao estudo da Língua Latina, utilizando a metodologia *UML-based Web Engineering* (UWE) (KOCH; MANDEL, 1999).

Para atingir este objetivo, definem-se os seguintes **objetivos específicos**:

- Realizar uma revisão da literatura sobre Sistemas Hipermídia Adaptativos, focando nos métodos e técnicas de adaptação (BRUSILOVSKY, 1996) e nos modelos de arquitetura formais (Dexter (HALASZ; SCHWARTZ, 1994), AHAM (BRA; HOUBEN; WU, 1999) e Munich/UWE (KOCH; WIRSING, 2001)).
- Analisar o domínio de ensino da Língua Latina, seu percurso histórico no Brasil (LEITE; BARBOSA *et al.*, 2014) e os principais métodos pedagógicos (Gramática-Tradução (ALMEIDA, 2011) e Leitura/Direto (JONES; SIDWELL, 2012; ORBERG, 2011)).

- Desenvolver o **Design Conceitual** da solução, detalhando o Modelo de Domínio (gramática, textos, provas), o Modelo de Usuário (aluno e professor) e o Modelo de Adaptação (pré-requisitos) através de um Diagrama de Classes UML.
- Propor o **Design de Navegação**, identificando as classes navegáveis («`navigationalClass`») e os contextos de acesso («`index`», «`query`») (KOCH; MANDEL, 1999).
- Detalhar o **Design de Apresentação**, modelando a composição estática e o comportamento dinâmico da interface (KOCH; MANDEL, 1999).

1.2 METODOLOGIA

Esta pesquisa adota o *Design Science Research* (DSR) como paradigma metodológico central (HEVNER *et al.*, 2004), de natureza aplicada e qualitativa, voltado à construção e avaliação de artefatos de tecnologia da informação para a solução de problemas relevantes.

A pesquisa bibliográfica serve como instrumento de rigor, sustentando os Capítulos 2 e 3 por meio da análise crítica da literatura sobre Hipermídias Adaptativas (BRUSILOVSKY, 1996; BRUSILOVSKY, 2001; PALAZZO, 2000) e sobre o ensino da língua latina (LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012; ALMEIDA, 2011).

O artefato produzido é um modelo, tipologia definida por Hevner *et al.* (2004): a especificação UML do sistema *Lingua Latina Adaptativa*, composta pelos Designs Conceitual, de Navegação e de Apresentação. O problema que o motiva, a rigidez dos métodos tradicionais de ensino de latim diante das necessidades individuais dos alunos, satisfaz o critério de **relevância do problema** das diretrizes do DSR (HEVNER *et al.*, 2004).

A avaliação do artefato nesta etapa (TCC 1) é de natureza descritiva e argumentativa: demonstra-se que a metodologia UWE é adequada para estruturar os requisitos adaptativos do domínio.

1.3 ESTRUTURA DO TRABALHO

Este Trabalho de Conclusão de Curso está estruturado em sete capítulos:

- O **Capítulo 1** apresenta a contextualização do problema pedagógico no ensino de Latim e introduz a Hipermídia Adaptativa como solução, definindo os objetivos e a estrutura da pesquisa.
- O **Capítulo 2**, "Fundamentação Teórica", revisa a literatura sobre Hipermídia Adaptativa, desde os conceitos e técnicas fundamentais de adaptação até as principais arquiteturas de referência (Dexter, AHAM, Munich/UWE e SHASIM).

- O **Capítulo 3**, "O Domínio da Língua Latina", analisa o percurso histórico do ensino de latim no Brasil, detalha os métodos pedagógicos concorrentes e explora as particularidades gramaticais da língua que impactam o design do sistema.
- O **Capítulo 4**, "Modelagem da Solução Proposta", apresenta a aplicação prática da metodologia de engenharia de software (KOCH; MANDEL, 1999), detalhando os modelos Conceitual, de Navegação e de Apresentação da solução *Lingua Latina Adaptiva*.
- O **Capítulo 5**, "Implementação da Solução Proposta", descreve a materialização do protótipo, a arquitetura adotada, o processo de desenvolvimento e as decisões técnicas que sustentam a aplicação.
- O **Capítulo 6**, "Testes e Avaliação de Usabilidade", apresenta a avaliação heurística do protótipo e discute os principais achados de usabilidade.
- O **Capítulo 7**, "Considerações Finais", resume os resultados alcançados e retoma as contribuições do trabalho, além de indicar possibilidades de continuidade.

2 HIPERMÍDIAS ADAPTATIVAS

O presente capítulo tem como objetivo estabelecer a fundamentação teórica que sustenta o desenvolvimento deste trabalho. Inicialmente, apresenta-se um breve histórico e a definição de hipermídia, diferenciando-a de sistemas convencionais.

Na sequência, o estudo aprofunda-se no conceito de Hipermídia Adaptativa (HA), que é o pilar central desta pesquisa. Serão analisados os métodos e técnicas que permitem a um sistema hipermídia adaptar-se ao usuário, com foco nos modelos de Adaptação de Conteúdo e Adaptação de Navegação.

2.1 HIPERMÍDIAS

Conforme (GROSS; Akşimşek; STEPINSKI, 2023), pode-se traçar o início da ideia no clássico artigo de Vannevar Bush (BUSH *et al.*, 1945); Ali, há a introdução do conceito seminal de hipermídia cuja manifestação ocorre no dispositivo teórico chamado *Memex*, que funcionava como uma extensão da memória humana, permitindo a ligação associativa de informações:

> Um MEMEX é um dispositivo que permitirá a uma pessoa armazenar todos os seus livros, arquivos, e comunicações, e que é mecanizado de tal forma que poderá ser consultado com grande velocidade e flexibilidade. Na verdade, seria um suplemento ampliado e íntimo de sua memória. [...] Isto representa um próximo passo para a indexação associativa, cuja ideia básica consiste em possibilitar que cada um dos elementos selecione ou busque outro elemento automaticamente e imediatamente. Esta é a característica essencial do MEMEX. (BUSH, 2016). [1]

Porém, foi em 1965 que pela primeira vez os termos hipermídia e hipertexto vieram à luz, no artigo de Ted Nelson (NELSON, 1965) sobre a estrutura interna do projeto Xanadu:

> Permitam-me introduzir a palavra "hipertexto"para significar um corpo de material escrito ou pictórico interconectado de uma forma tão complexa que não poderia ser convenientemente apresentado ou representado em papel. Ele pode conter resumos, ou mapas de seus conteúdos e suas inter-relações; pode conter anotações, adições e notas de rodapé de estudiosos que o examinaram. Permitam-me sugerir que tal objeto e sistema, devidamente projetado e administrado, poderia ter grande potencial para a educação, aumentando o leque de escolhas do estudante, seu senso de liberdade, sua motivação e sua compreensão intelectual.
>
> [...]Filmes, gravações de som e gravações de vídeo também são sequências lineares [...] O hiperfilme – um filme navegável ou de sequência variável – é

[1] Original: “A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory. [...] It affords an immediate step, however, to associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the memex. (BUSH *et al.*, 1945)”

> apenas uma das possíveis hipermídias que exigem nossa atenção (Tradução nossa). [2]

Embora Nelson tenha introduzido ambos os termos, eles são frequentemente utilizados de forma intercambiável. A distinção fundamental, como apontado por Hall, Davis e Hutchings (2012), é que o hipertexto, em seu sentido estrito, aplica-se apenas a sistemas baseados em texto; a hipermídia é simplesmente a extensão do hipertexto para incluir dados multimídia, como áudio e vídeo (HALL; DAVIS; HUTCHINGS, 2012).

A visão de Nelson e Bush só começou a se materializar comercialmente com o advento dos computadores pessoais. Os primeiros produtos notáveis a chegarem ao mercado foram o Hyperties e o Guide. O Guide, desenvolvido por Peter Brown, foi lançado comercialmente em 1986 para a plataforma Macintosh (HALL; DAVIS; HUTCHINGS, 2012).

Contudo, o evento que mais popularizou o hipertexto foi o lançamento do HyperCard (Figura 1) pela Apple em 1987 (HALL; DAVIS; HUTCHINGS, 2012). A Apple tomou a decisão estratégica de distribuir o HyperCard gratuitamente com cada Macintosh vendido, introduzindo o conceito a um público massivo (HALL; DAVIS; HUTCHINGS, 2012). O modelo do HyperCard, baseado em "pilhas"(*stacks*) e "cartões"(*cards*) com botões e links do tipo "goto", tornou-se a norma de design para a interação hipermídia na época (HALL; DAVIS; HUTCHINGS, 2012).

Paralelamente, o conceito de hipermídia foi aplicado em escala global com o desenvolvimento da **World Wide Web (WWW)** por Tim Berners-Lee no CERN, a partir de 1989 (HALL; DAVIS; HUTCHINGS, 2012). O sucesso da Web foi impulsionado por seus protocolos abertos (HTTP e HTML) e pela facilidade de uso proporcionada por navegadores gráficos como o Mosaic (HALL; DAVIS; HUTCHINGS, 2012).

Este novo sistema girava em torno de uma hipermídia central, o HTML (Figura 2), que funcionava como um sistema de publicação de documentos interligados. Conforme (GROSS; Akşimşek; STEPINSKI, 2023), o HTML foi inicialmente concebido como uma hipermídia somente de leitura:

> O sistema que Berners-Lee, Fielding e muitos outros haviam criado girava em torno de uma hipermídia: o HTML. O HTML começou como uma hipermídia apenas de leitura, usada para publicar documentos acadêmicos. Estes documentos eram ligados entre si por meio de *tags* âncora que criavam *hyperlinks* entre eles, permitindo aos usuários navegar rapidamente entre os documentos (Tradução nossa). [3]

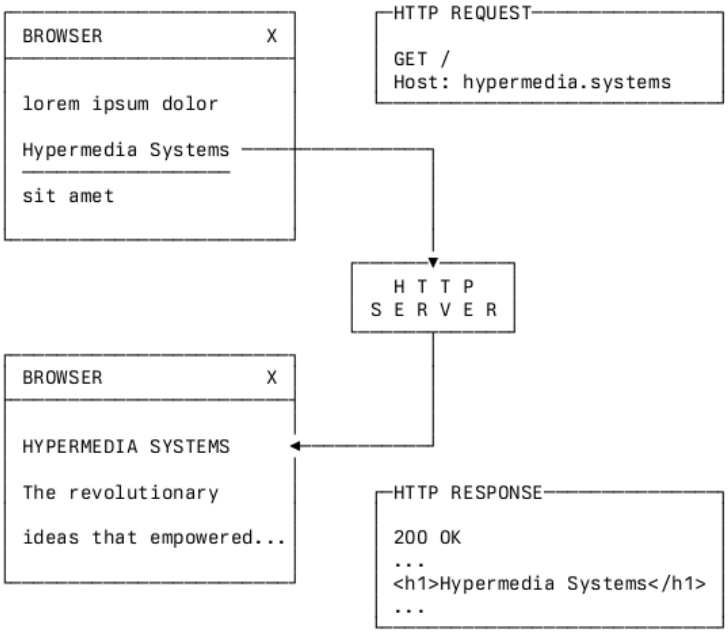
---

[2] Original: “Let me introduce the word "hypertext"to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper. It may contain sunmmries, or maps of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it. Let me suggest that such an object and system, properly designed and administered, could have great potential for education, increasing the student’s range of choices, his sense of freedom, his motivation, and his intellectual grasp. [...]Films, sound recordings, and video recordings are also linear strings [...] The hyperfilm– a browsable or vari-sequenced movie– is only one of the possible hypermedia

Figura 1 – Exemplo Apple Hypercard 2.4

Fonte: Disponível em: <https://www.macintoshrepository.org/2632-hypercard-2-4>. Acesso em: 4 nov. 2025

Figura 2 – O modelo de navegação hipermídia na WWW através do protocolo HTTP.



Fonte: Adaptado de (GROSS; Akşimşek; STEPINSKI, 2023).

that require our attention. (NELSON, 1965)"

[3] Original: "The system that Berners-Lee, Fielding and many others had created revolved around a hypermedia: HTML. HTML started as a read-only hypermedia, used to publish (at first) academic documents. These documents were linked together via anchor tags which created hyperlinks between them, allowing users to quickly

2.2 HIPERMÍDIAS ADAPTATIVAS

Embora a hipermídia, como a World Wide Web, tenha revolucionado o acesso à informação, a sua implementação tradicional padece de uma limitação fundamental: a sua natureza estática. Sistemas de hipermídia tradicionais oferecem o mesmo conteúdo e os mesmos links para todos os usuários (BRUSILOVSKY, 2001). Esta abordagem é frequentemente descrita como *one-size-fits-all*.

Assim, um sistema hipermídia tradicional falha em reconhecer a diversidade do seu público, apresentando o mesmo conteúdo e os mesmos links para usuários com diferentes níveis de conhecimento, interesses e objetivos (BRUSILOVSKY, 1996; BRUSILOVSKY, 2001; PALAZZO, 2000). Tais usuários são tratados de forma idêntica (BRUSILOVSKY, 1996). ilustra-se esta problemática com clareza:

> Por exemplo, um sistema tradicional de hipermídia educacional apresentará a mesma explicação estática e sugerirá a mesma próxima página para estudantes com objetivos educacionais e conhecimento do assunto amplamente distintos. Similarmente, uma enciclopédia eletrônica estática apresentará a mesma informação e o mesmo conjunto de links para artigos relacionados para leitores com diferentes conhecimentos e interesses (Tradução nossa). [4]

Este descompasso entre a informação estática e o usuário dinâmico pode levar a problemas de usabilidade, como a sobrecarga cognitiva (excesso de informação irrelevante) ou a desorientação em um hiperespaço por vezes caótico (PALAZZO, 2000). Para solucionar estas limitações, surge o campo da Hipermídia Adaptativa: uma área de pesquisa na interseção da hipermídia com a modelagem do usuário (BRUSILOVSKY, 2001; BRUSILOVSKY, 1996). O objetivo é personalizar a interação, tornando o sistema dinâmico:

> Hipermídia Adaptativa (HA) é a área da ciência da computação que se ocupa do estudo e desenvolvimento de sistemas, arquiteturas, métodos e técnicas capazes de promover a adaptação de hiperdocumentos e hipermídia em geral às expectativas, necessidades, preferências e desejos de seus usuários. (PALAZZO, 2000)

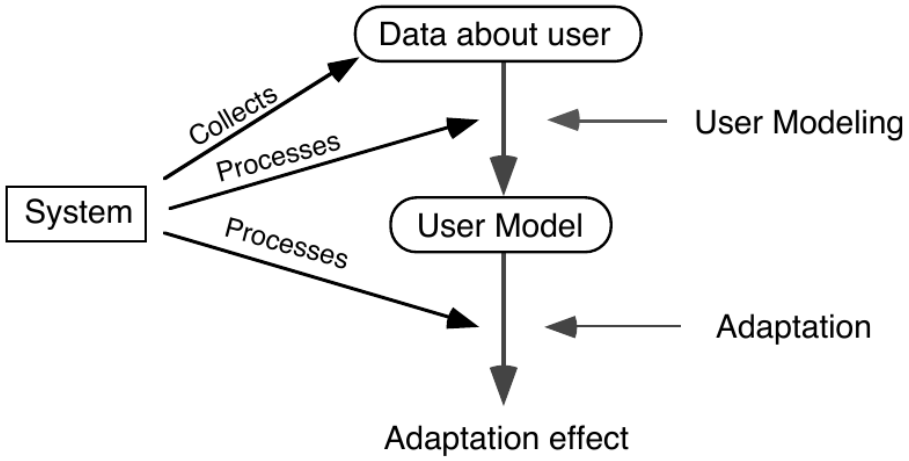
Por fim, para que um sistema seja considerado de Hipermídia Adaptativa, ele deve satisfazer a três critérios básicos (BRUSILOVSKY, 1996; PALAZZO, 2000):

1. Ser um sistema de hipertexto ou hipermídia;
2. Possuir um Modelo do Usuário;
3. Ser capaz de adaptar a hipermídia utilizando este modelo.

navigate between documents. (GROSS; Akşimşek; STEPINSKI, 2023)"

[4] Original: "For example, a traditional educational hypermedia system will present the same static explanation and suggest the same next page to students with widely differing educational goals and knowledge of the subject. Similarly, a static electronic encyclopedia will present the same information and same set of links to related articles to readers with different knowledge and interests. (BRUSILOVSKY, 2001)"

Figura 3 – Espaços de Adaptação em Hipermídia Adaptativa Clássica.



Fonte: Adaptado de (BRUSILOVSKY, 1996).

2.2.1 Modelagem do Usuário

Como estabelecido na seção anterior, o que diferencia a Hipermídia Adaptativa (HA) é sua capacidade de se moldar ao usuário. O componente que torna essa adaptação possível é o Modelo do Usuário. Este modelo é a característica distintiva de qualquer sistema adaptativo (BRUSILOVSKY; MILLÁN, 2007).

A modelagem do usuário é o processo de coletar dados do indivíduo e utilizá-los para construir uma representação de suas características essenciais (BRUSILOVSKY, 1996). Em termos simples, o modelo do usuário é a fonte de conhecimento que o sistema possui sobre o indivíduo; uma base de dados persistente que armazena informações sobre seus objetivos, preferências, conhecimento e interesses (BRUSILOVSKY, 2001; GARLATTI; PRIE, 2004; PALAZZO, 2000).

Conforme Brusilovsky e Millán (2007), o modelo do usuário é essencial para o sistema realizar o efeito adaptativo:

> O modelo de usuário é uma representação de informações sobre um usuário individual que é essencial para que um sistema adaptativo forneça o efeito de adaptação, ou seja, para se comportar de forma diferente para diferentes usuários. [...] Para criar e manter um modelo de usuário atualizado, um sistema adaptativo coleta dados para o modelo de usuário a partir de várias fontes, que podem incluir a observação implícita da interação do usuário e a solicitação explícita de entrada direta do usuário. Este processo é conhecido como modelagem de usuário (Tradução nossa).[5]

[5] Original: "The user model is a representation of information about an individual user that is essential for an adaptive system to provide the adaptation effect, i.e., to behave differently for different users. [...] To create and maintain an up-to-date user model, an adaptive system collects data for the user model from various sources that may include implicitly observing user interaction and explicitly requesting direct input from the user. This process is known as user modeling. (BRUSILOVSKY; MILLÁN, 2007)"

O ciclo de adaptação (Figura 3) é, portanto, totalmente dependente deste modelo.

Embora o conceito de modelagem do usuário seja central, sua implementação em sistemas de hipermídia apresenta um desafio específico: a dificuldade em obter dados confiáveis automaticamente. Em sistemas de diálogo ou tutores inteligentes tradicionais, a entrada do usuário é rica (respostas a problemas, comandos explícitos). Em hipermídia, a entrada é pobre.

Brusilovsky (1996) aponta esta dificuldade, afirmando que a observação do usuário em hipermídia fornece informações insuficientes para a modelagem:

> A única informação sobre as ações do usuário que o sistema pode registrar é o caminho do usuário pelo hiperespaço e o tempo gasto em cada nó. [...] o facto do usuário ter visitado uma página várias vezes ou gasto um tempo razoável nela não garante sequer que o usuário leu atentamente o seu conteúdo. Este tipo de informação não é confiável e não pode ser usado como a única fonte para construir o modelo de usuário (Tradução nossa).[6]

Palazzo (2000) reforça esta visão, afirmando que "a observação da ação do usuário não oferece elementos suficientes para a sua modelagem"(PALAZZO, 2000, p. 33), pois o caminho percorrido e o tempo gasto no nodo são "insuficientes para a produção de um modelo confiável"(PALAZZO, 2000, p. 33).

Dado que a modelagem puramente automática (implícita) em hipermídia é inerentemente não confiável, os sistemas de HA devem recorrer a fontes adicionais de informação, muitas vezes envolvendo o usuário diretamente no processo (BRUSILOVSKY, 1996). Isso é conhecido como Modelagem Colaborativa de Usuário.

Alguns componentes do modelo, como o histórico de navegação, podem ser deduzidos pelo sistema, mas outros, como as preferências, não. Conforme Palazzo (2000), "alguns componentes do modelo do usuário, tais como sua história e preferências, não podem em absoluto ser deduzidos e precisam ser fornecidos diretamente pelo usuário"(PALAZZO, 2000, p. 33).

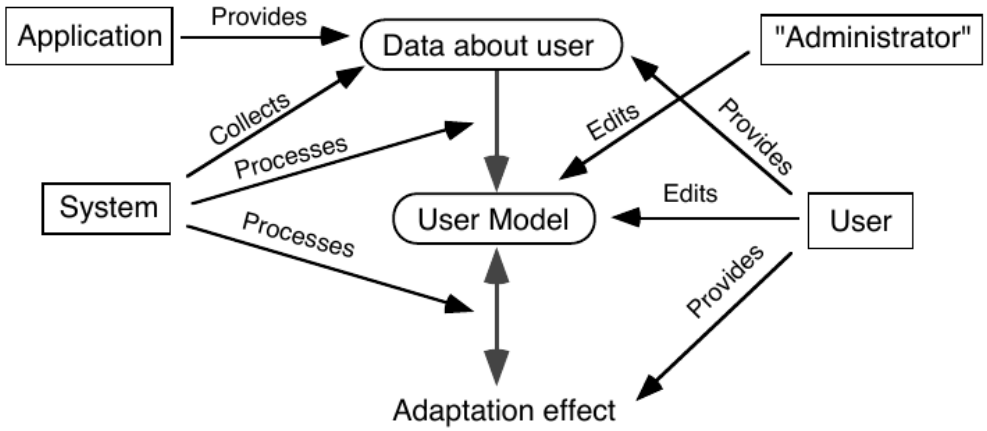
Esta aquisição de dados pode ocorrer de múltiplas formas, como ilustrado na Figura 4. O sistema pode coletar dados da aplicação, mas também pode receber informações explícitas do próprio usuário ou de um administrador (BRUSILOVSKY, 1996).

Assim, um modelo de usuário é composto por um conjunto de características que representam o indivíduo. A literatura em HA, notavelmente Palazzo (2000) e Brusilovsky e Millán (2007), identifica um conjunto central de características que podem ser modeladas.:

a) **Conhecimento:** É a característica mais importante para sistemas educacionais (AES) e de hipermídia adaptativa (AHS) (PALAZZO, 2000; BRUSILOVSKY; MILLÁN, 2007).

[6] Original: “The only information about the user’s actions which the system can record is the user’s path through the hyperspace and the time spent on each node. [...] the fact that the user has visited a page several times or spent reasonable time on it does not even guarantee that the user attentively read its content. This kind of information is not reliable and can not be used as the only source for building the user model. (BRUSILOVSKY, 1996)”

Figura 4 – Modelagem Colaborativa do Usuário em Hipermídia Adaptativa.



Fonte: (BRUSILOVSKY, 1996)

Representa o conhecimento do usuário sobre o assunto (o domínio). É uma característica dinâmica, pois o sistema deve ser capaz de "reconhecer as modificações produzidas no conhecimento do usuário para atualizar adequadamente o seu modelo"(PALAZZO, 2000, p. 32).

b) **Interesses:** Representa os interesses de longo prazo do usuário. Nos últimos anos, os interesses têm competido com o conhecimento como a característica mais importante a ser modelada, especialmente em sistemas de informação e recomendação (BRUSILOVSKY; MILLÁN, 2007).

c) **Objetivos:** Representa o propósito imediato do usuário no sistema. É a característica mais volátil, podendo mudar várias vezes dentro da mesma sessão (PALAZZO, 2000; BRUSILOVSKY; MILLÁN, 2007). Responde à pergunta: "O que exatamente [o usuário] deseja obter dele?"(PALAZZO, 2000, p. 32).

d) **Background e História:** *Background* refere-se à experiência prévia do usuário fora do domínio principal do sistema (ex: profissão) (BRUSILOVSKY; MILLÁN, 2007; PALAZZO, 2000). *Experiência*, por outro lado, denota a familiaridade do usuário com a própria estrutura e navegação do sistema hipermídia (PALAZZO, 2000).

e) **Preferências e Traços Individuais:** São características estáveis que definem o indivíduo. As *Preferências* geralmente não podem ser deduzidas e precisam ser declaradas pelo usuário (PALAZZO, 2000; GARLATTI; PRIE, 2004, p. 32-33). Os *Traços Individuais* incluem estilos cognitivos (ex: holista/serialista) e estilos de aprendizagem (BRUSILOVSKY; MILLÁN, 2007).

2.2.2 Métodos de Adaptação de Conteúdo

A adaptação de conteúdo, também conhecida como apresentação adaptativa, foca em como a informação textual ou multimídia é apresentada em uma página específica. O objetivo é ajustar o que o usuário vê com base em seu conhecimento, objetivos ou preferências (BRUSILOVSKY, 1996; BRUSILOVSKY, 2001). Conforme destacado em (BRA; HOUBEN; WU, 1999), pode ser desejável apresentar a informação de diferentes maneiras, dependendo das características de cada usuário (BRA; HOUBEN; WU, 1999, p. 2).

Os principais métodos para realizar a adaptação de conteúdo incluem:

a) **Explicações Adicionais:** Este é um dos métodos mais comuns e intuitivos (BRUSILOVSKY, 1996). A ideia central é gerenciar a visibilidade da informação com base no modelo do usuário. Detalhes técnicos complexos podem ser ocultados de usuários iniciantes, pois eles poderiam não compreendê-los. Inversamente, explicações introdutórias, essenciais para novatos, podem ser omitidas para usuários especialistas, que não precisam mais delas, tornando a apresentação menos verbosa e mais direta (BRUSILOVSKY, 1996).

b) **Explicações como Pré-requisito:** Este método foca na estrutura de dependência do conhecimento. Antes de apresentar a explicação de um conceito principal, o sistema insere dinamicamente explicações de todos os conceitos de pré-requisito que o usuário (segundo seu modelo) ainda não domina (BRUSILOVSKY, 1996). Isso garante que o usuário tenha a base necessária para compreender o novo assunto, seguindo uma ordem pedagógica lógica.

c) **Explicações Comparativas:** Este método tira proveito do conhecimento existente do usuário sobre conceitos similares. Se o modelo do usuário indica que ele conhece um conceito "A"e está aprendendo um conceito "B"similar, o sistema pode gerar uma explicação comparativa. Esta explicação destaca ativamente as semelhanças e diferenças entre A e B, o que é particularmente eficaz em domínios como o aprendizado de linguagens de programação (BRUSILOVSKY, 1996).

d) **Variantes de Explicação:** Este método assume que, para diferentes usuários, a informação necessária pode ser essencialmente diferente, e não apenas um subconjunto de uma informação maior. Em vez de apenas mostrar ou ocultar partes, o sistema armazena múltiplas versões (variantes) de uma mesma explicação. Com base no modelo do usuário (seu conhecimento, background, etc.), o sistema seleciona e apresenta a variante mais adequada (BRUSILOVSKY, 1996).

e) **Ordenação de Fragmentos:** Este método foca na ordem da apresentação dentro de uma mesma página. A ordem "na qual os itens são exibidos (em uma página) também pode ser diferente para diferentes usuários"(BRA; HOUBEN; WU, 1999, p. 2). O sistema analisa os fragmentos de informação disponíveis (parágrafos, imagens,

exemplos) e os reordena, apresentando primeiro aqueles considerados mais relevantes para os objetivos ou nível de conhecimento do usuário (BRUSILOVSKY, 1996).

2.2.3 Métodos de Adaptação de Navegação

A adaptação de navegação, ou suporte à navegação adaptativa, visa orientar o usuário através do hiperespaço. Isso é feito modificando a forma como os links são apresentados, com o objetivo de guiar o usuário para informações relevantes e mantê-lo afastado de informações não relevantes (BRA; HOUBEN; WU, 1999, p. 3). Um sistema pode, por exemplo, fornecer "um conjunto sugerido de links mais relevantes para prosseguir"(BRUSILOVSKY, 2001, p. 1).

Os métodos de suporte à navegação, conforme resumidos em (PALAZZO, 2000), incluem:

a) **Condução Global:** Este método foca em guiar o usuário por um caminho ótimo através de múltiplas páginas em direção a um objetivo final (como concluir um módulo de aprendizado ou encontrar uma informação específica). O sistema tenta "ajudar o usuário a encontrar o caminho mais curto para a informação que ele deseja, com possíveis desvios minimizados"(PALAZZO, 2000). É um método muito utilizado em sistemas de recuperação de informação e tutores educacionais (BRUSILOVSKY, 1996).

b) **Condução Local:** Possui um "alcance muito menor"(PALAZZO, 2000) que a condução global. Em vez de planejar um caminho longo, a condução local "ocupa-se de um único passo"(PALAZZO, 2000). Ela tenta sugerir, na página atual, quais links são mais relevantes para o próximo clique, considerando as preferências, conhecimento e experiência do usuário (PALAZZO, 2000; BRUSILOVSKY, 1996).

c) **Suporte à Orientação Local:** Este método ajuda o usuário a entender seu posicionamento imediato na rede (PALAZZO, 2000). Frequentemente, isso é feito limitando as escolhas para evitar sobrecarga cognitiva. A decisão de limitar pode ser baseada em:

   - **Baseado em Conhecimento:** Ocultando links para conteúdos que o sistema considera que o usuário "ainda não está apto a aprendê-los"(PALAZZO, 2000), como tópicos que exigem pré-requisitos ainda não vistos (BRUSILOVSKY, 1996).
   - **Baseado em Objetivos:** Ocultando links que "não estão relacionados com os atuais objetivos educacionais"(PALAZZO, 2000) ou de navegação imediata do usuário.

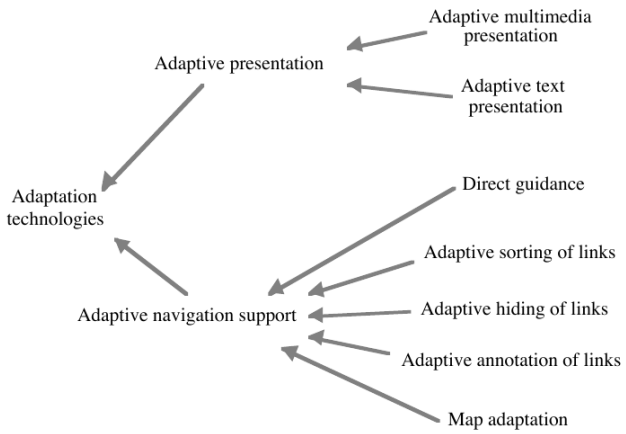
d) **Suporte à Orientação Global:** Diferente da orientação local, este método foca em auxiliar o usuário a "entender a estrutura de todo o hiperespaço que constitui o domínio de navegação do sistema"(PALAZZO, 2000, p. 6). Isso geralmente é feito através

da adaptação de mapas globais do site ou sistema, onde o usuário pode ver seu progresso ou posição (BRUSILOVSKY, 1996).

e) **Gerenciamento de Visões Personalizadas:** Este é um método focado em organizar o espaço de informação para o usuário, permitindo o "gerenciamento de visões personalizadas (na estrutura de links)"(BRA; HOUBEN; WU, 1999, p. 3). Isso pode se materializar na criação de listas de links personalizadas ou favoritos adaptativos, que evoluem com os interesses do usuário (BRUSILOVSKY, 1996).

2.2.4 Técnicas de Adaptabilidade

Figura 5 – Tecnologias em hipermídias adaptativas



Fonte: Adaptado de (BRUSILOVSKY, 1996).

Enquanto os métodos representam a ideia conceitual da adaptação, as técnicas são as implementações de nível mais baixo que executam essa ideia (BRUSILOVSKY, 1996). Um mesmo método pode ser implementado por diferentes técnicas, e uma mesma técnica pode ser usada para implementar diferentes métodos (BRUSILOVSKY, 1996).

As técnicas de adaptabilidade, referidas como "tecnologias"(Figura 5) em (BRUSILOVSKY, 1996), dividem-se em duas categorias principais (conforme ilustrado na Figura 3): adaptação do conteúdo da página e adaptação dos links (BRUSILOVSKY, 1996).

2.2.4.1 Técnicas de Adaptação de Conteúdo

As técnicas de adaptação de conteúdo, ou apresentação adaptativa, são as implementações práticas que permitem aos métodos (discutidos na Seção 2.2.2) funcionarem (BRUSILOVSKY, 1996). Elas variam de baixo nível, exigindo mais configuração, até alto nível, oferecendo estruturas mais robustas.

As principais técnicas identificadas na literatura (BRUSILOVSKY, 1996) são:

a) **Texto Condicional:** Esta é uma técnica de baixo nível, mas muito flexível. A informação sobre um conceito é dividida em múltiplos "pedaços"(chunks) de texto. Cada pedaço é associado a uma condição lógica baseada no modelo do usuário (como seu nível de conhecimento). Ao montar a página, o sistema avalia essas condições e exibe apenas os pedaços de texto cujas condições são verdadeiras, permitindo, por exemplo, ocultar detalhes de um usuário iniciante ou inserir explicações comparativas se um conceito relacionado já for conhecido (BRUSILOVSKY, 1996).

b) **Stretchtext:** Uma técnica de nível mais alto que consiste em porções de texto que podem ser expandidas ou recolhidas pelo usuário. A adaptatividade define o estado inicial desses blocos: o sistema apresenta o texto considerado relevante para o usuário já expandido (visível), enquanto o texto considerado não relevante é apresentado de forma contraída (recolhido), muitas vezes exibindo apenas uma palavra ou frase-chave que o usuário pode clicar para expandir (BRUSILOVSKY, 1996).

c) **Variações de Páginas:** Considerada a técnica de apresentação adaptativa mais simples, consiste em armazenar versões alternativas completas da mesma página. Cada versão é preparada para um estereótipo específico de usuário (por exemplo, "iniciante"ou "especialista"). Quando o usuário acessa o nó, o sistema simplesmente seleciona e exibe a variante da página que melhor corresponde ao seu modelo (BRUSILOVSKY, 1996).

d) **Variações de Fragmento:** Esta é uma implementação mais refinada e granular do que a variação de páginas. Em vez de duplicar a página inteira, o sistema armazena variações apenas para porções (fragmentos) específicos de informação. A página final é montada dinamicamente, combinando os fragmentos que melhor correspondem ao conhecimento do usuário sobre cada conceito individualmente (BRUSILOVSKY, 1996).

e) **Baseado em Frames:** Esta é considerada a técnica mais poderosa de adaptação de conteúdo, sendo capaz de, em teoria, implementar todas as outras. Nela, a informação sobre um conceito é estruturada em um "frame", que possui diversos "slots"(atributos). Esses slots podem conter diferentes explicações, exemplos, links, etc. Regras de apresentação especiais são usadas para decidir quais slots de um frame devem ser apresentados a um usuário específico e em qual ordem, permitindo uma adaptação muito detalhada (BRUSILOVSKY, 1996).

2.2.4.2 Técnicas de Adaptação de Navegação

As técnicas de adaptação de navegação, ou suporte à navegação adaptativa, são as tecnologias usadas para implementar os métodos de orientação (discutidos na Seção 2.2.3). Elas auxiliam os usuários a se orientarem no hiperespaço, adaptando a forma como os links são apresentados com base no modelo do usuário (BRUSILOVSKY, 1996).

As principais técnicas (ou tecnologias) são (BRUSILOVSKY, 1996):

a) **Orientação Direta:** A tecnologia mais simples de suporte à navegação. O sistema decide qual é o "melhor"nó para o usuário visitar a seguir, com base em seu modelo (como seu objetivo ou conhecimento). Isso pode ser implementado destacando visualmente o link sugerido ou, mais comumente, fornecendo um link dinâmico "Próximo". A principal desvantagem é a limitação da liberdade do usuário, oferecendo pouca ajuda se ele não quiser seguir a sugestão (BRUSILOVSKY, 1996).

b) **Ordenação Adaptativa:** Esta técnica reordena dinamicamente os links em uma página (como um menu ou índice) de acordo com o modelo do usuário. Os links considerados mais relevantes para o usuário são movidos para o topo da lista, facilitando sua localização. Esta técnica, no entanto, tem a desvantagem de alterar a estabilidade espacial da interface e se aplica principalmente a links não-contextuais (BRUSILOVSKY, 1996).

c) **Ocultação:** Uma das técnicas mais utilizadas, consiste em restringir o espaço de navegação ocultando links para páginas consideradas "não relevantes". A relevância pode ser determinada pelos objetivos do usuário ou pelo seu nível de conhecimento (por exemplo, ocultando links para tópicos avançados que o usuário ainda não está pronto para aprender). O objetivo principal é proteger o usuário da complexidade do hiperespaço e reduzir a sobrecarga cognitiva (BRUSILOVSKY, 1996).

d) **Anotação Adaptativa:** Em vez de ocultar links, esta técnica os "aumenta"com informações adicionais. O sistema insere comentários, ícones ou altera a cor dos links para informar o usuário sobre o estado atual do nó de destino. Por exemplo, um link pode ser anotado como "recomendado", "não pronto para aprender"ou, na forma mais comum, "já visitado". Esta técnica é considerada poderosa por manter a estabilidade da navegação, ao mesmo tempo que oferece orientação (BRUSILOVSKY, 1996).

e) **Mapa Adaptativo:** Refere-se a técnicas que adaptam a apresentação de mapas do hiperespaço (sejam globais ou locais). Na maioria das implementações, a estrutura do mapa em si não é alterada, mas as técnicas de ocultação e anotação são aplicadas aos nós e links visíveis no mapa (por exemplo, colorindo nós já visitados ou ocultando áreas irrelevantes). Pesquisas mais avançadas exploram a adaptação da própria estrutura do mapa (BRUSILOVSKY, 1996).

2.3 MODELOS DE SISTEMAS HIPERMÍDIA ADAPTATIVOS

Para o desenvolvimento de um sistema de software complexo, como uma hipermídia adaptativa, é fundamental o uso de um modelo de referência. Conforme (HALL; DAVIS; HUTCHINGS, 2012), a meta desses modelos é capturar as abstrações importantes encontradas

em diversos sistemas para criar uma base que permita a "comparação de sistemas"e o "desenvolvimento de padrões de intercâmbio e interoperabilidade"(HALL; DAVIS; HUTCHINGS, 2012).

No contexto da Hipermídia Adaptativa (HA), um sistema é definido como um conjunto de nós e links que "adapta (personaliza) dinamicamente vários aspectos visíveis do sistema às necessidades, preferências ou conhecimento de um usuário individual"(KOCH; WIRSING, 2002). Assim, esta seção tem como objetivo explorar algumas dessas opções existentes.

2.3.1 Dexter

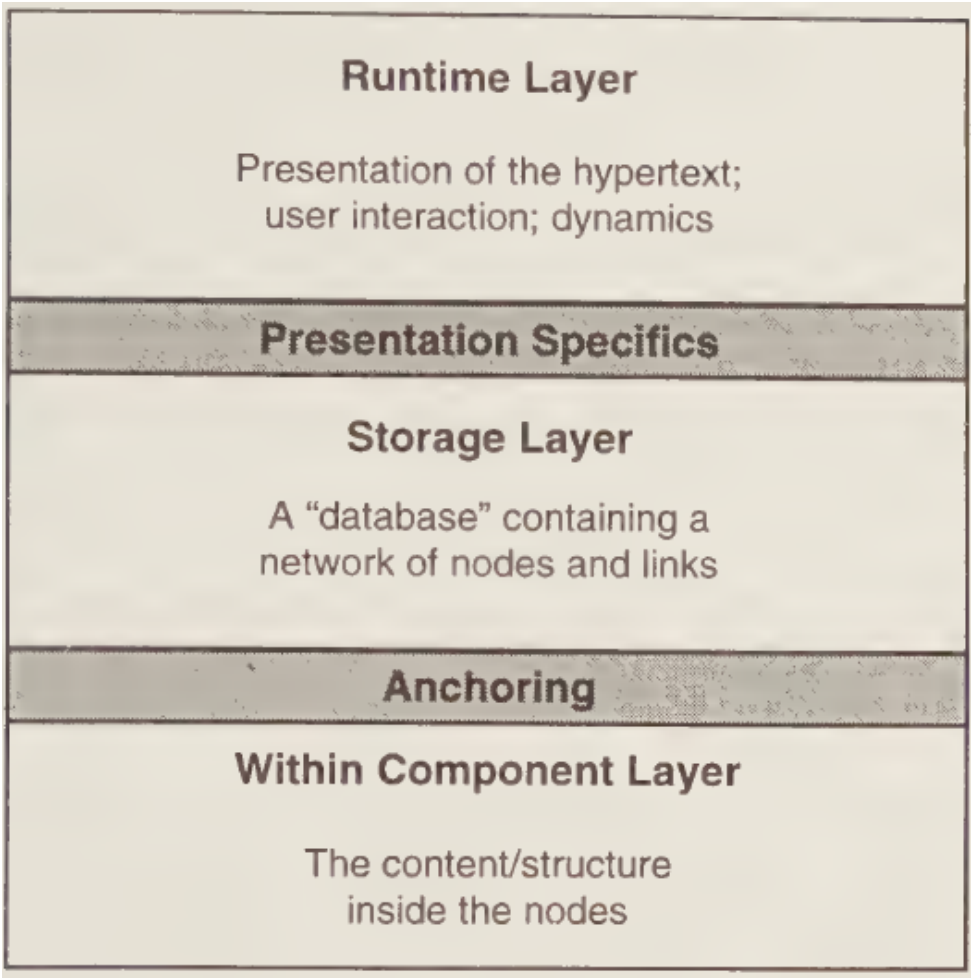
O Modelo Dexter, proposto por Halasz e Schwartz, é considerado o principal modelo de referência formal para sistemas de hipertexto (HALASZ; SCHWARTZ, 1994; HALL; DAVIS; HUTCHINGS, 2012). Seu objetivo foi "capturar as abstrações importantes"encontradas na maioria dos sistemas de hipermídia existentes e futuros (como NoteCards, Intermedia e KMS) (HALASZ; SCHWARTZ, 1994, p. 30). Ao fazer isso, o modelo fornece uma "base de princípios"para comparar diferentes sistemas e desenvolver padrões de intercâmbio e interoperabilidade entre eles (HALASZ; SCHWARTZ, 1994, p. 30), (HALL; DAVIS; HUTCHINGS, 2012, p. 22).

Para evitar a confusão terminológica que existia na época (onde o termo "nó"tinha significados muito diferentes em cada sistema), o Dexter introduziu o termo neutro componente para se referir às unidades básicas de informação (HALASZ; SCHWARTZ, 1994, p. 30, 32).

A arquitetura Dexter divide formalmente um sistema de hipermídia em três camadas distintas (HALASZ; SCHWARTZ, 1994, p. 30, 32), (HALL; DAVIS; HUTCHINGS, 2012, p. 22-23), conforme ilustrado na Figura 6.

- **Camada de Armazenamento:** Esta é a essência do hipertexto (HALASZ; SCHWARTZ, 1994, p. 30). É, efetivamente, um "banco de dados"que descreve a rede de nós (componentes) e links (HALASZ; SCHWARTZ, 1994; HALL; DAVIS; HUTCHINGS, 2012, p. 32, 23). O foco principal do modelo Dexter está nesta camada (HALASZ; SCHWARTZ, 1994, p. 30). Nela, os componentes são tratados como "contêineres genéricos"de dados; a camada de armazenamento não sabe se o conteúdo de um componente é texto, gráfico ou animação (HALASZ; SCHWARTZ, 1994, p. 32).
- **Camada Interna do Componente:** Esta camada cobre o "conteúdo e a estrutura dentro dos componentes"(HALASZ; SCHWARTZ, 1994, p. 30, 32), (HALL; DAVIS; HUTCHINGS, 2012, p. 22). O modelo Dexter propositalmente não elabora esta camada (HALASZ; SCHWARTZ, 1994, p. 32), pois seria irrealista tentar criar um modelo genérico que cobrisse todos os possíveis tipos de mídia (texto, imagens, simulações etc.) (HALASZ; SCHWARTZ, 1994, p. 32). Assume-se que outros modelos específicos (como os de documentos de texto ou gráficos) serão usados para lidar com este nível (HALASZ; SCHWARTZ, 1994, p. 32).

Figura 6 – A arquitetura do Modelo Dexter.



Fonte: Adaptado de (HALL; DAVIS; HUTCHINGS, 2012).

- **Camada de Execução:** Esta camada descreve os mecanismos que "suportam a interação do usuário com o hipertexto"(HALASZ; SCHWARTZ, 1994, p. 30), (HALL; DAVIS; HUTCHINGS, 2012, p. 22). Ela captura os "aspectos dinâmicos e interacionais"(HALASZ; SCHWARTZ, 1994, p. 32), ou seja, as ferramentas para acessar, visualizar e manipular a rede de componentes (HALASZ; SCHWARTZ, 1994, p. 32). Assim como a camada interna, o modelo Dexter fornece apenas uma definição básica de seus mecanismos (HALASZ; SCHWARTZ, 1994, p. 32).

O aspecto mais crucial do Modelo Dexter são as duas interfaces que conectam a Camada de Armazenamento (onde a rede de links é definida) às outras duas camadas (HALASZ; SCHWARTZ, 1994, p. 30, 32):

1. **Ancoragem:** Esta é a interface entre a Camada de Armazenamento e a Camada Interna do Componente (HALASZ; SCHWARTZ, 1994, p. 30, 32), (HALL; DAVIS; HUTCHINGS, 2012, p. 23). A ancoragem é o mecanismo que permite ao sistema (na camada de armazenamento) "endereçar (referir-se) a locais ou itens dentro do conteúdo"(HALASZ; SCHWARTZ, 1994,

p. 32). Isso permite, por exemplo, que um link aponte não apenas para um documento inteiro, mas para um "trecho de caracteres"(span-to-span) específico dentro dele, mantendo a separação entre o link (armazenamento) e o conteúdo (interno) (HALASZ; SCHWARTZ, 1994, p. 32).

2. **Especificações de Apresentação:** Esta é a interface entre a Camada de Armazenamento e a Camada de Execução (HALASZ; SCHWARTZ, 1994, p. 30, 32), (HALL; DAVIS; HUTCHINGS, 2012, p. 23). É um mecanismo que permite que informações sobre como um componente deve ser apresentado ao usuário sejam codificadas na própria rede de hipertexto (HALASZ; SCHWARTZ, 1994, p. 32). Por exemplo, um link pode conter uma especificação de apresentação que instrui a camada de execução a "executar a animação"de destino, enquanto outro link para o mesmo componente pode instruir a "visualizar no editor"(HALASZ; SCHWARTZ, 1994, p. 32).

2.3.2 AHAM

Enquanto o Modelo Dexter fornece uma arquitetura formal para a hipermídia em geral, ele não aborda nativamente a personalização ou adaptação. Para preencher essa lacuna, De Bra, Houben e Wu propuseram o AHAM, um modelo de referência projetado especificamente para Sistemas de Hipermídia Adaptativos (AHS) (BRA; HOUBEN; WU, 1999; WU *et al.*, 2000).

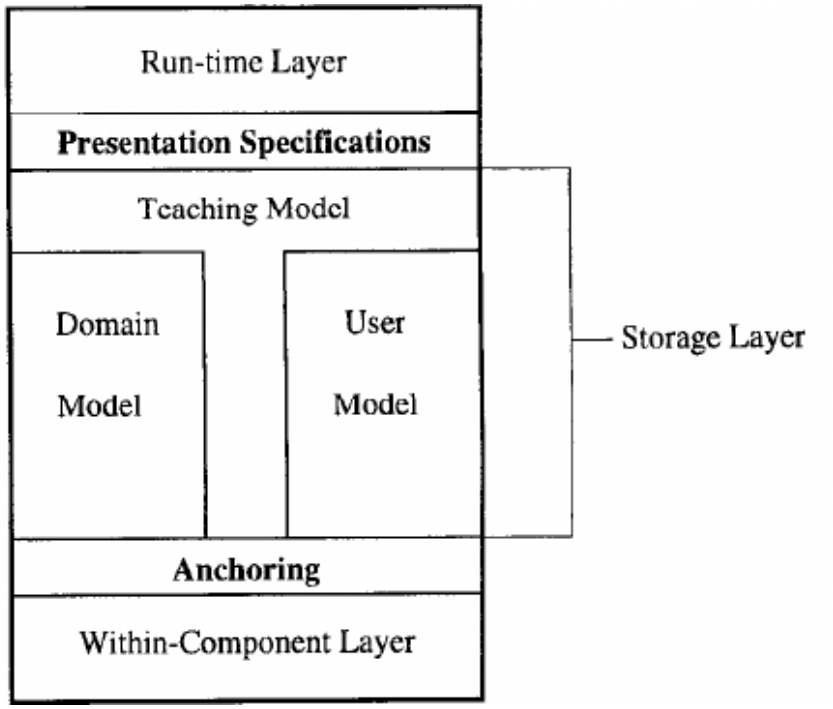
O AHAM é explicitamente "baseado no modelo Dexter"(BRA; HOUBEN; WU, 1999, p. 1) e o estende para incluir formalmente os mecanismos de adaptação (WU *et al.*, 2000, p. 2). A principal inovação do AHAM é sua definição da Camada de Armazenamento (Storage Layer) do Dexter. O AHAM "aumenta o Dexter com recursos para fazer adaptação"(BRA; HOUBEN; WU, 1999, p. 1), dividindo a Camada de Armazenamento em três sub-modelos explícitos, conforme ilustrado na Figura 7: o Modelo de Domínio, o Modelo de Usuário e o Modelo de Adaptação (ou Ensino) (BRA; HOUBEN; WU, 1999, p. 3).

Esta "divisão em um modelo de domínio (DM), modelo de usuário (UM) e modelo de adaptação (AM) fornece uma clara separação de interesses"(WU *et al.*, 2000, p. 2), que é crucial para o desenvolvimento e a autoria de um sistema adaptativo.

Uma aplicação de hipermídia adaptativa completa, segundo este modelo, é uma 4-tupla composta por: $< DM, UM, AM, AE >$ (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 6, 8).

a) **Modelo de Domínio:** Representa a visão do autor sobre o domínio da aplicação (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 2, 3). Ele descreve como o conteúdo da informação é estruturado (BRA; HOUBEN; WU, 1999, p. 2), geralmente em termos de "conceitos e relações conceituais"(como links ou pré-requisitos) (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 3, 4). Os conceitos podem ser atômicos (fragmentos de informação) ou compostos (páginas que agrupam fragmentos) (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 3, 4).

Figura 7 – A arquitetura do Modelo AHAM.



Fonte: Adaptado de (BRA; HOUBEN; WU, 1999)

b) **Modelo de Usuário:** Este componente compartilha o papel central com o DM (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 2, 3). É um "registro permanente e continuamente atualizado"(BRA; HOUBEN; WU, 1999, p. 2) que representa "aspectos relevantes do usuário, como preferências, conhecimento e interesses"(WU *et al.*, 2000, p. 2). Essencialmente, o UM representa "como o usuário se relaciona com o modelo de domínio"(BRA; HOUBEN; WU, 1999, p. 3), monitorando, por exemplo, "quanto o usuário sabe sobre cada um dos conceitos"(BRA; HOUBEN; WU, 1999, p. 3).

c) **Modelo de Adaptação:** Também chamado de Modelo de Ensino (Teaching Model), este modelo especifica como a adaptação deve ocorrer (BRA; HOUBEN; WU, 1999, p. 3). Ele "consiste em regras de adaptação"(WU *et al.*, 2000, p. 2) (ou "regras pedagógicas"(BRA; HOUBEN; WU, 1999, p. 1)) que definem como o sistema deve "combinar o modelo de domínio e o modelo de usuário para realizar a adaptação real"(BRA; HOUBEN; WU, 1999, p. 2).

d) **Motor de Adaptação:** Este é o "ambiente de software"(o componente de sistema) que "executa as regras"(WU *et al.*, 2000, p. 2, 5) definidas no Modelo de Adaptação. O AE é o mecanismo que "realiza a adaptação real"(BRA; HOUBEN; WU, 1999, p. 2), como atualizar o modelo do usuário ou gerar as especificações de apresentação (por exemplo, decidir mostrar ou ocultar um fragmento) (WU *et al.*, 2000; BRA; HOUBEN; WU, 1999, p. 2, 5-6).

2.3.3 Munique

O Modelo de Munique é outra abordagem para formalizar a arquitetura de sistemas de hipermídia adaptativos, apresentada por Koch e Wirsing (KOCH; WIRSING, 2002). Assim como o AHAM, ele é definido como uma "extensão do Modelo Dexter"(KOCH; WIRSING, 2002, p. 1) e preserva a sua estrutura de três camadas (Execução, Armazenamento e Interna do Componente), mas "estende a funcionalidade de cada camada para incluir os aspectos de modelagem do usuário e adaptação"(KOCH; WIRSING, 2002, p. 2).

A principal novidade do Modelo de Munique é a sua abordagem e foco. Enquanto o AHAM adota uma perspectiva mais próxima de banco de dados, o Modelo de Munique adota um "ponto de vista de engenharia de software orientado a objetos"(KOCH; WIRSING, 2002, p. 2):

1. Uma "representação visual intuitiva"(KOCH; WIRSING, 2002, p. 1) através da UML, que permite visualizar rapidamente os conceitos, sua organização e seus relacionamentos (KOCH; WIRSING, 2002, p. 2).

2. Uma "especificação formal inequívoca"(KOCH; WIRSING, 2002, p. 1) usando a OCL, que suplementa os diagramas gráficos com semântica formal e restrições (KOCH; WIRSING, 2002, p. 2).

A arquitetura do Modelo de Munique é "similar à arquitetura do modelo de referência AHAM"(KOCH; WIRSING, 2002, p. 2). A Camada de Armazenamento é dividida em três sub-modelos (KOCH; WIRSING, 2002, p. 3), conforme ilustrado na Figura 8.

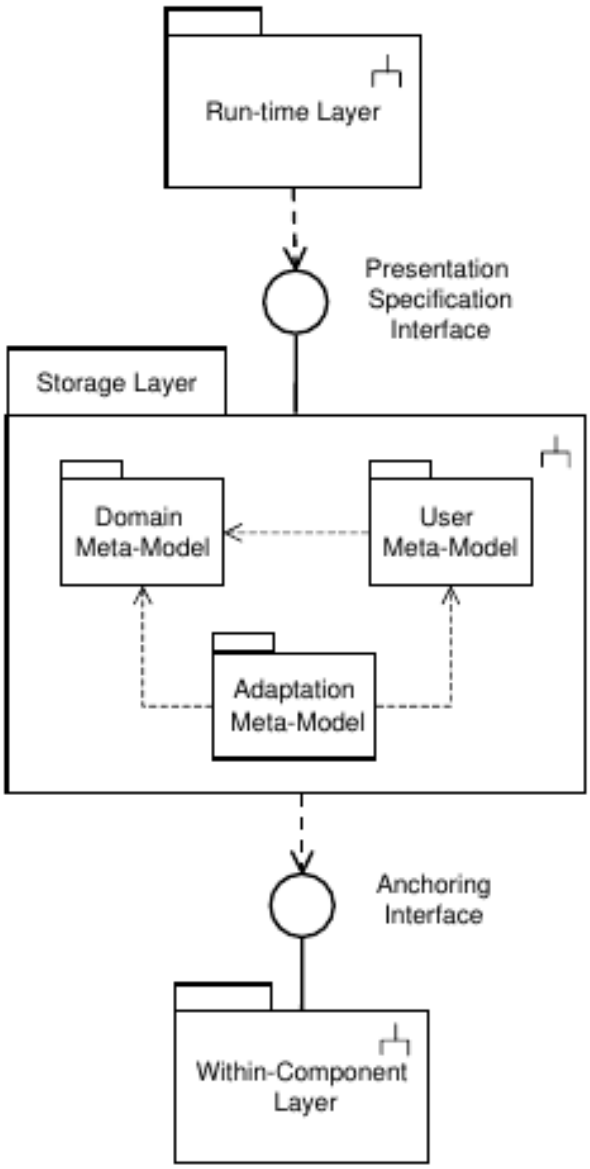
Este modelo de referência formal serve como a base para a abordagem de engenharia de software conhecida como UWE (KOCH; WIRSING, 2001, p. 2). A UWE é a metodologia de design baseado em modelo (KOCH; MANDEL, 1999, p. 2) proposta pelos mesmos autores para o desenvolvimento de sistemas de hipermídia, incluindo os adaptativos (KOCH; WIRSING, 2001, p. 1).

Uma característica central da UWE, e uma prática recomendada no desenvolvimento de hipermídia, é a "separação da análise de domínio da especificação da estrutura de navegação, bem como do design da interface do usuário"(KOCH, 1999, p. 2). Esta separação de interesses (conteúdo, navegação e apresentação (KOCH; MANDEL, 1999, p. 2)) é uma "característica comum"das metodologias de hipermídia robustas (KOCH, 1999, p. 2) e é essencial para gerir a complexidade dos sistemas de hipermídia adaptativa, para os quais as abordagens genéricas de engenharia de software não são suficientes (KOCH; WIRSING, 2001, p. 1).

O processo de design da UWE é, portanto, dividido em três etapas principais que se constroem sobre os sub-modelos do Modelo de Munique (KOCH; MANDEL, 1999, p. 5):

1. **Design Conceitual (e Adaptativo):** Produz um modelo de classes do domínio, dos usuários e da adaptação.

Figura 8 – A arquitetura do Modelo Munique.



Fonte: (KOCH; WIRSING, 2002)

2. **Design de Navegação:** Define a estrutura de navegação da hipermídia como uma visão sobre o modelo conceitual.

3. **Design de Apresentação:** Modela a interface de usuário estática e dinâmica.

Toda esta metodologia é "inteiramente baseada UML"(KOCH; MANDEL, 1999, p. 2), utilizando os mecanismos de extensão da UML para modelar não apenas o conteúdo, mas também a navegação e a apresentação (KOCH; MANDEL, 1999, p. 10).

2.3.4 AHAM-MI

Proposto por Bugay (2006), é um modelo de referência focado especificamente em sistemas educacionais (BUGAY *et al.*, 2006; TAKIKAWA, 2010).

Este modelo é descrito como um "modelo híbrido 'AHAM-Munich'"(BUGAY *et al.*, 2006), pois utiliza conceitos de ambos os predecessores. Ele adota a estrutura de três camadas (Execução, Armazenamento e Interna de Componentes) e a decomposição da Camada de Armazenamento nos três sub-modelos: Modelo de Domínio, Modelo de Usuário e Modelo de Adaptação (BUGAY *et al.*, 2006; TAKIKAWA, 2010).

A principal inovação do AHAM-MI é a inclusão explícita da teoria das Inteligências Múltiplas de Howard Gardner no núcleo da adaptação (BUGAY *et al.*, 2006; TAKIKAWA, 2010). Isso modifica os modelos de duas maneiras principais:

1. **No Modelo do Usuário:** Além de armazenar o conhecimento do aluno sobre o domínio, o UM no AHAM-MI é estendido para armazenar "o nível de desenvolvimento de cada uma de suas inteligências"(ex: Lingüística, Lógico-matemática, Espacial, etc.) (BUGAY *et al.*, 2006).

2. **No Modelo de Adaptação:** O modelo de adaptação (ou "ensino") utiliza esta informação. Ele "seleciona o conteúdo a ser apresentado levando em conta o conhecimento do usuário... e utilizando também o desenvolvimento das suas inteligências... para influir na adaptação"(BUGAY *et al.*, 2006). O sistema usa a inteligência mais evidenciada do aluno (ex: "visual ou auditiva"(TAKIKAWA, 2010)) para selecionar um "cenário"de apresentação de conteúdo que seja mais adequado ao seu perfil (BUGAY *et al.*, 2006).

2.3.5 SHASIM

O modelo SHASIM, proposto por Puga (2008), é um Sistema Hipermídia Adaptativo projetado especificamente para o contexto da "Educação Baseada em Web"(PUGA, 2008).

A principal característica deste modelo é o seu critério de adaptação: ele seleciona "conteúdos apropriados ao estilo cognitivo do aluno"(PUGA, 2008). Para identificar este estilo, o modelo investiga as "Inteligências Múltiplas"do usuário. A partir da combinação das inteligências mais desenvolvidas, o sistema classifica o aluno em uma de 10 "personas-cognitivas"pré-definidas (PUGA, 2008).

O SHASIM utiliza como base de referência o Modelo Munich, mas o "adequou à proposta educacional e ao estudo dos signos"(PUGA, 2008). A arquitetura do SHASIM é estruturada da seguinte maneira:

- **Camada da Sessão:** Equivalente à camada de execução (Run-Time Layer), responsável pela interação com o usuário (PUGA, 2008).

- **Camada de Armazenamento:** O núcleo do sistema, dividido nos seguintes subsistemas:
  - **Subsistema Modelo do Usuário:** Armazena os dados do aluno, incluindo seu progresso, preferências e os dados da sondagem inicial que definem suas Inteligências Múltiplas (PUGA, 2008).
  - **Subsistema Modelo do Domínio:** Estrutura o conteúdo educacional (ex: cursos, planos de estudo) (PUGA, 2008).
  - **Subsistema Modelo de Adaptação:** Contém as regras que utilizam o Modelo do Usuário para adaptar o Domínio (PUGA, 2008).
  - **Subsistema Modelo dos Signos:** Uma das principais adições do SHASIM. Este módulo é responsável pela "manutenção, seleção e composição dos signos"(instrucionais e de interação) que serão apresentados ao aluno, com base na sua persona-cognitiva (PUGA, 2008).
- **Camada dos Componentes:** Corresponde à camada interna do componente (Within-Component Layer) do Dexter, contendo os objetos de aprendizagem em si (PUGA, 2008).

2.4 SÍNTESE COMPARATIVA DOS MODELOS

Os cinco modelos apresentados ao longo deste capítulo compartilham uma base conceitual comum — a arquitetura de três camadas do Modelo Dexter —, mas diferem significativamente em seus objetivos, mecanismos de adaptação e contextos de aplicação. O Quadro 1 sintetiza as principais dimensões de comparação entre eles.

A análise comparativa evidencia uma trajetória de evolução dos modelos. O Dexter estabelece a estrutura formal de três camadas (Execução, Armazenamento e Interna do Componente), mas não contempla adaptação (HALASZ; SCHWARTZ, 1994). O AHAM e o Modelo de Munique surgem como extensões diretas do Dexter, incorporando formalmente o tripé DM–UM–AM à Camada de Armazenamento (BRA; HOUBEN; WU, 1999; KOCH; WIRSING, 2002); a diferença central entre ambos reside na notação: o AHAM privilegia a precisão matemática, enquanto o Munique adota UML e OCL para aproximar o modelo das práticas de engenharia de software, servindo de base para a metodologia UWE (KOCH; WIRSING, 2002).

Os modelos AHAM-MI e SHASIM representam especializações orientadas ao contexto educacional. Ambos incorporam a teoria das Inteligências Múltiplas de Gardner (BUGAY *et al.*, 2006; PUGA, 2008), porém com enfoques distintos: o AHAM-MI estende o Modelo do Usuário para registrar o nível de cada inteligência e utiliza esse dado diretamente no Modelo de Adaptação; o SHASIM vai além ao introduzir o conceito de *persona cognitiva* e um subsistema específico para a gestão de signos instrucionais, reforçando a dimensão semiótica da apresentação adaptativa.

Quadro 1 – Comparativo entre os modelos de sistemas hipermídia adaptativos.

Dimensão	**Dexter**	**AHAM**	**Munique**	**AHAM-MI**	**SHASIM**
Autores / Ano	Halasz e Schwartz (1994)	De Bra, Houben e Wu (1999)	Koch e Wirsing (2002)	Bugay (2006)	Puga (2008)
Modelo base	—	Dexter	Dexter	AHAM + Munique	Munique
Foco principal	Estrutura formal do hipertexto	Adaptação em AHS de uso geral	Engenharia de software orientada a objetos	Adaptação educacional com Inteligências Múltiplas	Adaptação pelo estilo cognitivo do aluno
Critério de adaptação	Não possui (estático)	Conhecimento, preferências e interesses do usuário	Conhecimento, preferências e interesses do usuário	Conhecimento e Inteligências Múltiplas (Gardner)	Personas cognitivas derivadas das Inteligências Múltiplas
Modelo de Usuário	Não possui	Sim (UM)	Sim (UM)	Sim, estendido com nível de cada inteligência	Sim, inclui sondagem de IM e persona cognitiva
Modelo de Adaptação	Não possui	Sim (AM) — regras pedagógicas	Sim (AM) — regras em OCL	Sim — seleciona cenário de apresentação	Sim — seleciona signos instrucionais por persona
Notação/ Representação	Formal/ matemática	Formal/ matemática	UML + OCL	UML (herdada do Munique)	Não especificado formalmente
Contexto educacional	Não específico	Não específico (uso geral)	Não específico (uso geral)	Sim — sistemas educacionais	Sim — Educação Baseada em *Web*

Fonte: Elaborado pelo autor.

Diante desse panorama, o presente trabalho adota a metodologia UWE, fundamentada no Modelo de Munique, por sua aderência às práticas de engenharia de software orientada a objetos, pela notação UML que facilita a comunicação e validação dos modelos produzidos, e pela separação explícita entre os modelos conceitual, de navegação e de apresentação — princípio essencial para o gerenciamento da complexidade de um sistema hipermídia adaptativo (KOCH, 1999; KOCH; WIRSING, 2001).

3 A LÍNGUA LATINA

A língua latina foi o idioma falado pelos *Latini*, os habitantes do Lácio (*Latium*), uma região da Itália central que incluía a cidade de Roma (*Roma*) (ORBERG, 1999). Originalmente apenas uma das muitas línguas itálicas – coexistindo com o etrusco, o osco e o úmbrico (LOURENÇO, 2021) – o latim expandiu-se agressivamente junto com o domínio de Roma, o *imperium Romanum* (ORBERG, 1999).

Esse imperialismo romano foi também um "imperialismo linguístico"(LOURENÇO, 2021). O latim gradualmente suplantou e "exterminou"as outras línguas da península (LOURENÇO, 2021) e, nos séculos seguintes, difundiu-se pela maior parte da Europa Ocidental, Norte da África e Oriente Próximo (ORBERG, 1999). No seu auge, o império deixou mais de 130.000 inscrições em latim, de Lisboa a Damasco (LOURENÇO, 2021).

A única língua que o latim não conseguiu nem quis apagar foi o grego (LOURENÇO, 2021). O grego manteve sua posição dominante no Mediterrâneo Oriental, e as classes cultas romanas eram frequentemente bilíngues, criando um mundo com duas línguas globais: o grego e o latim (ORBERG, 1999; LOURENÇO, 2021).

Após a queda do Império Ocidental, o latim falado nas províncias (o "latim vulgar") não desapareceu, mas evoluiu gradualmente para as línguas românicas modernas: italiano, francês, espanhol, romeno e o português (ORBERG, 1999; LOURENÇO, 2021).

Embora o latim seja hoje frequentemente chamado de língua "morta", visto que não é a língua materna de nenhum povo, o termo é considerado enganoso (ORBERG, 1999). O latim nunca acabou de fato (LOURENÇO, 2021). Graças à sua adoção pela Igreja Romana e pelas classes educadas, o latim manteve uma vitalidade extraordinária, permanecendo "inabalável como a língua comum das classes educadas da Europa"durante toda a Idade Média e como o principal veículo da "erudição internacional até o século XVIII"(ORBERG, 1999).

O rastro linguístico do latim continua a crescer através das suas línguas descendentes, faladas hoje por centenas de milhões de pessoas (LOURENÇO, 2021). Para um falante de português – que é, em essência, "uma forma de latim"(LOURENÇO, 2021) –, aprender a gramática latina não é apenas um exercício acadêmico, mas o ato de "pisar a ponte mental"que leva da nossa língua para a "explosão de nitidez que é o latim propriamente dito"(LOURENÇO, 2021). Como o poeta Vergílio escreveu, estudar latim é "entrar numa floresta antiga"(LOURENÇO, 2021).

3.1 PARTICULARIDADES GRAMATICAIS

Para compreender a lógica de um sistema de ensino de Latim, é fundamental entender a estrutura da própria língua. De forma moderna, a gramática é definida como a disciplina que

"expõe... as leis que regem o idioma"(ALMEIDA, 2011, p. 15), tradicionalmente dividida em Fonética, Morfologia e Sintaxe.

É no contraste dessas áreas com o português que a complexidade do aprendizado se revela.

3.1.1 Fonética

A Fonética "trata dos sons da linguagem articulada"(ALMEIDA, 2011, p. 15). A principal diferença fonética entre as duas línguas é que o latim clássico possui vogais quantitativas (longas e breves), onde a duração do som da vogal podia alterar o significado de uma palavra (ex: *mălum* "o mal"vs. *mālum* "a maçã") e era crucial para a métrica poética (ALMEIDA, 2011). O português perdeu completamente essa distinção quantitativa. A prosódia portuguesa baseia-se na tonicidade (sílaba tônica) e no timbre (vogais abertas e fechadas), características que eram, em sua maioria, irrelevantes ou secundárias no latim clássico.

3.1.2 Morfologia

A Morfologia estuda a "forma das palavras"(ALMEIDA, 2011, p. 15). Esta é, sem dúvida, a diferença mais profunda e o maior desafio para o falante de português.

O latim é uma língua altamente sintética ou flexional. Isso significa que a função gramatical de uma palavra (se ela é sujeito, objeto direto, etc.) é indicada por suas terminações, chamadas casos (Nominativo, Genitivo, Dativo, Acusativo, Ablativo e Vocativo) (ERNOUT; THOMAS, 2008; ALMEIDA, 2011).

O português, por outro lado, é uma língua analítica. Ele perdeu o sistema de casos e expressa as funções gramaticais através de duas ferramentas principais: o uso de preposições ("de", "para", "com", "a") e a ordem fixa das palavras na frase (ALMEIDA, 2011).

3.1.3 Sintaxe

A Sintaxe, que trata da "relação lógica das palavras na oração"(ALMEIDA, 2011, p. 15) e da "estrutura da frase"(ERNOUT; THOMAS, 2008, p. V), é uma consequência direta da morfologia.

Como a morfologia do latim (os casos) já define a função da palavra, a ordem das palavras é extremamente flexível. A ordem Sujeito-Verbo-Objeto (SVO) é comum na prosa, mas pode ser livremente alterada para ênfase, clareza ou estilo (ex: SOV, VSO) sem alterar o significado fundamental da frase (ERNOUT; THOMAS, 2008, p. 1).

No português, o oposto é verdadeiro. Como a morfologia não indica a função, a sintaxe depende rigidamente da ordem das palavras. A estrutura SVO (Sujeito-Verbo-Objeto) é a base da sintaxe portuguesa.

3.2 O ENSINO

A história do ensino de latim no Brasil é longa e marcada por profundas mudanças filosóficas e legais, que alteraram tanto seus objetivos quanto seus métodos (LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012; HECK, 2013). O ensino da língua latina no país pode ser dividido em três grandes fases: a jesuítica, a pombalina e o período pós-LDB de 1961, que culminou nos debates metodológicos atuais (LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012).

3.2.1 O Período Jesuítico (1549-1759)

A história do ensino de latim no Brasil tem início com a chegada dos jesuítas em 1549 (LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012). A educação jesuítica, baseada no plano de estudos *Ratio Studiorum*, possuía uma concepção pedagógica "tradicional religiosa"(FERNANDES *et al.*, 2012) (ou humanista-cristã) (HECK, 2013).

Neste modelo, o latim era a disciplina central e primordial (LEITE; BARBOSA *et al.*, 2014). Mais do que uma matéria, era o "veículo de informação e estudo"para a maior parte das outras disciplinas (LEITE; BARBOSA *et al.*, 2014). A metodologia consistia na leitura, análise e imitação de autores clássicos (especialmente Cícero).

3.2.2 As Reformas Pombalinas (A partir de 1759)

A hegemonia jesuíta terminou em 1759 com sua expulsão por ordem do Marquês de Pombal, que instituiu o sistema de "aulas régias"(LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012). As reformas pombalinas foram guiadas pelas ideias iluministas de Luís António Verney, expressas em sua obra Verdadeiro método de estudar (1746) (LEITE; BARBOSA *et al.*, 2014).

Verney criticava o método jesuíta por focar excessivamente na "memorização de regras gramaticais", dedicando pouco tempo à leitura e a exercícios que promovessem o domínio ativo do idioma (LEITE; BARBOSA *et al.*, 2014). Paradoxalmente, a reforma pombalina, ao proibir que se falasse latim nas aulas e ao tratá-lo como "língua morta"em oposição à "língua viva"usada pelos jesuítas, pode ter solidificado a abordagem puramente gramatical que, na teoria, combatia (LEITE; BARBOSA *et al.*, 2014). Esta tensão "entre o estudo centrado em processos gramaticais aos métodos de leitura"(LEITE; BARBOSA *et al.*, 2014) definiria o ensino da língua pelos séculos seguintes.

3.2.3 O Século XIX

Com a chegada da família real ao Brasil (início do século XIX), o foco da educação mudou para a criação de cursos profissionalizantes e técnicos, como academias militares e escolas

de medicina (LEITE; BARBOSA *et al.*, 2014). Esses novos currículos "negligenciavam o antes onipresente estudo das humanidades", relegando o latim a um nicho.

Isso deu início ao que ficou conhecido como a "batalha pelo humanismo"no século XX, um debate entre educadores que defendiam uma formação pragmática e aqueles que defendiam a manutenção do currículo humanista (incluindo o latim) como essencial para a "formação geral sem preocupação com a especialização"(LEITE; BARBOSA *et al.*, 2014).

3.2.4 O Método Gramática-Tradução no Brasil

Durante a maior parte dos séculos XIX e XX, a metodologia predominante no Brasil foi a abordagem da gramática e tradução (FERNANDES *et al.*, 2012; LEITE; BARBOSA *et al.*, 2014). Esta é a abordagem mais antiga e que "ainda se faz presente no ensino do latim"(FERNANDES *et al.*, 2012).

Nesta abordagem, o ensino da segunda língua se dá pela primeira, com explicações gramaticais na língua materna (FERNANDES *et al.*, 2012). O objetivo principal não é a comunicação (fala ou audição), mas sim a "apreciação da cultura e da literatura da segunda língua"(FERNANDES *et al.*, 2012) e o desenvolvimento intelectual (FERNANDES *et al.*, 2012; ALMEIDA, 2011).

O principal expoente brasileiro dessa visão foi Napoleão Mendes de Almeida, autor da clássica *Gramática Latina* (ALMEIDA, 2011). Em seu prefácio, Almeida defende que a "verdadeira importância do latim"não é auxiliar no aprendizado do português (ALMEIDA, 2011), mas sim "desenvolver a inteligência", "aguçar o intelecto"e "desenvolver o espírito de análise, de observação, de raciocínio".

Neste método, a tradução de textos literários é vista como uma "preocupação nefasta"se introduzida prematuramente. O aluno deve primeiro dominar exaustivamente a morfologia e a sintaxe, tanto do português quanto do latim, antes de se aventurar nos autores clássicos.

3.2.5 Declínio Legal e a Mudança para Métodos de Leitura

O latim teve seu "último reduto"no currículo obrigatório com a reforma Capanema, em 1942, que o manteve em todas as séries do curso ginasial (LEITE; BARBOSA *et al.*, 2014; HECK, 2013).

Contudo, o "golpe definitivo"(LEITE; BARBOSA *et al.*, 2014) veio com a Lei de Diretrizes e Bases da Educação (LDB) de 1961 (Lei 4.024/61), que tornou o latim uma disciplina "optativa"(facultativa) (LEITE; BARBOSA *et al.*, 2014; FERNANDES *et al.*, 2012; HECK, 2013). Isso levou ao seu quase completo desaparecimento do ensino secundário e o "entrincheiramento"da disciplina em poucos cursos universitários de Letras (LEITE; BARBOSA *et al.*, 2014; HECK, 2013).

Hoje, o latim ocupa um "espaço mínimo nos currículos"(LEITE; BARBOSA *et al.*, 2014).

Com a drástica redução na carga horária, os objetivos mudaram (LEITE; BARBOSA *et al.*, 2014). O debate não é mais sobre o "ornamento do espírito"ou apenas sobre a estrutura gramatical (HECK, 2013). O consenso atual é que o objetivo principal é o "desenvolvimento da habilidade de leitura de textos originais"e a compreensão da cultura romana (LEITE; BARBOSA *et al.*, 2014).

Isso levou à adoção de métodos modernos focados na leitura. Destacam-se o Aprendendo Latim (JONES; SIDWELL, 2012; FERNANDES *et al.*, 2012) e o Lingua Latina per se illustrata (ORBERG, 2011). O primeiro método é considerado o "mais eficiente"para o objetivo de "dar acesso aos textos latinos no original, de forma gradual e sólida"(JONES; SIDWELL, 2012). Já o segundo, de Hans Ørberg, utiliza "a forma natural de aprender uma língua pelo método direto"(HECK, 2013), onde o aluno aprende a língua "a partir de textos", com o léxico e as estruturas gramaticais explicadas "à margem dos textos"e a memorização facilitada pela "recorrência do vocabulário"(HECK, 2013, p. 24). Ao contrário do método gramática-tradução, estas abordagens não dissociam a língua da cultura e, em vez de frases soltas, apresentam "textos inseridos em seu contexto histórico-cultural"desde as primeiras lições (JONES; SIDWELL, 2012).

3.3 CONSIDERAÇÕES SOBRE O ENSINO E A SOLUÇÃO PROPOSTA

O percurso histórico da língua latina, desde sua expansão imperial até seu "entrincheiramento"(LEITE; BARBOSA *et al.*, 2014) nos currículos modernos, revela uma tensão pedagógica fundamental que persiste até hoje.

De um lado, o método clássico de Gramática-Tradução, defendido no Brasil por expoentes como Napoleão Mendes de Almeida, que vê o latim como uma ferramenta para "desenvolver a inteligência"e o "espírito de análise"(ALMEIDA, 2011). Esta abordagem exige o domínio exaustivo da estrutura gramatical como pré-requisito para a leitura de textos (FERNANDES *et al.*, 2012; ALMEIDA, 2011). De outro lado, os métodos modernos focados na leitura, como o *Aprendendo Latim* (JONES; SIDWELL, 2012) ou o *Lingua Latina per se illustrata* (ORBERG, 2011), que surgiram da necessidade de focar na "habilidade de leitura de textos originais"(LEITE; BARBOSA *et al.*, 2014) no "espaço mínimo"(LEITE; BARBOSA *et al.*, 2014) que a disciplina ocupa atualmente.

O método de Gramática-Tradução, embora academicamente rigoroso, impõe uma imensa barreira de entrada, exigindo o domínio de todo o sistema antes de permitir o acesso aos textos literários, uma abordagem impraticável para a carga horária atual (FERNANDES *et al.*, 2012). Por sua vez, os métodos de leitura, embora mais práticos, ainda apresentam uma trajetória linear, *one-size-fits-all*, incapaz de se ajustar às diferentes necessidades e ritmos dos alunos.

É neste ponto que a hipermídia adaptativa, objeto central deste trabalho, oferece uma solução. Em vez de forçar uma escolha pedagógica única entre "primeiro a gramática"ou "primeiro o texto", um sistema adaptativo pode personalizar o caminho para cada aluno. Dessa forma, a hipermídia adaptativa não substitui os métodos, mas os potencializa. Ela permite a criação de um percurso de estudo individualizado que respeita o ritmo do aluno e resolve a tensão

secular entre o ensino rigoroso da estrutura gramatical e a fruição cultural dos textos.

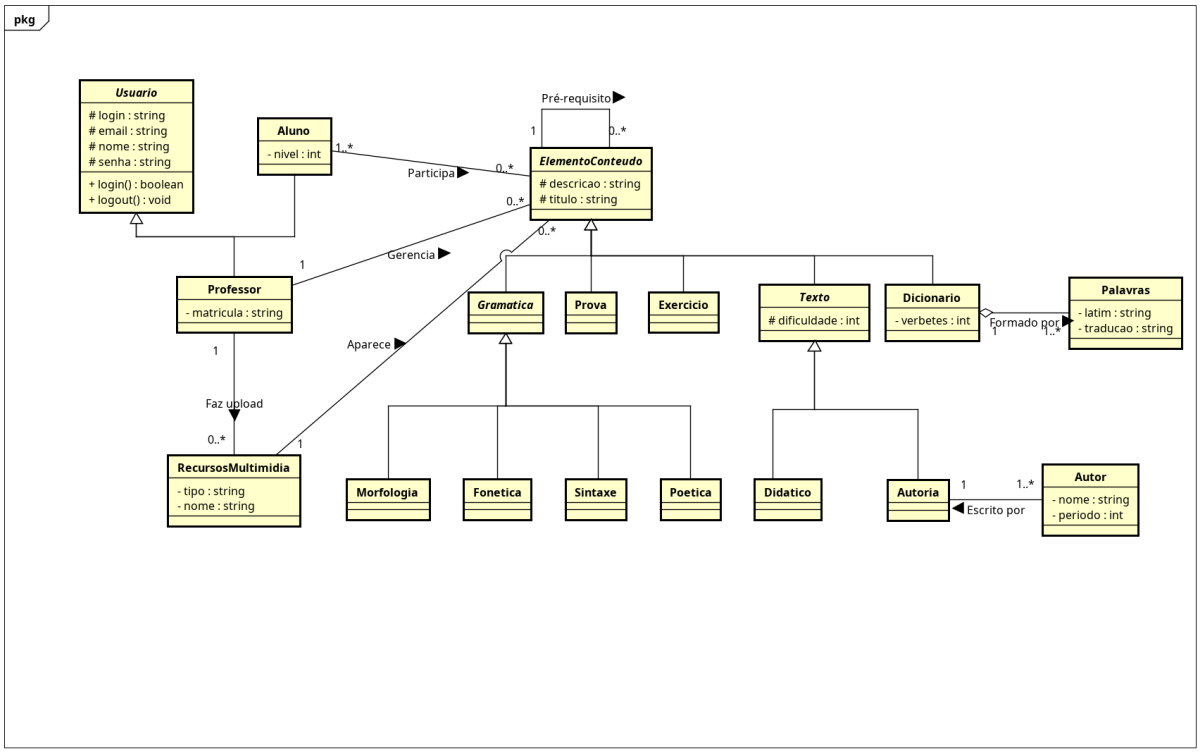
4 LINGUA LATINA ADAPTATIVA

Este capítulo apresenta a modelagem do sistema proposto, que aqui chamaremos de *Lingua Latina Adaptativa*. A arquitetura de software desta solução é fundamentada na metodologia de design baseada em UML proposta por Koch et al. (KOCH; MANDEL, 1999; KOCH; WIRSING, 2001). Esta abordagem é ideal para o nosso projeto, pois "trata conteúdos, estrutura e apresentação nos mesmos termos"(KOCH; MANDEL, 1999, p. 2), permitindo uma separação clara entre as fases de design.

4.1 DESIGN CONCEITUAL E ADAPTATIVO

O primeiro passo é o design conceitual, cujo "objetivo principal é capturar a semântica do domínio"(KOCH; MANDEL, 1999, p. 6). Para um SHA, este modelo é estendido para incluir os "aspectos de modelagem do usuário e mecanismos de adaptação baseados em regras"(KOCH; WIRSING, 2001), bem como os diferentes atores do sistema.

Figura 9 – Diagrama de classes do design conceitual



Fonte: Autor

O diagrama de classes UML apresentado na Figura 9 representa este modelo conceitual. Ele inclui todos os conceitos que são relevantes para a aplicação e para os diferentes grupos de usuários que foram identificados (KOCH; MANDEL, 1999, p. 6). Seguindo a metodologia, as

atividades desta etapa consistem em encontrar classes, especificar atributos e operações, definir estruturas hierárquicas (KOCH; MANDEL, 1999, p. 7), resultando no "modelo estrutural orientado a objetos"(KOCH; MANDEL, 1999, p. 2) do domínio.

No modelo do *Lingua Latina Adaptativa*, a classe `Usuario` funciona como uma generalização para os dois atores identificados: `Professor` e `Aluno`. Esta abordagem permite modelar os atributos e operações comuns (como `login`, `nome`, `senha`) em um único local, enquanto as especializações capturam as particularidades de cada grupo de usuários.

O núcleo do domínio do problema é representado pela classe `ElementoConteudo`. Esta classe age como uma superclasse para todos os tipos de informação que o sistema irá gerenciar, utilizando a técnica de generalização e especialização. Os tipos específicos de conteúdo, como `Gramatica`, `Prova`, `Exercicio`, `Texto` e `Dicionario`, herdam de `ElementoConteudo`. Algumas dessas classes, por sua vez, são também generalizações, como `Gramatica` (que se divide em `Morfologia`, `Fonetica`, etc.) e `Texto` (que se divide em `Didatico` e `Autoria`).

As associações definem as relações entre essas classes. As mais importantes são as que ligam os usuários ao conteúdo: o `Professor` Gerencia o `ElementoConteudo`, e o `Aluno` Participa dele. Outras associações detalham o domínio, como a ligação entre `Autoria` e `Autor`, e `Dicionario` e `Palavras`. O modelo também captura recursos multimídia (`RecursosMultimidia`) que podem ser carregados pelo `Professor` e associados a um `ElementoConteudo`, de forma similar ao exemplo do “Film Assistant” que modela atributos como áudio e vídeo (KOCH; MANDEL, 1999, p. 7).

Crucialmente para um sistema adaptativo, uma associação reflexiva "Pre-requisito"na classe `ElementoConteudo` modela a dependência entre os próprios conteúdos, uma semântica de domínio vital para a futura modelagem da navegação.

4.1.1 Modelo de Adaptação

O design conceitual de um SHA, conforme a metodologia UWE, deve ser estendido para incluir os "aspectos de modelagem do usuário e mecanismos de adaptação baseados em regras"(KOCH; WIRSING, 2001). Esse conjunto de regras constitui o Modelo de Adaptação(MA), que especifica como o sistema deve combinar o modelo de domínio e o modelo de usuário para "realizar a adaptação real"(BRA; HOUBEN; WU, 1999, p. 2). No Lingua Latina Adaptativa, o MA é formalizado por meio da *Object Constraint Language* (OCL).

A OCL é uma linguagem declarativa padronizada pela *Object Management Group* (OMG) (Object Management Group, 2014), projetada como complemento formal à UML. Ela permite expressar restrições (*invariants*), pré e pós-condições de operações e expressões de consulta sobre modelos de objetos de forma precisa e não-ambígua. Uma propriedade fundamental da OCL é a ausência de efeitos colaterais: a avaliação de uma expressão OCL nunca altera o estado

do modelo (Object Management Group, 2014). A escolha da OCL para a formalização do MA justifica-se por três razões: (i) integração nativa com os diagramas UML já adotados nas demais etapas da metodologia UWE; (ii) capacidade de expressar, em um único formalismo, regras de pré-requisito, invariantes de consistência e pós-condições de operação; e (iii) independência de plataforma de implementação, adequada ao nível de modelagem deste trabalho.

4.1.1.1 Técnicas de Adaptação Empregadas

Com base na fundamentação teórica das Seções 2.2.4.1 e 2.2.4.2, o *Lingua Latina Adaptativa* emprega as técnicas de adaptação sintetizadas no Quadro 2.

Quadro 2 – Técnicas de adaptação empregadas no *Lingua Latina Adaptativa*

Técnica	**Categoria**	**Localização no sistema**
Anotação Adaptativa	Navegação	Links para `ElementoConteudo` mudam de aparência (*Bloqueado*, *Disponível*, *Dominado*) conforme o progresso do `Aluno`; links bloqueados permanecem visíveis porém desabilitados
Orientação Direta	Navegação	O `guidedTour` sugere automaticamente o próximo conteúdo desbloqueado
Texto Condicional / *Stretchtext*	Conteúdo	Explicações gramaticais adicionais são exibidas ou recolhidas conforme o histórico de domínio do `Aluno`

Fonte: Autor

4.1.1.2 Extensão do Modelo de Domínio

Para que as regras OCL possam operar, o modelo conceitual é estendido com uma classe de associação `Progresso` entre `Aluno` e `ElementoConteudo`, responsável por rastrear o estado de cada aluno em cada conteúdo. Os atributos desta classe são: `status` (enumeração `StatusConteudo`: *Bloqueado*, *Disponível*, *Lido*, *Dominado*), `tentativas` e `acertos` (inteiros não-negativos). A partir desta extensão, define-se o alias OCL de conveniência apresentado no Algoritmo 1, que torna explícito no próprio modelo de usuário o conjunto de conteúdos já dominados pelo aluno e simplifica a leitura das demais regras.

Algoritmo 1 – Alias `conteudosDominados` na classe `Aluno`

```
1 context Aluno
2 def: conteudosDominados : Set(ElementoConteudo) =
3   self.progressos
4     ->select(p | p.status = StatusConteudo::Dominado)
5     ->collect(p | p.conteudo)
```

```
6       ->asSet()
```

4.1.1.3 Regras OCL do Modelo de Adaptação

As regras a seguir estão em conformidade com a especificação OCL 2.4 (Object Management Group, 2014). Cada regra é identificada por um código (AM-*n*), acompanhada de sua semântica em linguagem natural e da técnica de adaptação que formaliza.

AM-1 — Acesso por Pré-requisito (Ocultação / Orientação Direta): um `Aluno` só pode acessar um `ElementoConteudo` se todos os seus pré-requisitos já tiverem sido dominados. Para conteúdos sem pré-requisitos, a operação retorna `true` diretamente, pois `forAll` sobre conjunto vazio é verdadeiro em OCL (Object Management Group, 2014). Na interface, um resultado `false` implica que o link recebe a anotação `Anotado_Bloqueado` — permanecendo visível porém desabilitado — e o `guidedTour` não avança (orientação direta). A visibilidade do link é preservada intencionalmente, conforme a decisão de design fundamentada em Brusilovsky (1996) e formalizada em AM-4.

Algoritmo 2 – AM-1: operação `podeAcessar`

```
1  context Aluno
2  def: podeAcessar(c : ElementoConteudo) : Boolean =
3    c.prerequisitos->forAll(pre |
4      self.progressos->exists(p |
5        p.conteudo = pre and p.status = StatusConteudo::Dominado
6      )
7    )
```

AM-2 — Consistência do Status (Invariante): as invariantes abaixo garantem que o atributo `status` de um `Progresso` respeita a progressão lógica dos estados, formalizando as transições do Diagrama de Estado apresentado na Figura 13.

Algoritmo 3 – AM-2: invariantes de consistência de `Progresso`

```
1  context Progresso
2  inv StatusDominadoRequerAcerto:
3    self.status = StatusConteudo::Dominado
4    implies (self.tentativas > 0 and self.acertos >= 1)
5
6  inv StatusLidoRequerDesbloqueio:
7    (self.status = StatusConteudo::Lido
8     or self.status = StatusConteudo::Dominado)
9    implies self.aluno.podeAcessar(self.conteudo)
10
11 inv TentativasNaoNegativas:
12   self.tentativas >= 0 and self.acertos >= 0
13
```

```
14 inv AcertosNaoSupereamTentativas:
15   self.acertos <= self.tentativas
```

AM-3 — Próximo Conteúdo (Orientação Direta): define qual `ElementoConteudo` o `guidedTour trilhaAprendizagemAluno` deve apresentar a seguir — o primeiro, pela ordem pedagógica definida pelo `Professor` (atributo `ordem`), que o aluno pode acessar mas ainda não dominou. A operação emprega `allInstances()`, que a especificação OCL 2.4 classifica como *compliance point* opcional (Object Management Group, 2014); no nível de modelagem deste trabalho, a operação é utilizada com sentido de *extent* da classe, cabendo à implementação traduzí-la para uma consulta indexada ao banco de dados.

Algoritmo 4 – AM-3: operação `proximoConteudo`

```
1  context Aluno
2  def: proximoConteudo() : ElementoConteudo =
3    ElementoConteudo.allInstances()
4      ->select(c |
5          self.podeAcessar(c)
6          and not self.progressos->exists(p |
7                p.conteudo = c
8                and p.status = StatusConteudo::Dominado
9              )
10       )
11     ->sortedBy(c | c.ordem)
12     ->first()
```

AM-4 — Estado do Link na Interface (Anotação Adaptativa): a aparência visual do link para um `ElementoConteudo` é determinada pelo status do progresso do aluno, sem que o link desapareça. Esta decisão de design — anotação em vez de ocultação pura — preserva a visibilidade do mapa cognitivo do aluno, o que está em consonância com Brusilovsky (1996), que aponta a anotação como mais favorável à autonomia do estudante do que a ocultação total.

Algoritmo 5 – AM-4: operação `estadoLink`

```
1  context ElementoConteudo
2  def: estadoLink(aluno : Aluno) : EstadoLink =
3    let prog : Progresso =
4      aluno.progressos->any(p | p.conteudo = self)
5    in
6    if prog.oclIsUndefined()
7       or prog.status = StatusConteudo::Bloqueado then
8      if aluno.podeAcessar(self) then
9        EstadoLink::Anotado_Disponivel
10     else
11       EstadoLink::Anotado_Bloqueado
12     endif
13   else if prog.status = StatusConteudo::Dominado then
```

```
14      EstadoLink::Anotado_Dominado
15    else
16      EstadoLink::Anotado_Disponivel
17    endif endif
```

AM-5 — Texto Condicional (*Stretchtext*): um fragmento de explicação gramatical adicional — por exemplo, uma nota morfológica comparativa para quem já domina outra língua flexional — só é apresentado se o aluno já tiver lido ou dominado o conteúdo pré-requisito associado. Para fragmentos sem pré-requisito (atributo `conteudoRequisito` nulo), o fragmento é sempre visível.

Algoritmo 6 – AM-5: operação `visivelPara` em `FragmentoExplicacao`

```
1  context FragmentoExplicacao
2  def: visivelPara(aluno : Aluno) : Boolean =
3    self.conteudoRequisito.oclIsUndefined()
4    or aluno.progressos->exists(p |
5         p.conteudo = self.conteudoRequisito
6         and (p.status = StatusConteudo::Lido
7              or p.status = StatusConteudo::Dominado)
8       )
```

AM-6 / AM-6b — Atualização do Modelo do Usuário (Ciclo AHAM): as regras AM-6 e AM-6b são complementáres. AM-6 especifica as pós-condições no nível do objeto `Progresso` (o que muda no estado); AM-6b ancora essa atualização na operação concreta `resolver()` da classe `Exercicio`, que detém o `gabarito`. A operação recebe o aluno como parâmetro explícito, pois a especificação OCL 2.4 não define nenhuma variável automática para o objeto invocante em pós-condições (Object Management Group, 2014) — apenas `self`, `result` e os parâmetros nomeados são referenciáveis. Juntas, fecham o ciclo de adaptação descrito no AHAM (BRA; HOUBEN; WU, 1999): a submissão correta move o conteúdo para *Dominado* e desbloqueia os conteúdos dependentes via AM-1; a submissão errada retorna o aluno ao estado *Lido* para revisão, correspondendo às transições `UserSubmitsCorrect` e `UserSubmitsWrong` do Diagrama de Estado (Figura 13).

Algoritmo 7 – AM-6: pós-condições de atualização em `Progresso`

```
1  context Progresso
2  post SubmissaoCorretaDominaConteudo:
3    (self.acertos = self.acertos@pre + 1
4     and self.tentativas = self.tentativas@pre + 1)
5    implies self.status = StatusConteudo::Dominado
6
7  post SubmissaoErradaRetornaLeitura:
8    (self.acertos = self.acertos@pre
9     and self.tentativas = self.tentativas@pre + 1
10    and (self.status@pre = StatusConteudo::Lido
```

```
11          or self.status@pre = StatusConteudo::Disponivel))
12    implies self.status = StatusConteudo::Lido
```

Algoritmo 8 – AM-6b: operação `Exercicio::resolver`

```
1  context Exercicio::resolver(resposta : String, aluno : Aluno)
2  pre RespostaFornecida:
3    resposta <> ''
4
5  post AtualizarProgresso:
6    let prog : Progresso =
7      self.elementoConteudo.progressos->any(p | p.aluno = aluno)
8    in
9    if resposta = self.gabarito then
10     prog.status = StatusConteudo::Dominado
11     and prog.acertos = prog.acertos@pre + 1
12     and prog.tentativas = prog.tentativas@pre + 1
13   else
14     prog.status = StatusConteudo::Lido
15     and prog.tentativas = prog.tentativas@pre + 1
16   endif
```

O Quadro 3 sintetiza o mapeamento entre as técnicas de adaptação empregadas, as regras OCL que as formalizam, o componente AHAM responsável pela sua execução e o objetivo pedagógico reconciliado no *Lingua Latina Adaptativa*.

4.2 DESIGN DE NAVEGAÇÃO

O design de navegação é uma etapa crítica e é construído com base no modelo conceitual definido na seção anterior (KOCH; MANDEL, 1999, p. 5). O seu objetivo é definir a estrutura da aplicação de hipermídia, descrevendo como a navegação pode ocorrer (KOCH; MANDEL, 1999, p. 2).

O resultado desta etapa é um modelo navegacional, que pode ser visto como uma "visão sobre o modelo conceitual"(KOCH; MANDEL, 1999, p. 5). A metodologia UWE divide este design em dois modelos subsequentes: o Modelo de Classes Navegacional e o Modelo de Estrutura Navegacional (KOCH; MANDEL, 1999, p. 8).

4.2.1 Modelo de Classes Navegacional

O primeiro passo é o Modelo de Classes Navegacional. Este modelo define uma visão sobre o modelo conceitual que mostra quais classes do domínio poderão ser visitadas e quais associações tornar-se-ão links (KOCH; MANDEL, 1999, p. 8).

A sua construção envolve duas atividades principais:

Quadro 3 – Mapeamento técnica de adaptação × regra OCL × objetivo pedagógico

Técnica	**Regras**	**Componente AHAM**	**Objetivo pedagógico**
Orientação Direta	AM-1, AM-3	Motor + Modelo de Adaptação	**Gramática-Tradução:** base gramatical sólida antes de avançar
Anotação Adaptativa	AM-1, AM-4	Motor → Interface	**Ambos:** visibilidade do percurso (bloqueado/disponível/dominado) sem sobrecarga cognitiva; links bloqueados ficam desabilitados mas visíveis
Texto Condicional / *Stretchtext*	AM-5	Motor → Interface	**Método de Leitura:** auxílio gramatical pontual e não intrusivo
Ciclo AHAM	AM-2, AM-6, AM-6b	Motor → Modelo de Usuário	**Personalização:** ajusta o sistema ao ritmo individual

Fonte: Autor

1. **Seleção de Classes:** Classes do modelo conceitual que não são relevantes para a navegação (ou seja, não terão uma página própria) são eliminadas ou "reduzidas a atributos de outras classes"(KOCH; MANDEL, 1999, p. 9).

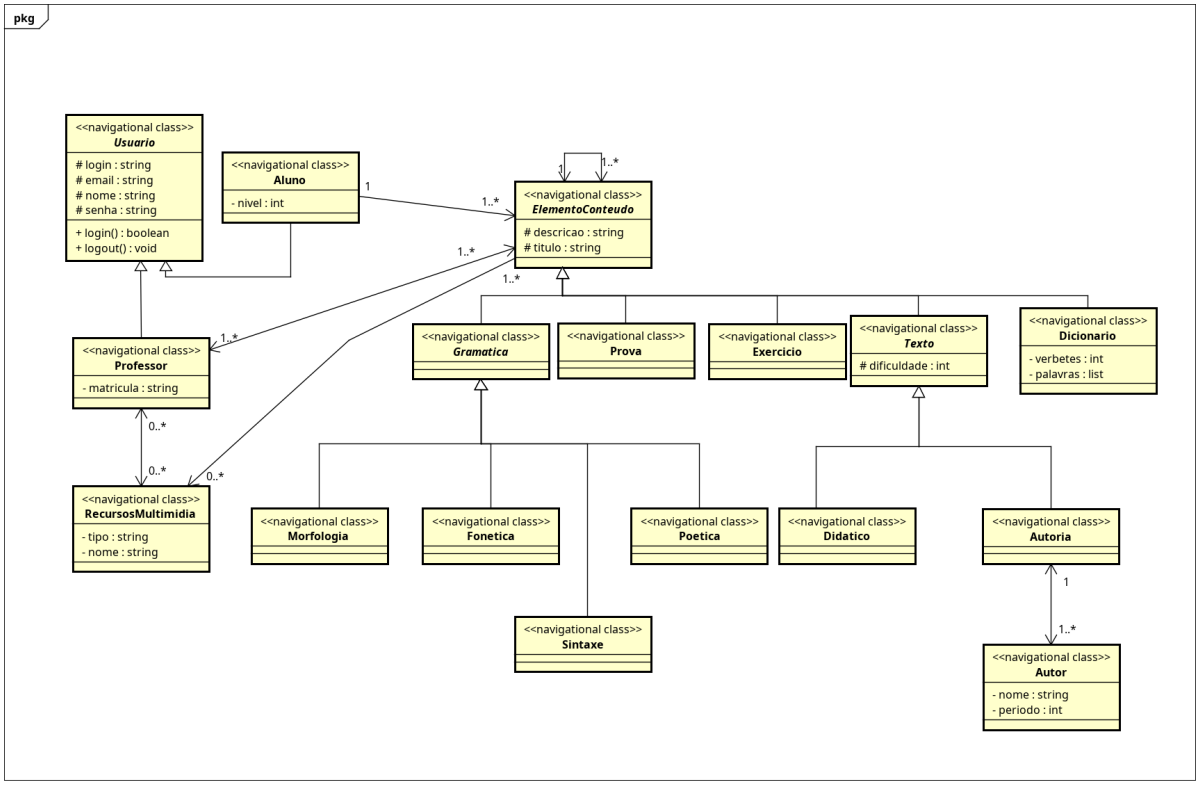
2. **Definição de Navegabilidade:** A "direção da navegação"é especificada para as associações, indicando o fluxo que o usuário pode seguir (KOCH; MANDEL, 1999, p. 9).

Aplicando estes princípios ao Modelo Conceitual do *Lingua Latina Adaptativa*, o Modelo de Classes Navegacional resultante (Figura 10) tem as seguintes características:

- **Classes Mantidas:** Todas as classes que representam destinos de navegação são estereotipadas como «`navigationalClass`» (KOCH; MANDEL, 1999, p. 9). Isso inclui `Aluno`, `Professor`, `Autor`, `RecursosMultimidia`, e todas as subclasses concretas de `ElementoConteudo` (`Morfologia`, `Fonetica`, `Sintaxe`, `Poetica`, `Prova`, `Exercicio`, `Didatico`, `Autoria` e `Dicionario`). As classes abstratas (`Usuario` e `ElementoConteudo`) são mantidas para preservar a hierarquia estrutural e as associações polimórficas.

- **Classes Reduzidas:** A classe `Palavras` foi identificada como não sendo um nó navegável independente. Ela foi incorporada à classe `Dicionario` como um "atributo derivado"(KOCH; MANDEL, 1999, p. 9), com a notação `/palavras: list`. O conte-

Figura 10 – Diagrama do modelo navegacional



Fonte: Autor

údo de `Palavras` (latim, traducao) será, assim, apresentado como parte da página do `Dicionario`, e a associação "Formado por"foi internalizada.

- **Definição de Navegabilidade:** O diagrama especifica a direção exata dos links, substituindo as associações conceituais por setas de navegabilidade. As principais navegações definidas são:
  - **`Aluno` -> `ElementoConteudo`:** A navegação é unidirecional. O aluno navega para os conteúdos em que "Participa", mas o inverso não é permitido para proteger a privacidade dos alunos inscritos.
  - **`Professor` <-> `ElementoConteudo`:** A navegação é bidirecional. O professor navega para os conteúdos que "Gerencia", e de um conteúdo, pode-se navegar de volta para o professor responsável.
  - **`ElementoConteudo` -> `RecursosMultimidia`:** A navegação é unidirecional, representando o fluxo primário do usuário, que acessa o conteúdo e, a partir dele, ao recurso multimídia associado.
  - **`Autoria` <-> `Autor`:** A navegação é bidirecional, permitindo uma exploração rica. O usuário pode ver o autor de um texto e, da página do autor, ver todos os textos de sua autoria.

- **`ElementoConteudo -> Professor:`** O diagrama também define a navegação unidirecional de um conteúdo para o professor que o "Gerencia", permitindo ao aluno identificar o responsável.
- **`ElementoConteudo <-> ElementoConteudo:`** A associação reflexiva "Pre-requisito"foi definida como bidirecional, permitindo ao aluno navegar de um conteúdo para o(s) seu(s) pré-requisito(s) e vice-versa.
- **`RecursosMultimidia -> Professor:`** Permite a navegação de um recurso para o professor que realizou o *upload*.

4.2.2 Modelo de Estrutura Navegacional

Segunda e última etapa do design de navegação. Ele é construído com base no Modelo de Classes Navegacional que acabamos de definir (KOCH; MANDEL, 1999, p. 373).

Enquanto o modelo anterior definia quais classes podiam ser visitadas e os links diretos entre elas, este modelo define como esses nós navegacionais são de fato acessados e organizados do ponto de vista do usuário (KOCH; MANDEL, 1999, p. 374).

Para fazer isto, a metodologia introduz novos elementos de modelo que funcionam como ferramentas de acesso à informação (KOCH; MANDEL, 1999, p. 439). Os mais importantes são:

`navigationalContext` É o tipo base. Representa uma sequência simples de nós, geralmente derivados de uma classe ou de uma associação. Koch et al. exemplificam com "todos os festivais (festivais por evento)"(KOCH; MANDEL, 1999, p. 383). No contexto do *Lingua Latina Adaptativa*, um exemplo seria "todos os Autores por ordem alfabética".

`groupedContext` É um contexto mais complexo, definido como uma "sequência de sequências"de nós navegacionais (KOCH; MANDEL, 1999, p. 383). O exemplo dado é "filmes por ator", que denota uma sequência de atores, onde cada ator, por sua vez, tem uma sequência de filmes em que participou (KOCH; MANDEL, 1999, p. 383). No contexto do *Lingua Latina Adaptativa*, poderíamos ter "Elementos de Conteúdo por Professor".

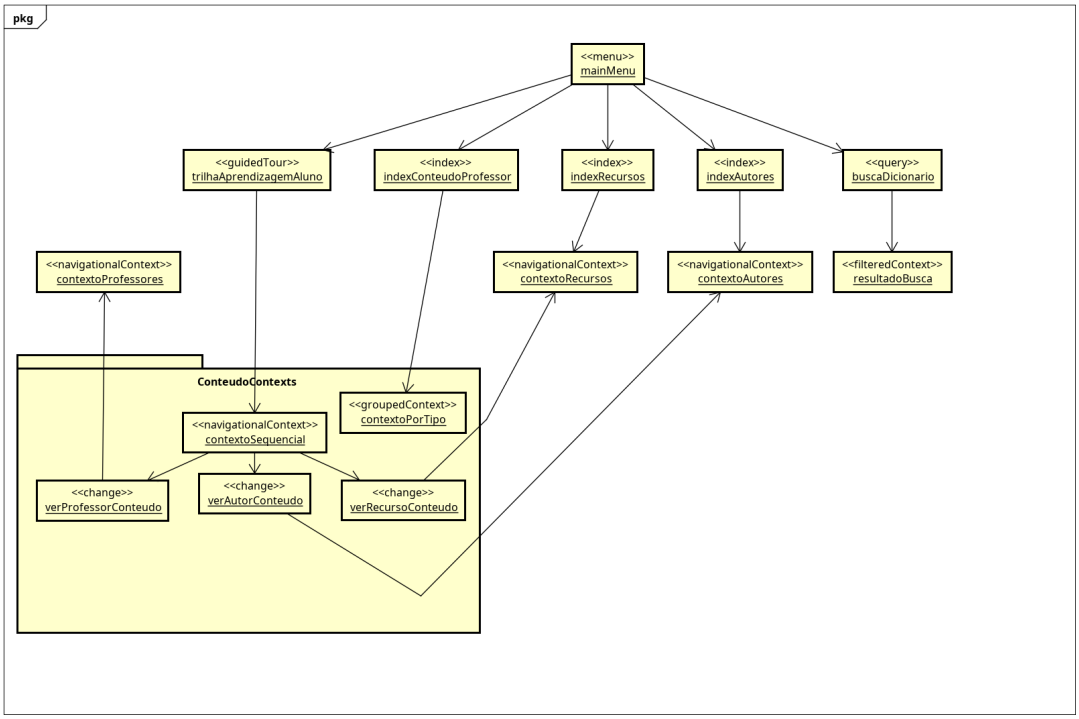
`filteredContext:` Este contexto permite uma seleção dinâmica de elementos que satisfazem uma propriedade específica (KOCH; MANDEL, 1999, p. 383). Esta propriedade é, geralmente, fornecida pelo usuário através de uma consulta (query) (KOCH; MANDEL, 1999, p. 383). Um exemplo seria o resultado da consulta "todos os filmes nomeados para a categoria 'melhor música original' em 1998"(KOCH; MANDEL, 1999). No contexto do sistema proposto, seria o resultado de uma pesquisa no `Dicionario`.

Primitivas de Acesso: São as "portas de entrada"para os contextos navegacionais. As principais são:

- `menu`: Um ponto de entrada principal, como uma página inicial (*homepage*).

- `index`: Permite acesso direto a qualquer elemento de um contexto (ex: uma lista alfabética de todos os `Autores`) .
- `query`: Um formulário de pesquisa que resulta num contexto filtrado (ex: pesquisar um termo no `Dicionario`).
- `guidedTour` Dá acesso ao primeiro item de um contexto e permite apenas a navegação sequencial (seguinte/anterior).

Figura 11 – Diagrama do modelo estrutural navegacional



Fonte: Autor

O diagrama resultante não é um diagrama de classes, mas sim um Diagrama de Objetos UML (KOCH; MANDEL, 1999, p. 468), que mostra como instâncias destas primitivas (um objeto `mainMenu`, um objeto `indexAutores`) se ligam aos contextos de navegação.

O Modelo de Estrutura Navegacional resultante para o *Lingua Latina Adaptativa* está ilustrado na Figura 11.

O ponto de partida é o objeto «`menu`» `mainMenu`, que fornece o acesso principal às diferentes primitivas de acesso do sistema:

- `trilhaAprendizagemAluno`: Um «`guidedTour`» que representa o caminho de aprendizagem sequencial do Aluno, conduzindo-o ao `contextoSequencial` («`navigational`
  Isto força o aluno a percorrer os conteúdos na ordem definida pelos pré-requisitos, conforme modelado no diagrama de classes navegacional.

- `indexConteudoProfessor`: Um «`index`» que serve como ponto de entrada de gestão para o Professor, dando acesso ao `contextoPorTipo` («`groupedContext`»), onde os conteúdos são apresentados agrupados por categoria (Provas, Exercícios, Gramática, etc.).
- `indexRecursos`: Um «`index`» para listar todos os recursos multimídia, ligando-se ao `contextoRecursos`.
- `indexAutores`: Um «`index`» para listar todos os autores, ligando-se ao `contextoAutores`.
- `buscaDicionario`: Uma «`query`» para pesquisar no dicionário, gerando o `resultadoBusca` («`filteredContext`»).

O núcleo da navegação do conteúdo é modelado pelo pacote `ConteudoContexts`, que agrupa os contextos e as mudanças de contexto associadas à navegação de conteúdo. Dentro deste pacote, o `contextoSequencial` é o contexto principal do Aluno. A partir dele, a metodologia UWE prevê a definição de mudanças de contexto («`change`») (KOCH; MANDEL, 1999, p. 383), que permitem ao usuário saltar para outro espaço navegacional e regressar ao ponto de origem. As mudanças de contexto modeladas são:

- `verProfessorConteudo` («`change`»): A partir do `contextoSequencial`, permite ao aluno navegar para o `contextoProfessores` («`navigationalContext`»), onde pode identificar o professor responsável pelo conteúdo em estudo.
- `verAutorConteudo` («`change`»): Permite navegar do `contextoSequencial` para o `contextoAutores` («`navigationalContext`»), consultando o autor de um texto de autoria.
- `verRecursoConteudo` («`change`»): Permite navegar do `contextoSequencial` para o `contextoRecursos` («`navigationalContext`»), acessando os recursos multimídia associados ao conteúdo.

Este diagrama de objetos, portanto, não mostra as classes, mas sim a "arquitetura da informação"do sistema, conectando menus, índices, *guided tours* e mudanças de contexto aos agrupamentos navegacionais que estruturam a experiência de cada perfil de usuário.

4.3 DESIGN DE APRESENTAÇÃO

A etapa final do processo de design é o Design de Apresentação (*Presentational Design*) (KOCH; MANDEL, 1999). O seu objetivo é modelar uma "interface de usuário abstrata"(KOCH; MANDEL, 1999). Esta etapa é construída com base no Modelo de Estrutura Navegacional (Seção

4.2.2), definindo "a forma como os nós navegacionais irão aparecer"e "quais objetos da interface do usuário ativarão a navegação"(KOCH; MANDEL, 1999, p. 879).

Esta abordagem, em conformidade com a metodologia UWE, segue a separação de interesses, definindo a "forma como os nós navegacionais irão aparecer"e "quais objetos da interface do usuário ativarão a navegação"e as transformações que ocorrem (KOCH; MANDEL, 1999, p. 879). O Design de Apresentação é dividido em dois modelos subsequentes (KOCH; MANDEL, 1999):

1. **Modelo de Apresentação Estático:** Descreve a estrutura da interface do usuário, ou seja, "como as interfaces de usuário são construídas"através de diagramas de composição.
2. **Modelo de Apresentação Dinâmico:** Descreve o comportamento dos objetos da interface, usando Diagramas de Estado UML , para modelar a resposta a eventos do usuário.

4.3.1 Modelo de Apresentação Estático

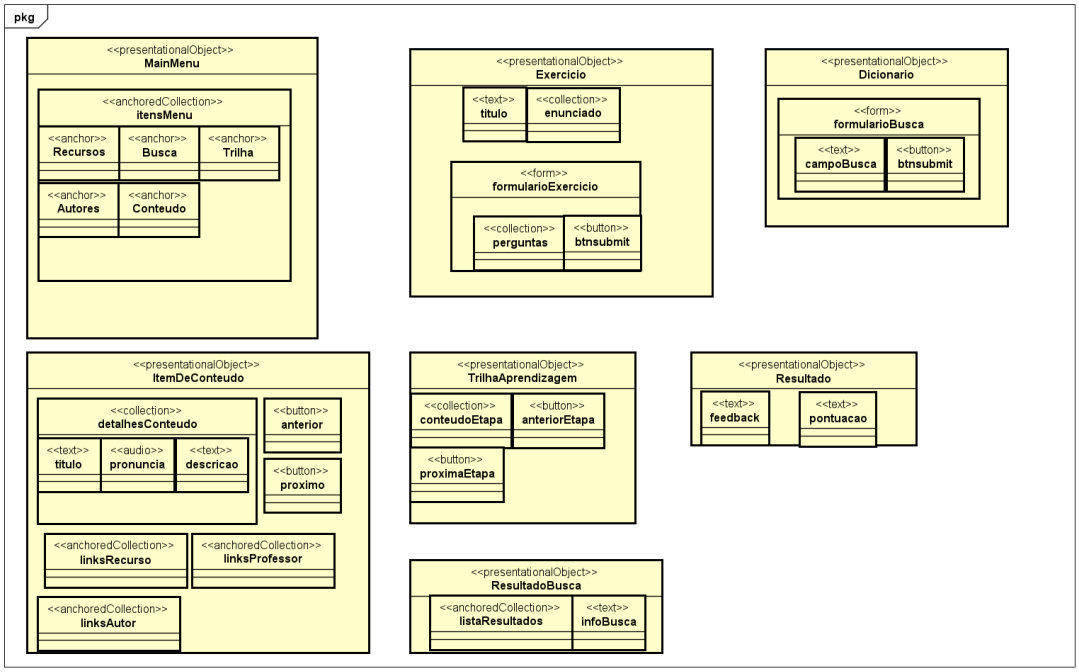
O Modelo de Apresentação Estático define "como os nós navegacionais são apresentados ao usuário"(KOCH; MANDEL, 1999). A técnica de modelagem utilizada é o diagrama de composição UML, que descreve a interface como uma composição de "objetos de interface de usuário"(KOCH; MANDEL, 1999).

Tais objetos podem ser primitivos ou compostos por outros objetos (KOCH; MANDEL, 1999). A metodologia define um conjunto de estereótipos para estes objetos ,sendo os principais utilizados no nosso modelo:

- «`presentationalObject`»: O contentor principal que se baseia num nó navegacional.
- «`anchor`»: Uma "área clicável"que é o ponto de partida para a navegação.
- «`anchoredCollection`»: Uma coleção de «`anchor`».
- «`text`»: Um bloco de texto (KOCH; MANDEL, 1999, p. 909).
- «`audio`»: Um objeto multimídia de áudio.
- «`button`»: Uma área clicável para uma ação interna (como submeter um formulário).
- «`form`»: Usado para solicitar informação ao usuário.
- «`collection`»: Um conjunto de outros objetos de interface.

A Figura 12 ilustra o diagrama de composição resultante para a interface de uma lição, integrando estes elementos e estereótipos para definir a estrutura visual da página.

Figura 12 – Diagrama do modelo estrutural de apresentação



Fonte: Autor

4.3.2 Modelo de Apresentação Dinâmico

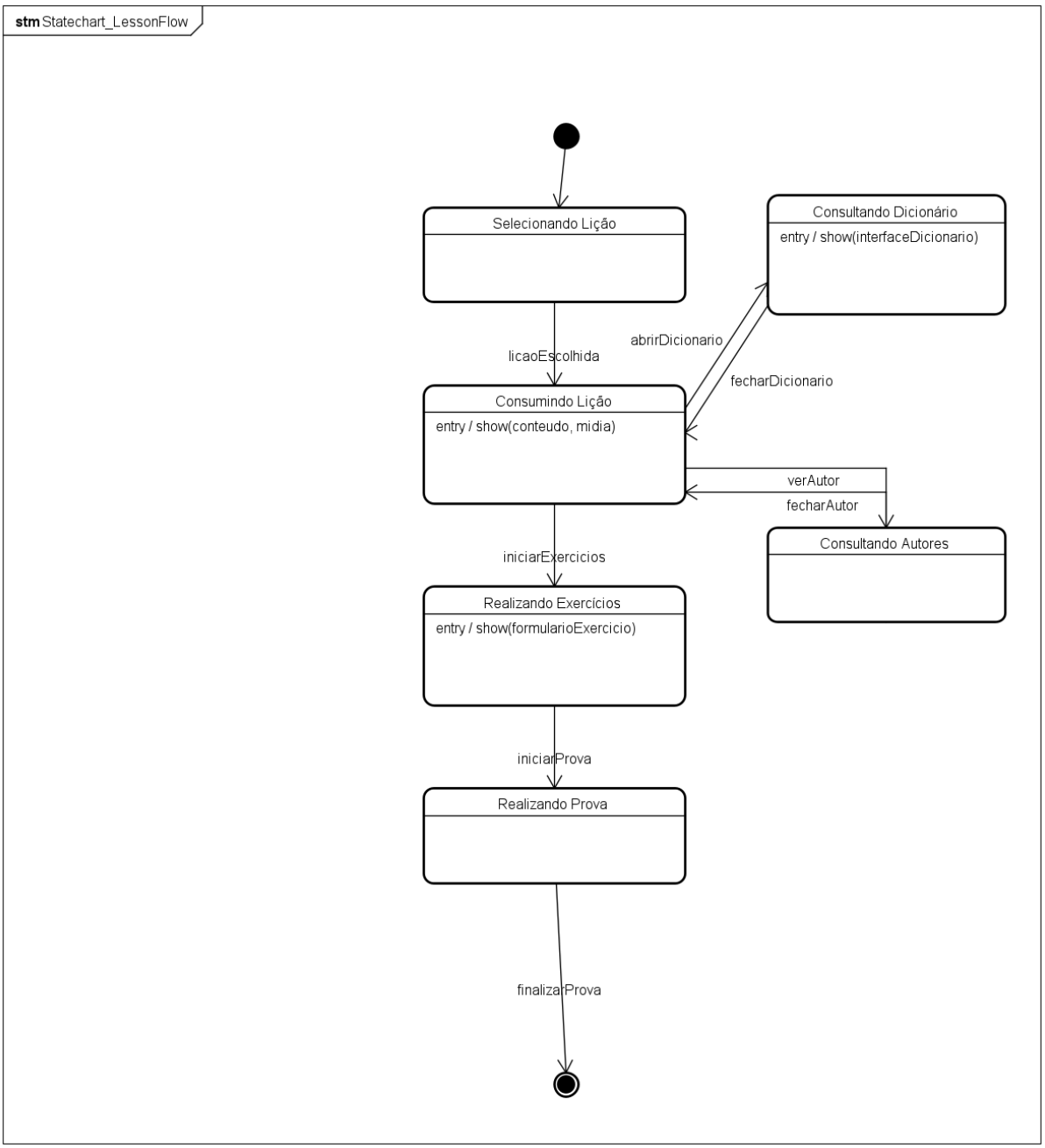
Enquanto o modelo estático define o layout, o modelo dinâmico descreve o "comportamento dos objetos de apresentação". Ele especifica como a interface reage a eventos externos ou eventos internos. A metodologia utiliza Diagramas de EstadoUML para modelar esta dinâmica (KOCH; MANDEL, 1999).

A Figura 13 modela o ciclo de vida de um `ElementoConteudo` (como uma lição de morfologia) na interface do aluno, ilustrando a interação entre o aluno e o motor de adaptação.

O fluxo de apresentação para um aluno, conforme a Figura 13, segue os seguintes estados e transições:

- **Estado Inicial (Lesson Locked):** Por padrão, um `ElementoConteudo` está no estado `Lesson Locked` (Lição Bloqueada). O aluno pode ver o título, mas não pode acessá-lo.
- **Transição (Adaptação):** O sistema (Modelo de Adaptação) dispara um evento interno `PrerequisitesMet` (Pré-requisitos Cumpridos) assim que o Modelo de Usuário (‘Conhecimento‘) indica que o aluno dominou os conceitos necessários. Isso move a lição para o estado `Lesson Unlocked` (Lição Desbloqueada). Na interface, isso pode ser representado pela técnica de *anotação adaptativa* (ex: o link muda de cinza para azul).
- **Transição (Interação):** Um evento externo `UserClicksLink` (disparado pelo aluno) move o sistema para o estado `Reading Content` (Lendo Conteúdo).

Figura 13 – Modelo de Apresentação Dinâmico



Fonte: Autor

- **Estado (Reading Content):** Este é um superestado. Enquanto o aluno está nesta página, um evento interno de tempo (`TimeOnPage > 1min`) dispara uma ação interna (‘Update User Model [Status: ’Lido’]‘) que atualiza o Modelo de Usuário sem mudar o estado visível.
- **Transição (Avaliação):** Quando o aluno aciona o evento externo `UserClicksExercise`, o sistema transita para o estado `Solving Exercise` (Resolvendo Exercício).
- **Transição (Remediação ou Sucesso):** A partir do estado de exercício, existem dois cami-

nhos. Se o aluno disparar `UserSubmitsWrong` (Resposta Errada), o sistema o retorna ao estado `Reading Content`, forçando uma revisão. Se o aluno disparar `UserSubmitsCorrec` (Resposta Correta), o sistema move a lição para seu estado final, `Lesson Complete` (Lição Concluída), e o Modelo de Usuário é atualizado para "dominado".

Este diagrama de estado demonstra como o comportamento da interface é governado por uma combinação de eventos externos do usuário e eventos internos do motor de adaptação (KOCH; MANDEL, 1999), garantindo um fluxo pedagógico controlado.

4.4 IMPLEMENTAÇÃO: REQUISITOS, ARQUITETURA E TECNOLOGIAS

A modelagem apresentada nas seções anteriores constitui a especificação do sistema. A presente seção complementa esse trabalho com os requisitos funcionais e não-funcionais derivados do modelo, a arquitetura de software adotada para o desenvolvimento e a pilha tecnológica selecionada. Por fim, delimitam-se explicitamente os escopos do Produto Mínimo Viável (MVP) e as funcionalidades que ficam fora do escopo desta etapa.

4.4.1 Metodologia de Desenvolvimento: Spec-Driven Development

A abordagem adotada para o desenvolvimento do MVP é o *Spec-Driven Development* (SDD), paradigma no qual a especificação — e não o código — é tratada como o artefato primário do processo de desenvolvimento (PISKALA, 2026). Com a disseminação de assistentes de codificação baseados em inteligência artificial, o SDD ganhou renovada relevância: ao estabelecer a especificação como fonte de verdade, elimina-se a ambiguidade que leva esses assistentes a gerar código inconsistente com os requisitos do sistema (PISKALA, 2026).

Piskala (2026) propõem três níveis de rigor para o SDD:

a) **Spec-First:** a especificação completa é redigida antes de qualquer linha de código; o código é derivado ou verificado contra ela;

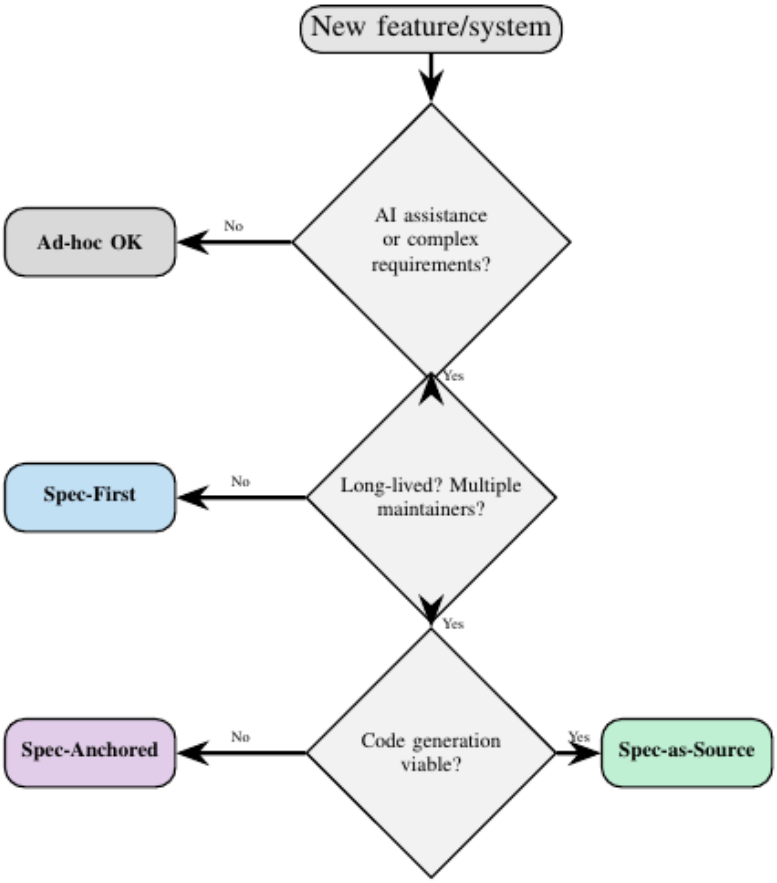
b) **Spec-Anchored:** a especificação ancora um código preexistente, sendo mantida em sincronia com a evolução do sistema;

c) **Spec-as-Source:** a especificação formal é o único artefato de entrada; todo o código é gerado automaticamente a partir dela.

Para auxiliar na seleção do nível mais adequado, Piskala (2026) apresentam o fluxograma reproduzido na Figura 14.

Aplicando o fluxograma ao contexto deste projeto: (1) o desenvolvimento conta com assistência de IA (*GitHub Copilot*) e envolve requisitos adaptativos de alta complexidade — condição que indica o uso de SDD; (2) o sistema não possui base de código preexistente a ser

Figura 14 – Fluxograma de seleção do nível de rigor do SDD



Fonte: (PISKALA, 2026)

ancorada, descartando o nível *Spec-Anchored*; (3) o projeto é conduzido por um único desenvolvedor em prazo definido pelo TCC, o que torna a geração totalmente automatizada de código (*Spec-as-Source*) inadequada para este escopo. O nível resultante é, portanto, o **Spec-First**.

Este enquadramento é coerente com a estrutura do próprio trabalho: seja na especificação — os modelos UWE (Conceitual, Navegacional e de Apresentação) como nas regras OCL AM-1 a AM-6. Assim, O código produzido na etapa seguinte será, portanto, um artefato secundário derivado desta especificação.

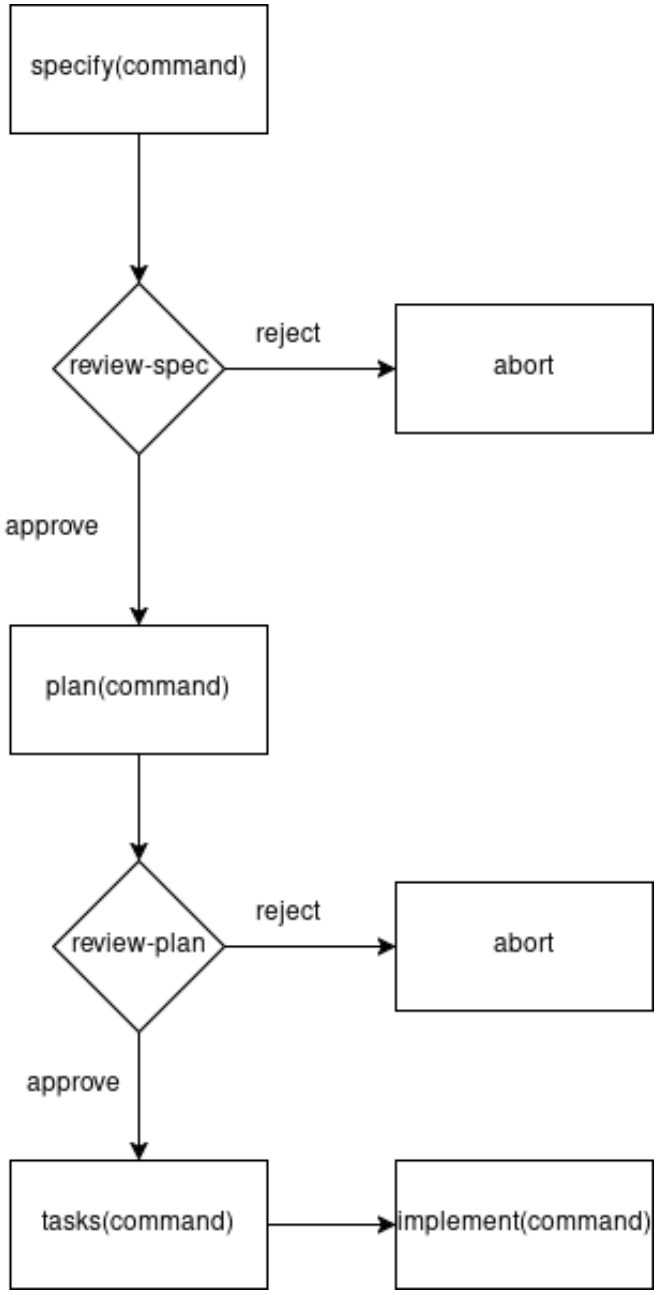
4.4.2 Spec Kit Agents

Para operacionalizar o SDD *Spec-First* durante a implementação, adota-se o *Spec Kit Agents* (TAGHAVI; BHAVANI, 2026), um *pipeline* multi-agente composto por papéis de gerente de produto (PM) e desenvolvedor, estruturado em quatro fases sequenciais: *Specify*, *Plan*, *Tasks* e *Implement*.

O diferencial do *Spec Kit Agents* em relação a abordagens agenticas convencionais é a introdução de *hooks* de *context-grounding*: sondas somente-leitura que, ao início de cada fase,

consultam o repositório para fundamentar o agente em evidências concretas — arquivos reais, nomes de classes, esquemas de banco de dados —, evitando a geração de APIs fictícias e violações arquiteturais (TAGHAVI; BHAVANI, 2026). Validação intermediária entre fases garante que o artefato produzido em cada etapa esteja em conformidade com o ambiente antes de prosseguir.

Figura 15 – Fluxo de trabalho do *Spec Kit Agents*



Fonte: Autor, adaptado de GitHub Spec Kit (2026)

A Figura 15 ilustra o fluxo completo. A fase *Specify* recebe como entrada os modelos UWE e as regras OCL deste capítulo; a fase *Plan* decompõe a especificação em tarefas técnicas rastreáveis; a fase *Tasks* gera os critérios de aceitação por tarefa; e a fase *Implement* produz o código verificado contra a especificação e o repositório.

A adoção desta abordagem justifica-se por dois fatores de qualidade:

a) **Rastreabilidade arquitetural:** os *hooks* de validação garantem que o código gerado para cada módulo Next.js (Quadro 6) respeite as fronteiras de domínio definidas na especificação UWE, em conformidade com o RNF04 (manutenibilidade);

b) **Evidência empírica:** Taghavi e Bhavani (2026) reportam que os *hooks* de *context-grounding* melhoram a qualidade avaliada em $+0{,}15$ pontos (em uma escala de 1 a 5) e mantêm compatibilidade de testes entre 99,7% e 100% em nível de repositório, sobre 128 execuções cobrindo 32 funcionalidades em cinco repositórios distintos.

4.4.3 Requisitos do Sistema

Os requisitos a seguir foram derivados diretamente dos modelos conceitual, de adaptação, navegacional e de apresentação elaborados neste capítulo. Os requisitos funcionais descrevem as capacidades que o sistema deve prover; os não-funcionais estabelecem restrições de qualidade sobre como essas capacidades devem ser realizadas.

Quadro 4 – Requisitos funcionais do *Lingua Latina Adaptativa*

ID	Categoria	Descrição
RF01	Gestão de Usuários	O sistema deve permitir o cadastro, a autenticação e o gerenciamento de perfil de `Aluno` e `Professor`.
RF02	Modelo de Usuário	O sistema deve persistir o progresso do `Aluno` em cada `ElementoConteudo` (`status`, `tentativas`, `acertos`).
RF03	Gestão de Conteúdo	O `Professor` deve poder criar, editar e remover instâncias de `Gramatica`, `Texto`, `Exercicio`, `Prova` e `Dicionario`.
RF04	Gestão de Conteúdo	O `Professor` deve poder associar `RecursosMultimidia` a um `ElementoConteudo`.
RF05	Gestão de Conteúdo	O `Professor` deve poder definir relações de pré-requisito entre conteúdos e a ordem pedagógica de cada item.
RF06	Apresentação Adaptativa	O `Professor` deve poder cadastrar `FragmentoExplicacao` com conteúdo de pré-requisito opcional, para uso com *stretchtext*.
RF07	Navegação Adaptativa	O sistema deve exibir os links de conteúdo com anotação visual de estado (*Bloqueado*, *Disponível*, *Dominado*), conforme a regra AM-4.
RF08	Navegação Adaptativa	Links para conteúdos cujos pré-requisitos não foram cumpridos devem permanecer desabilitados, conforme a regra AM-1.
RF09	Navegação Adaptativa	O sistema deve oferecer um `guidedTour` que sugere automaticamente o próximo conteúdo desbloqueado ao `Aluno`, conforme a regra AM-3.
RF10	Apresentação Adaptativa	O sistema deve exibir ou ocultar fragmentos de explicação gramatical adicional conforme o histórico de domínio do `Aluno` (*stretchtext*), conforme a regra AM-5.
RF11	Motor de Adaptação	Ao submeter um `Exercicio`, o sistema deve avaliar a resposta e atualizar o `Progresso` do aluno, conforme as regras AM-6 e AM-6b.
RF12	Motor de Adaptação	A transição para o estado *Dominado* deve desbloquear automaticamente os conteúdos que possuem o item concluído como pré-requisito.

Fonte: Autor

Quadro 5 – Requisitos não-funcionais do *Lingua Latina Adaptativa*

ID	Categoria	Descrição
RNF01	Usabilidade	A interface deve ser acessível via navegador *web*, sem necessidade de instalação local.
RNF02	Acessibilidade	O estado de anotação dos links deve ser distinguível sem dependência exclusiva de cor, em conformidade com as diretrizes WCAG 2.1.
RNF03	Desempenho	A avaliação das regras de adaptação (AM-1 a AM-6) deve ocorrer em tempo de resposta inferior a 500 milissegundos.
RNF04	Manutenibilidade	O motor de adaptação deve ser implementado como um módulo desacoplado da camada de apresentação, facilitando a evolução das regras sem impacto na interface.
RNF05	Segurança	O acesso às operações de gerenciamento de conteúdo deve ser restrito ao perfil `Professor`.
RNF06	Segurança	O progresso e os dados pessoais do `Aluno` devem ser persistidos com controle de acesso por sessão autenticada.
RNF07	Implantação	O sistema deve ser implantável como um único artefato (*monolito modular*), compatível com infraestrutura de baixo custo para o MVP.

Fonte: Autor

4.4.4 Arquitetura de Software

A partir dos requisitos elencados, adota-se o estilo arquitetural *Modular Monolith* para o MVP do *Lingua Latina Adaptativa*. Conforme Richards e Ford (2025), esse estilo organiza o sistema em módulos logicamente isolados que compartilham um único processo de implantação. A escolha é especialmente adequada quando a direção arquitetural ainda está sendo consolidada e quando os limites de domínio são bem compreendidos mas o overhead operacional de serviços distribuídos seria desproporcional ao escopo do projeto. Além disso, Richards e Ford (2025) destacam que o *Modular Monolith* preserva a opção de migração futura para microsserviços, bastando para isso respeitar as fronteiras de módulo desde o início do desenvolvimento.

O estilo se alinha diretamente à metodologia UWE adotada neste trabalho: cada fase de design — conceitual, adaptativo, navegacional e de apresentação — corresponde a um módulo bem delimitado no interior do monolito. Essa correspondência não é acidental: ela garante que a arquitetura de software seja diretamente rastreável à especificação UWE produzida neste capítulo, de modo que qualquer desvio de implementação em relação ao modelo possa ser identificado pela simples comparação entre módulo e diagrama correspondente. O sistema é implantado como um único artefato, mas com fronteiras de módulo rígidas que preservam a separação de interesses e facilitam a evolução independente de cada domínio. O Quadro 6 apresenta o mapeamento entre os módulos da arquitetura e os elementos do modelo UWE.

Quadro 6 – Mapeamento entre módulos da arquitetura e o modelo UWE

Módulo	**Corresponde a**	**Responsabilidade principal**
`usuarios`	Modelo de Usuário (UWE)	Autenticação, perfil, persistência do `Progresso`
`conteudo`	Modelo de Domínio	CRUD de todos os subtipos de `ElementoConteudo` e `RecursosMultimidia`
`adaptacao`	Modelo de Adaptação	Motor de regras AM-1 a AM-6; avaliação declarativa de pré-requisitos
`navegacao`	Design Navegacional	Lógica de `guidedTour`, anotação de links e mudanças de contexto
`apresentacao`	Design de Apresentação	Renderização de *stretchtext* e atualização parcial da interface

Fonte: Autor

4.4.5 Tecnologias Selecionadas

A pilha tecnológica foi definida a partir dos requisitos levantados, priorizando produtividade de desenvolvimento, aderência ao estilo *Modular Monolith* e suporte nativo às funcionalidades de adaptação.

Linguagem e *framework* principal — TypeScript e Next.js: O *framework* full-stack Next.js (Vercel, 2026) é adotado como base do sistema. Sua estrutura de diretórios baseada no *App Router* permite organizar o sistema em uma arquitetura de *Modular Monolith* de forma nativa, mapeando diretamente sobre os módulos do Quadro 6. Para a autoria e gestão de conteúdos (RF03, RF04 e RF05), será desenvolvida uma interface administrativa simplificada em React integrada às *Server Actions*. O motor de adaptação será implementado como um módulo de serviços desacoplado em TypeScript puro, em conformidade com o RNF04.

Banco de dados — PostgreSQL: O PostgreSQL (The PostgreSQL Global Development Group, 2026) é adotado como sistema gerenciador de banco de dados relacional. A escolha justifica-se pela robustez no suporte a chaves estrangeiras com integridade referencial e consultas eficientes sobre grafos de pré-requisitos — estrutura central ao modelo de adaptação. A modelagem de dados do sistema mapeia de forma segura as classes do Modelo de Domínio e permite que as regras de adaptação operem sobre objetos em TypeScript sem acoplamento direto ao SQL.

Interface e Apresentação Adaptativa — React Server Components (RSC): Para a camada de apresentação adaptativa, o Next.js utiliza React Server Components (RSC) combinados com componentes interativos do lado do cliente. Isso viabiliza a renderização eficiente no servidor das técnicas de anotação adaptativa (RF07) e a expansão dinâmica de conteúdos via *stretchtext* (RF10) com base no estado do progresso do usuário, sem a sobrecarga de carregar grandes volumes de JavaScript desnecessário.

A relação entre cada tecnologia e a especificação UWE produzida neste capítulo pode ser enunciada de forma direta, evidenciando a rastreabilidade exigida pelo nível *Spec-First* adotado (Seção 4.4.1):

a) o **Next.js** mapeia diretamente as classes e associações do Modelo de Domínio (Figura 9) em tipos estruturados TypeScript, servindo como base para as regras de adaptação;

b) o **PostgreSQL** armazena a rede relacional e garante a integridade referencial do grafo de pré-requisitos por meio de chaves estrangeiras, sustentando a aplicação das regras AM-1 e AM-3 sem lógica adicional na camada de aplicação;

c) o **Next.js / React** entrega o *stretchtext* (AM-5) e a anotação adaptativa de links (AM-4) de forma declarativa e reativa via componentes, preservando a hipermídia como primitiva de interface conforme o Design de Apresentação (Seção 4.3).

4.4.6 Escopo do MVP e Limites do Trabalho

O presente trabalho concentra-se integralmente na modelagem do sistema. O Quadro 7 delimita explicitamente o que foi e o que não foi implementado na etapa seguinte.

Quadro 7 – Delimitação do escopo de implementação do MVP

Dentro do escopo	**Fora do escopo (trabalhos futuros)**
Autenticação e gerenciamento de perfis (RF01)	Aplicativo móvel nativo
Painel do Professor: CRUD de conteúdos, pré-requisitos e recursos (RF03–RF06)	Geração automática de exercícios por inteligência artificial
Motor de adaptação declarativo: regras AM-1 a AM-6 (RF11, RF12)	Integração com sistemas externos de gestão de aprendizagem (LMS)
Navegação adaptativa com anotação de links e `guidedTour` (RF07–RF09)	Módulo de análise de aprendizagem (*learning analytics*)
Stretchtext adaptativo por fragmento gramatical (RF10)	Suporte a múltiplos idiomas de interface
Testes com usuários e avaliação de usabilidade	Implantação em ambiente de produção escalável
Uso do *Spec Kit Agents* como ferramenta de desenvolvimento assistido por IA (Seção 4.4.2)	Geração totalmente automática de código sem revisão humana (*Spec-as-Source*)

Fonte: Autor

5 IMPLEMENTAÇÃO DA SOLUÇÃO PROPOSTA

O presente capítulo detalha a implementação do protótipo do sistema *Lingua Latina Adaptativa*[1], cuja modelagem foi especificada no Capítulo 4. Descreve-se o processo de desenvolvimento orientado por especificações (*Spec-Driven Development*) com o auxílio de agentes de inteligência artificial, a arquitetura de software e decisões de banco de dados, o mecanismo de prevenção de ciclos no grafo de dependências, e uma análise funcional das interfaces a partir do passeio visual pelas capturas de tela da aplicação.

5.1 PROCESSO DE DESENVOLVIMENTO E FERRAMENTAS AGÊNTICAS

As modelagens desenvolvidas nas etapas anteriores foram concebidas precisamente para compor cada fase do fluxo de trabalho do *Spec-Driven Development* (SDD), de modo que a especificação do sistema se tornasse o fundamento direto para a geração do seu núcleo funcional. Nesse processo, o ciclo completo de SDD permitiu estruturar os principais componentes da solução, enquanto o uso do Codex CLI, por meio do modo `/plan` e da etapa de implementação, ficou reservado aos ajustes pontuais, aos refinamentos de integração e às correções necessárias para consolidar o sistema final. Tal estratégia mostrou-se mais adequada por evitar retrabalho, reduzir o consumo de *tokens* e contornar a burocracia inerente a um processo iterativo excessivamente fragmentado, que por vezes se aproxima da lógica de um modelo em cascata.

O desenvolvimento do protótipo adotou a abordagem de *Spec-Driven Development* (SDD) no nível *Spec-First*, na qual a especificação completa de modelagem UWE e as regras OCL formuladas no TCC 1 foram tratadas como o artefato primário e fonte da verdade. Todo o código do sistema foi derivado diretamente desses modelos.

Para operacionalizar o SDD de maneira ágil, utilizou-se um pipeline de desenvolvimento baseado em agentes de inteligência artificial e assistentes de codificação de última geração:

1. **GitHub Spec Kit:** A estrutura inicial e o esqueleto de serviços da aplicação Next.js foram gerados utilizando o framework *GitHub Spec Kit* (TAGHAVI; BHAVANI, 2026). O pipeline opera combinando agentes que atuam como gerentes de produto (PM) e desenvolvedores em fases sequenciais de especificação, planejamento, geração de tarefas e implementação. O diferencial desse framework é o uso de *context-grounding hooks* — sondas de leitura que analisam o ambiente real de desenvolvimento para evitar a criação de APIs

[1] A solução está hospedada e disponível online em: <https://tcc-nextjs.vercel.app/>.

inconsistentes e alucinações de código, garantindo que o código TypeScript respeitasse estritamente as restrições conceituais modeladas.

2. **Codex CLI:** Para os refinamentos pontuais e a codificação de regras e componentes, foi utilizado o assistente de linha de comando *Codex CLI* (OpenAI, 2026). O Codex CLI é uma interface de terminal textual (*Text User Interface* — TUI) que realiza a indexação completa do projeto, permitindo que a base de código seja editada e ajustada diretamente por agentes de IA. A ferramenta oferece em seu menu de configuração o suporte a múltiplos modelos de linguagem, dentre eles o GPT 5.5, GPT 5.4 e GPT 5.4 mini.

3. **Modo /plan do Codex CLI:** Cada funcionalidade do protótipo foi precedida pelo acionamento do comando `/plan` no Codex CLI. Para a elaboração do plano e arquitetura das regras da solução, foi utilizado o modelo avançado GPT 5.5, cuja capacidade analítica reduziu incompatibilidades estruturais; uma vez aprovada a estratégia de modificação pelo usuário, a escrita física de código e geração de testes foi delegada ao modelo GPT 5.4. No modo de planejamento, o Codex CLI não realiza alterações de código imediatas, mas gera um documento descritivo contendo a relação de arquivos a serem alterados, novos arquivos, dependências afetadas e critérios de aceitação específicos, executando a implementação somente sob a aprovação explícita do desenvolvedor humano.

A utilização coordenada dessas ferramentas agênticas acelera expressivamente a entrega de projetos de software. Os agentes agem de maneira assíncrona executando varreduras de código, análises sintáticas, consultas ao banco de dados e reescritas de arquivos de maneira paralela. O desenvolvedor passa de uma atividade puramente manual para um papel de arquiteto e revisor de código, onde o foco reside em validar se os planos propostos pelos agentes estão alinhados com as metas pedagógicas e de usabilidade do sistema.

5.2 ARQUITETURA E MAPEAMENTO DE MÓDULOS NO NEXT.JS

A base de código foi estruturada utilizando o framework full-stack **Next.js** com suporte a **TypeScript** e persistência relacional em banco de dados **PostgreSQL** (hospedado no provedor *Neon*). Conforme proposto na modelagem, o sistema adota o estilo arquitetural de *Modular Monolith* (Monolito Modular), garantindo que as fronteiras dos módulos conceituais do UWE fossem respeitadas na árvore de arquivos da aplicação.

O mapeamento de persistência de dados no banco de dados reflete o padrão de herança de classes do modelo conceitual, adotando o padrão de herança de tabelas múltiplas (*multi-table inheritance*) na modelagem relacional. A tabela central `conteudo_elementoconteudo` persiste os atributos genéricos das lições e testes, enquanto tabelas filhas (como `conteudo_morfologia` ou `conteudo_exercicio`) estendem essa estrutura contendo uma chave estrangeira do tipo um-para-um (`elementoconteudo_ptr_id`) direcionada ao registro pai. A camada de con-

sulta no Next.js (`lib/queries.ts`) resolve o polimorfismo em tempo de execução via *Outer Joins* e cláusulas condicionais `CASE WHEN` em SQL bruto, instanciando os tipos corretos na aplicação TypeScript.

5.3 PREVENÇÃO DE CICLOS E LÓGICA DO GRAFO DE PRÉ-REQUISITOS

A trilha de aprendizagem do sistema é modelada matematicamente como um Grafo Dirigido Acíclico (DAG). Cada nó representa um `ElementoConteudo` (Lição, Exercício ou Prova) e cada aresta dirigida representa um pré-requisito (`conteudo_prerequisitosconteudo`). A direção pedagógica aponta do pré-requisito para o conteúdo dependente ($A \rightarrow B$), exigindo que o aluno conclua A antes de acessar B.

A ocorrência de ciclos de dependência (como $A \rightarrow B \rightarrow C \rightarrow A$) representa um deadlock pedagógico: o aluno nunca conseguiria acessar a lição A porque ela exige o domínio de C, que por sua vez exige B, que exige a própria lição A. Por conseguinte, o sistema deve garantir que o grafo permaneça acíclico.

O algoritmo selecionado para realizar a detecção de dependências cíclicas é o Algoritmo de Tarjan para Componentes Fortemente Conectados (CFC) (TARJAN, 1972). O Tarjan realiza uma busca em profundidade (DFS) em tempo linear $O(|V| + |E|)$ sobre o grafo. A lógica de identificação de ciclos opera sob os seguintes critérios:

- Uma componente fortemente conectada é um subconjunto de nós em que cada nó é alcançável a partir de qualquer outro no mesmo subconjunto.
- No contexto de um grafo de pré-requisitos, se qualquer CFC encontrada contiver mais do que um único nó, ou se contiver um único nó com uma ligação reflexiva direcionada a si mesmo, está confirmada a existência de uma dependência cíclica inválida.

O sistema executa a rotina do algoritmo de Tarjan em duas situações distintas para assegurar a consistência:

1. **Validação em tempo de edição de conteúdo:** Quando o professor edita ou cria uma lição individual informando seus pré-requisitos, o sistema intercepta os dados na API e monta um grafo simulado contendo o estado futuro das ligações. A função de validação verifica a presença de CFCs cíclicas. Caso seja detectado um ciclo, a alteração é rejeitada com erro HTTP `400 Bad Request`, retornando uma mensagem estruturada que descreve o caminho exato do ciclo (ex: *"Esta alteração criaria uma dependência cíclica: A -> B -> A. Remova um dos pré-requisitos para continuar"*).
2. **Validação global no editor de trilha:** Quando o professor utiliza a interface visual do grafo para interligar múltiplos nós simultaneamente, a API de persistência global do grafo

executa uma varredura sobre todas as arestas enviadas pelo navegador. Se o Tarjan identificar qualquer ciclo, o banco de dados aborta a transação garantindo a integridade referencial.

A fim de demonstrar a materialização do algoritmo, o Algoritmo 9 apresenta o código-fonte em TypeScript que implementa a detecção de componentes fortemente conectados (CFC) utilizando o algoritmo de Tarjan, operando diretamente sobre a modelagem estruturada do grafo.

Algoritmo 9 – Algoritmo de Tarjan para detecção de componentes fortemente conectados

```
1  // Representacao de um grafo direcionado como lista de adjacencias
2  export type DirectedGraph = Record<string, readonly string[]>;
3
4  export function stronglyConnectedComponents(
5    graph: DirectedGraph
6  ): string[][] {
7    const vertices = new Set<string>();
8
9    for (const [vertex, neighbors] of Object.entries(graph)) {
10     vertices.add(vertex);
11     for (const neighbor of neighbors) {
12       vertices.add(neighbor);
13     }
14   }
15
16   const indexByVertex = new Map<string, number>();
17   const lowlinkByVertex = new Map<string, number>();
18   const onStack = new Set<string>();
19   const stack: string[] = [];
20   const components: string[][] = [];
21   let index = 0;
22
23   function strongconnect(vertex: string) {
24     indexByVertex.set(vertex, index);
25     lowlinkByVertex.set(vertex, index);
26     index += 1;
27     stack.push(vertex);
28     onStack.add(vertex);
29
30     for (const neighbor of graph[vertex] ?? []) {
31       if (!indexByVertex.has(neighbor)) {
32         strongconnect(neighbor);
33         lowlinkByVertex.set(
34           vertex,
35           Math.min(
36             lowlinkByVertex.get(vertex) ?? 0,
37             lowlinkByVertex.get(neighbor) ?? 0
```

```
38            )
39          );
40        } else if (onStack.has(neighbor)) {
41          lowlinkByVertex.set(
42            vertex,
43            Math.min(
44              lowlinkByVertex.get(vertex) ?? 0,
45              indexByVertex.get(neighbor) ?? 0
46            )
47          );
48        }
49      }
50
51      if (lowlinkByVertex.get(vertex) === indexByVertex.get(vertex)) {
52        const component: string[] = [];
53        let next = "";
54
55        do {
56          next = stack.pop()!;
57          onStack.delete(next);
58          component.push(next);
59        } while (next !== vertex);
60
61        components.push(component);
62      }
63    }
64
65    for (const vertex of vertices) {
66      if (!indexByVertex.has(vertex)) {
67        strongconnect(vertex);
68      }
69    }
70
71    return components;
72  }
```

Além disso, a validação que intercepta as ligações desenhadas pelo docente no editor visual baseia-se nos dados estruturados de nós e arestas da biblioteca React Flow, sendo implementada conforme a função descrita no Algoritmo 10. A rotina mapeia a lista de nós do React Flow (`contents`) e de conexões direcionadas (`edges`) para alimentar o algoritmo de Tarjan, retornando a mensagem de diagnóstico exata no formato de trilha percorrida para o erro de usabilidade.

Algoritmo 10 – Mapeamento e verificação de ciclos a partir de nós e conexões do React Flow

```
1  export interface PrerequisiteEdge {
2    from_conteudo_id: string | number;
```

```
3     to_conteudo_id: string | number;
4   }
5
6   export interface PrerequisiteContent {
7     id: string | number;
8     titulo: string;
9   }
10
11  export interface PrerequisiteCycleInput {
12    targetId: string | number;
13    targetTitle: string;
14    nextPrerequisites: Array<string | number>;
15    contents: PrerequisiteContent[];
16    edges: PrerequisiteEdge[];
17  }
18
19  export function detectPrerequisiteCycle(
20    input: PrerequisiteCycleInput
21  ): {
22    componentIds: string[];
23    cycleIds: string[];
24    message: string;
25  } | null {
26    const graph = buildGraph(input);
27    const labels = buildLabelMap(input);
28
29    // Executa Tarjan para encontrar componentes fortemente conectadas (CFC)
30    const components = stronglyConnectedComponents(graph);
31    const cyclicComponent = components.find((comp) => {
32      if (comp.length > 1) return true;
33      const [only] = comp;
34      return (graph[only] ?? []).includes(only); // Verifica auto-loop
35    });
36
37    if (!cyclicComponent) {
38      return null;
39    }
40
41    const cycleIds = findCyclePath(graph, cyclicComponent);
42    const cycleLabels = cycleIds.map((id) => labels.get(id) ?? id);
43
44    return {
45      componentIds: cyclicComponent,
46      cycleIds,
47      message: `Esta alteracao criaria uma dependencia ciclica: ` +
48        `${cycleLabels.join(" -> ")}. ` +
49        `Remova um dos pre-requisitos para continuar.`,
```

```
50      };
51  }
```

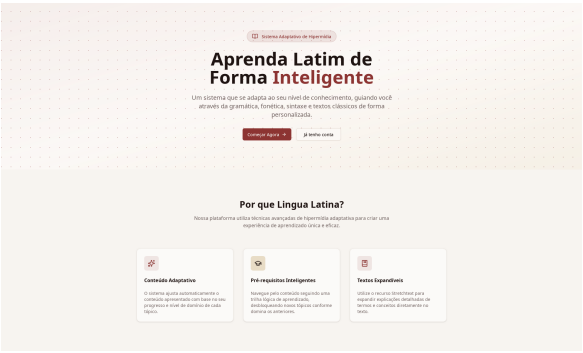
5.4 FLUXO DE UTILIZAÇÃO DO SISTEMA E PASSEIO VISUAL

Esta seção ilustra a implementação do sistema *Lingua Latina Adaptativa* por meio de capturas de tela organizadas na ordem lógica de utilização do software, conectando os elementos visuais à proposta conceitual da pesquisa.

5.4.1 Autenticação, Registro e Onboarding

O fluxo inicial do sistema é composto pelas telas de apresentação, cadastro e definição de proficiência do estudante.

Figura 16 – Telas de apresentação e autenticação do sistema.



(a) Página de apresentação (*Landing Page*)

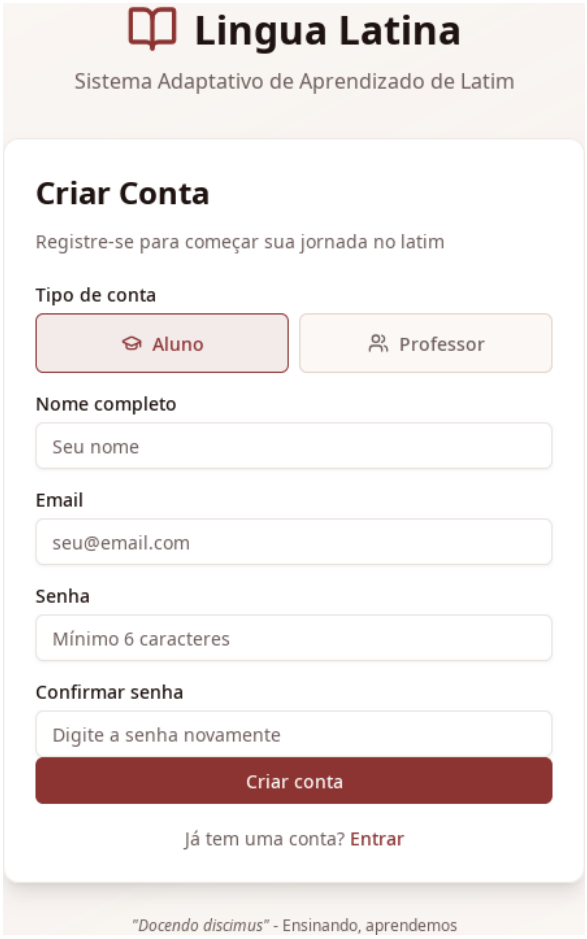
(b) Tela de login de usuário

Fonte: O Autor.

A Figura 16a apresenta a interface inicial da aplicação, introduzindo a proposta e permitindo o redirecionamento dos usuários. O formulário de login está ilustrado na Figura 16b, garantindo o controle de sessões conforme os requisitos do sistema.

O cadastro de novos usuários é efetuado por meio da interface representada na Figura 17a, onde define-se se o perfil criado será de Aluno ou Professor. Ao autenticar-se pela primeira vez, o Aluno é conduzido à tela de onboarding (Figura 17b) para escolher seu nível de proficiência inicial (*Básico*, *Intermediário* ou *Avançado*), o que preenche o atributo do Modelo de Usuário e determina quais nós da trilha estarão inicialmente no escopo do estudante.

Figura 17 – Formulários de registro e questionário de onboarding.



(a) Formulário de cadastro

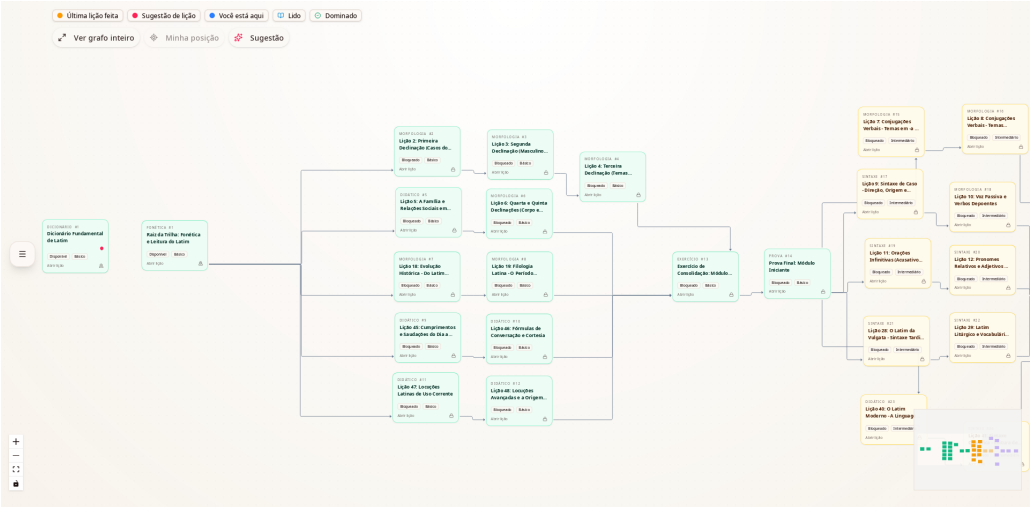
(b) Seleção de nível no primeiro login

Fonte: O Autor.

5.4.2 Visão Geral do Aluno: Home, Trilha e Dicionário

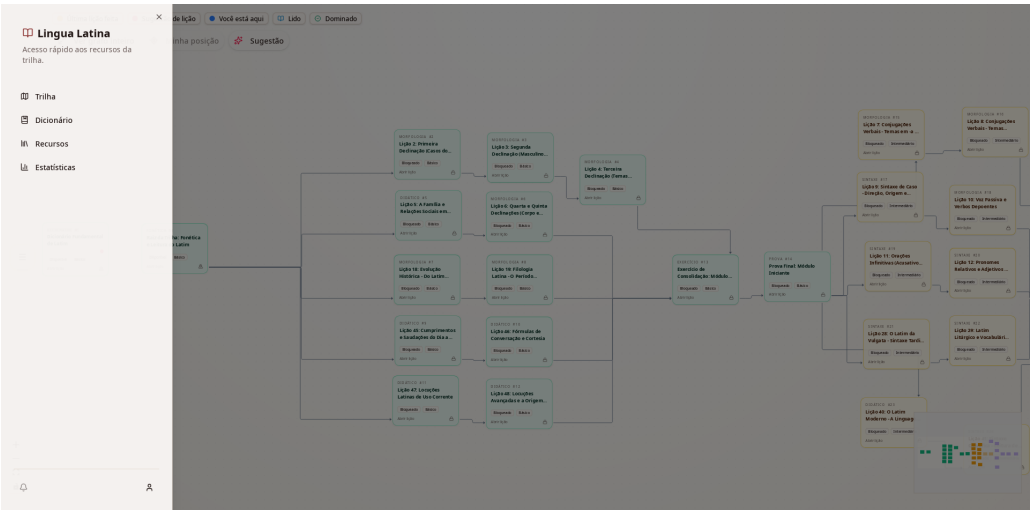
Uma vez autenticado e com nível definido, o aluno é direcionado à área interna do sistema.

Figura 18 – Visualização da trilha de aprendizagem como um grafo adaptativo (visão do aluno).



Fonte: O Autor.

Figura 19 – Menu principal de navegação do aluno no sistema.

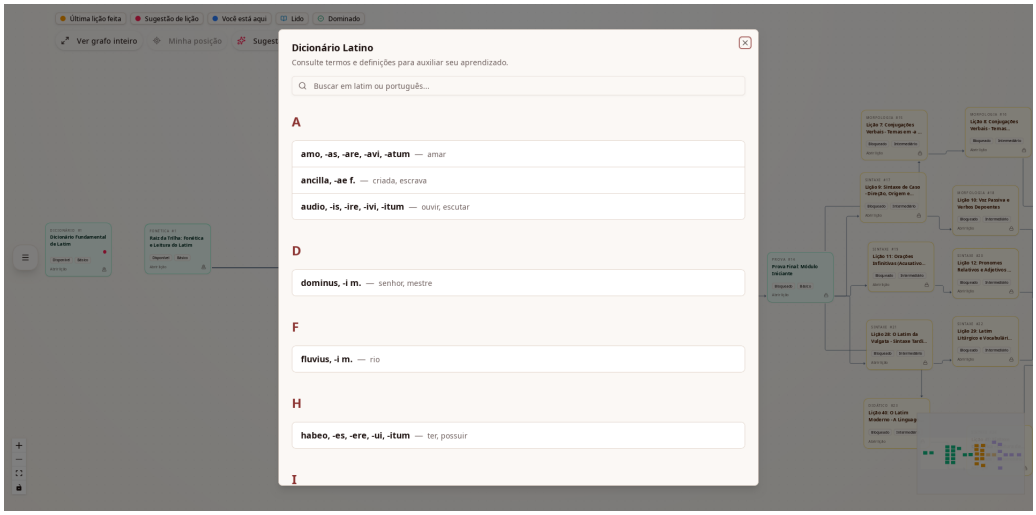


Fonte: O Autor.

A Figura 18 representa a trilha sob a perspectiva do Aluno. Cada lição ou avaliação é disposta graficamente. A reatividade de cores e ícones sinaliza o status atual do nó (bloqueado, disponível, dominado). O menu lateral retratado na Figura 19 oferece atalhos para os principais recursos do sistema.

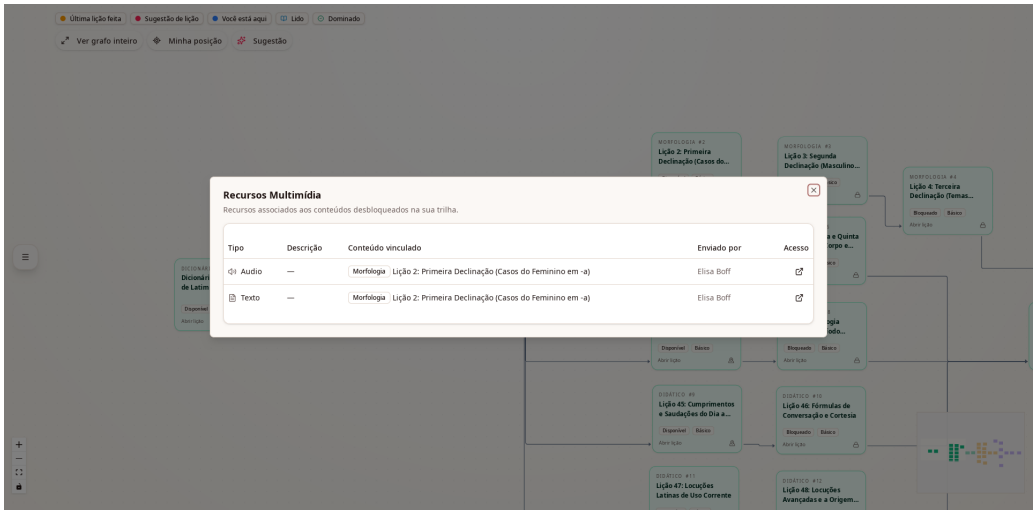
O Dicionário representado na Figura 20 funciona como uma ferramenta de apoio. O aluno pode realizar pesquisas de palavras e visualizar sua tradução correspondente. A listagem de recursos adicionais da Figura 21 consolida áudios e leituras extras cadastrados no sistema.

Figura 20 – Dicionário integrado para pesquisa de vocábulos clássicos.



Fonte: O Autor.

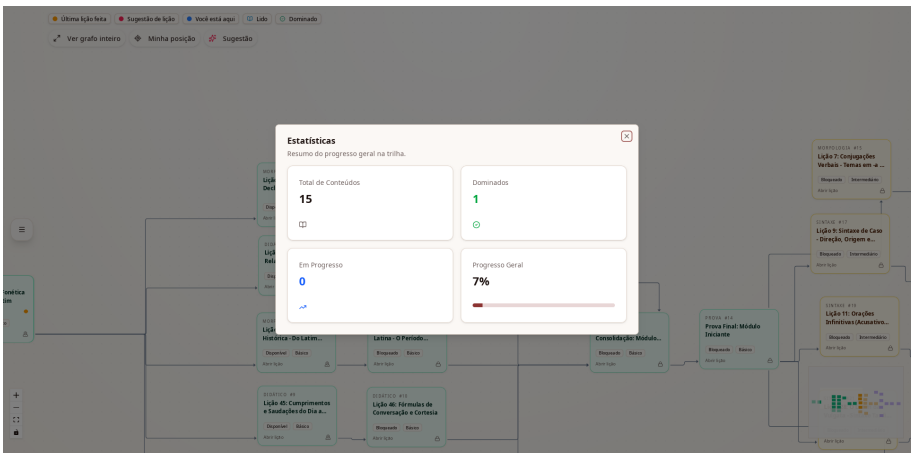
Figura 21 – Listagem geral de recursos multimídia disponíveis ao aluno.



Fonte: O Autor.

A tela de Estatísticas (Figura 22) exibe dados quantitativos sobre a jornada do Aluno, incluindo a porcentagem de progresso geral na trilha, a quantidade total de lições dominadas e o número de tentativas em exercícios.

Figura 22 – Painel de visualização de estatísticas individuais do Aluno.



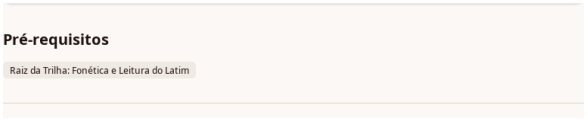
Fonte: O Autor.

5.4.3 Estudo de Lições e Apresentação Adaptativa

A interface de estudo de uma lição materializa os conceitos de Apresentação Adaptativa modelados no projeto.

Figura 23 – Visualização de lição com termos dinâmicos expandidos e exercícios.

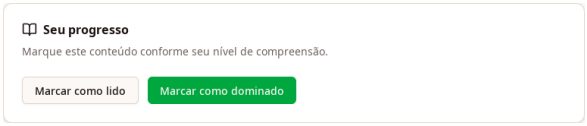
(a) Lição de latim com termos do dicionário



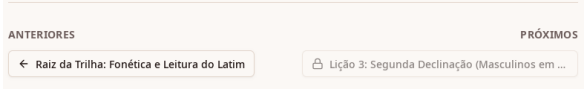
(b) Exibição de stretchtext explicativo

Fonte: O Autor.

Figura 24 – Formulários de resolução de questões integrados à lição.



(a) Questões da lição



(b) Questões e navegação entre lições

Fonte: O Autor.

O fluxo das Figuras 23a e 23b demonstra a técnica de *stretchtext* baseada em componentes reativos em React. No texto de estudo, palavras-chave cadastradas como fragmentos explicativos são sublinhadas com estilo pontilhado. Ao clicar em um termo (ex: *"insula"*), o sistema expande inline o bloco explicativo (morfologia e tradução) sem forçar o redirecionamento ou desorientar o aluno, atendendo à abordagem pedagógica do método de Leitura.

As Figuras 24a e 24b exibem as perguntas vinculadas ao final da lição. A interface expõe botões de navegação para os vizinhos imediatos do nó na trilha (anterior e próximo), os

quais permanecem habilitados ou bloqueados conforme a regra de pré-requisitos avaliada para o aluno.

Figura 25 – Navegação para recursos extras e biografia de autores vinculados.

(a) Recursos multimídia da lição (b) Biografia do autor associado

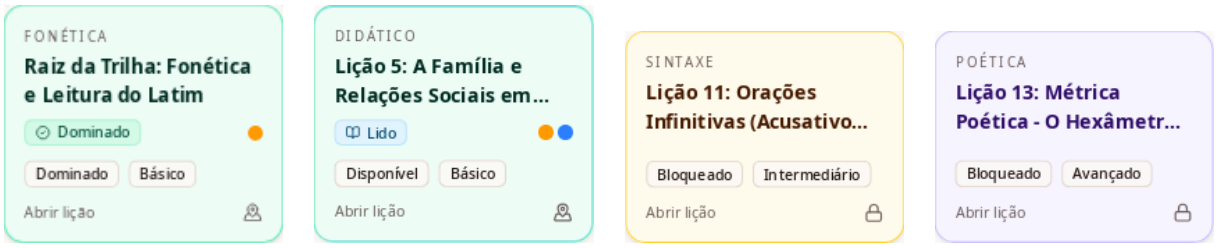
Fonte: O Autor.

As mudanças de contexto navegacionais (‘«change»‘) são exibidas na lição. A Figura 25a mostra os recursos de mídia associados à lição corrente, enquanto a Figura 25b apresenta a biografia do autor clássico que escreveu o texto original sob estudo, enriquecendo o contexto literário e cultural.

5.4.4 Análise dos Estados dos Nós da Trilha

A aparência visual de cada elemento do grafo reflete fielmente o estado do Aluno no banco de dados.

Figura 26 – Estilos visuais dos nós da trilha conforme o progresso do aluno.



(a) Nó Dominado (b) Nó Lido (c) Nó Disponível (d) Nó Avançado

Fonte: O Autor.

Conforme ilustrado na Figura 26, a estilização dos nós segue as seguintes regras de apresentação adaptativa:

- **Nó Dominado (Figura 26a):** Exibido em verde com ícone de *"check"*, indicando que o aluno resolveu com sucesso os exercícios e domina o conceito.
- **Nó Lido (Figura 26b):** Marcado com uma borda azul, indicando que o conteúdo já foi acessado e lido pelo aluno, mas os testes vinculados ainda não foram dominados.
- **Nó Disponível (Figura 26c e Figura 26d):** Apresentado em cores de acordo com a classificação de dificuldade do conteúdo (Básico, Intermediário ou Avançado), sinalizando que o aluno cumpre todos os pré-requisitos e pode acessá-lo.

Figura 27 – Visualização detalhada de uma lição bloqueada por pré-requisitos.

Fonte: O Autor.

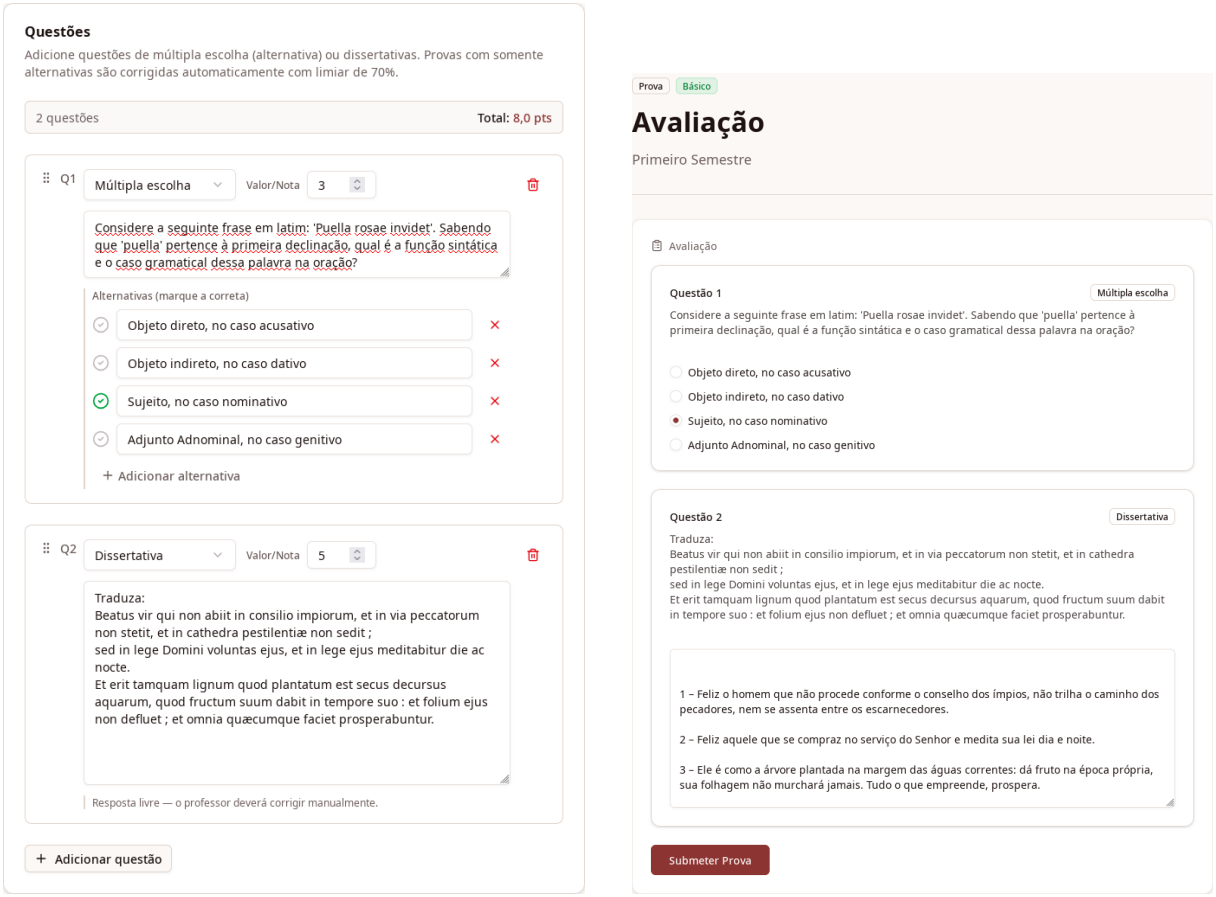
Se o aluno tentar forçar o acesso a uma URL de conteúdo cujos pré-requisitos não foram satisfeitos, o sistema renderiza a tela representada na Figura 27. Esta exibe o ícone de cadeado

e lista os tópicos específicos de gramática que precisam ser dominados previamente no grafo para liberar o acesso, protegendo o rigor estrutural defendido pela escola Gramática-Tradução.

5.4.5 Avaliação, Resolução de Provas e Feedback

A avaliação do aluno é composta por provas e exercícios com correção automática e suporte a questões dissertativas.

Figura 28 – Interfaces de gerenciamento e realização de provas.



(a) Formulário de criação de prova (visão do professor)

(b) Tela de realização de prova (visão do aluno)

Fonte: O Autor.

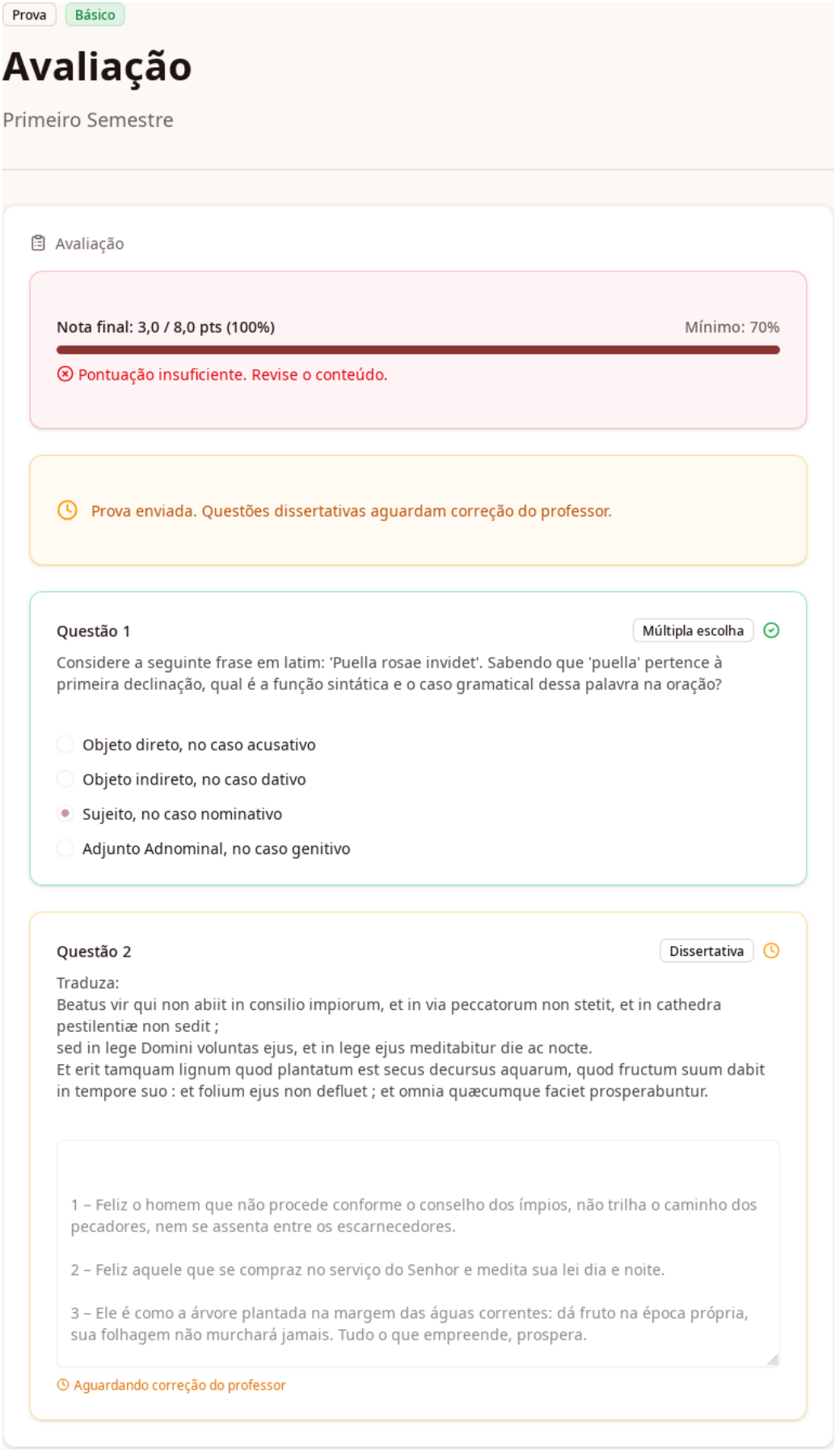
A Figura 28a ilustra a interface de criação de prova sob a perspectiva do professor, enquanto a Figura 28b apresenta como ela se exibe ao aluno para resolução. A prova pode conter questões de múltipla escolha (corrigidas automaticamente pelo motor de adaptação) e questões dissertativas (que demandam tradução manual de trechos e são direcionadas à correção do professor).

Ao submeter a avaliação, as respostas do tipo múltipla escolha recebem correção e feedback visual imediato (Figura 29). Quando a prova contém questões dissertativas, o nó entra em estado pendente até que o professor efetue a correção manual. Posteriormente, o aluno pode consultar a interface da Figura 30 para visualizar a nota final atribuída, comentários detalhados e as justificativas gramaticais inseridas pelo docente.

A centralização do feedback manual é viabilizada pela central de notificações do profes-

sor (Figura 31), que exibe um sinal visual no sino sempre que uma prova com respostas dissertativas é submetida por um aluno. Clicando na notificação, o professor é conduzido ao painel de correção (Figura 32), no qual pode ler a resposta do estudante, comparar com a resposta sugerida, atribuir a nota de 0 a 10 e redigir o comentário pedagógico. Salvar este formulário atualiza o Modelo de Usuário e pode desencadear o desbloqueio em cascata caso o aluno obtenha nota suficiente para dominar a avaliação.

Figura 29 – Tela de feedback e correção automática de alternativas para o aluno.

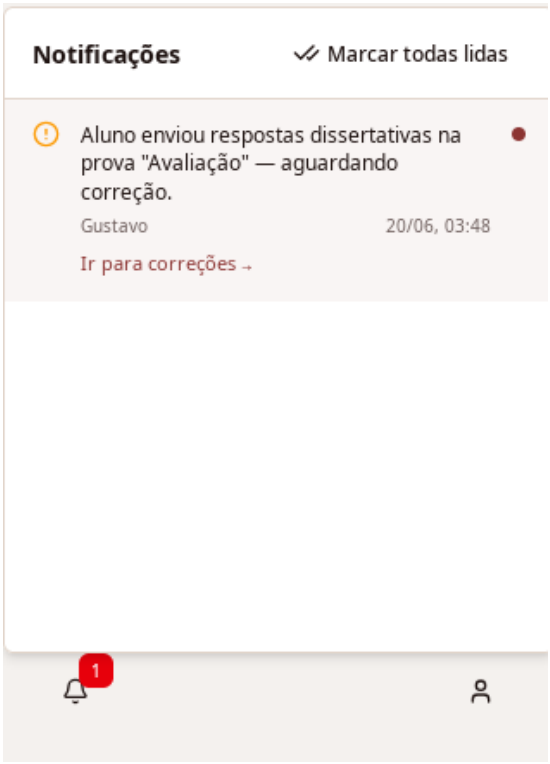


Fonte: O Autor.

Figura 30 – Tela de feedback e comentário do professor exibida ao aluno.

Fonte: O Autor.

Figura 31 – Central de notificações de avaliações pendentes (visão do professor).



Fonte: O Autor.

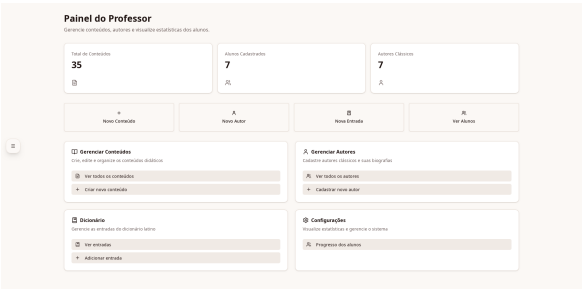
Figura 32 – Tela de avaliação e correção de respostas dissertativas pelo professor.

Fonte: O Autor.

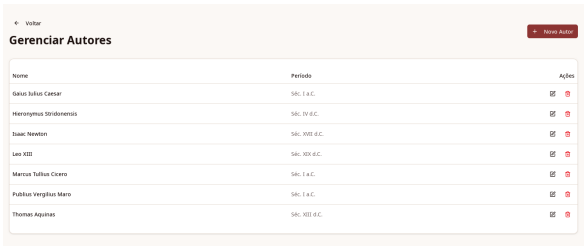
5.4.6 Interfaces Administrativas do Professor

O professor possui um painel administrativo completo para a gestão dos conteúdos do domínio da língua latina.

Figura 33 – Painel administrativo do professor e gerenciador de autores clássicos.



(a) Painel de Controle do Professor



(b) Listagem de Autores cadastrados

Fonte: O Autor.

O Painel de Controle do Professor (Figura 33a) centraliza a criação e gerenciamento dos dados do sistema. A Figura 33b ilustra a tabela de controle de autores de textos clássicos no banco de dados.

Figura 34 – Formulários de gerenciamento (cadastro e edição) de autores.

(a) Cadastro de Autor

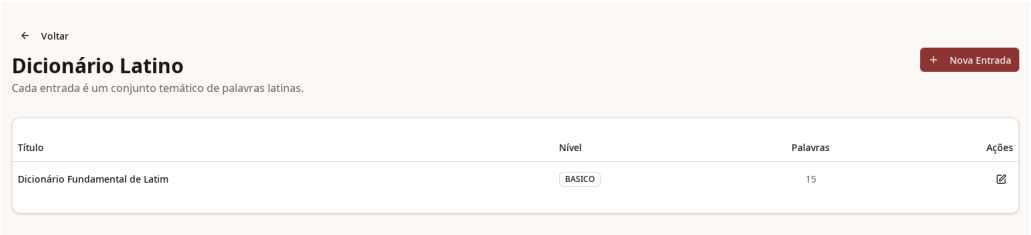
(b) Formulário de Edição de Autor

Fonte: O Autor.

As interfaces das Figuras 34a e 34b são formulários que permitem cadastrar o nome, período histórico e uma breve biografia do autor clássico, alimentando as páginas de auxílio bibliográfico exibidas aos alunos.

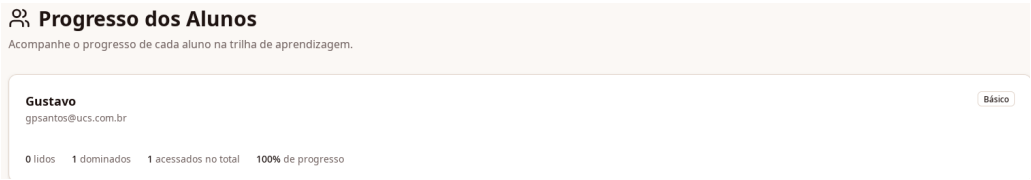
O gerenciamento de verbetes do dicionário é efetuado a partir da listagem representada na Figura 35. O professor pode também visualizar o progresso acumulado de todos os alunos

Figura 35 – Visualização administrativa dos verbetes do dicionário.



Fonte: O Autor.

Figura 36 – Listagem global do progresso e estatísticas dos alunos (visão do professor).



Fonte: O Autor.

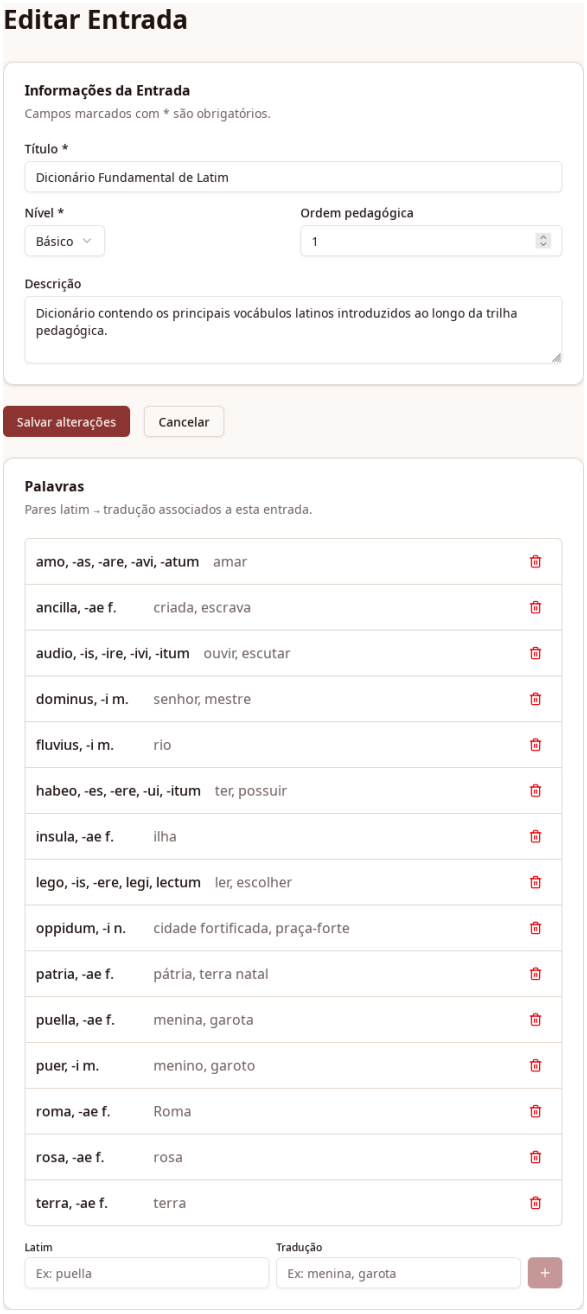
cadastrados por meio do painel ilustrado na Figura 36, inspecionando tentativas, acertos e status das lições para acompanhamento pedagógico.

As Figuras 37a e 37b exemplificam o cadastro e modificação de termos no dicionário (palavra em latim, tradução correspondente e classificação de nível), garantindo a consistência das palavras que serão mapeadas nas lições dos alunos.

O cadastro de um novo elemento na trilha de aprendizagem é operado por meio de um formulário estruturado em abas. Na primeira aba (Figura 38), o professor informa título, descrição, corpo do texto e nível. A segunda aba (Figura 39) permite selecionar graficamente os pré-requisitos que o aluno precisa dominar para desbloquear este nó. A terceira aba (Figura 40) gerencia o banco de questões associado. A quarta aba (Figura 41) realiza a vinculação dos termos gramaticais marcados para a expansão em *stretchtext* na interface do Aluno.

Figura 37 – Formulários de gerenciamento de termos do dicionário.

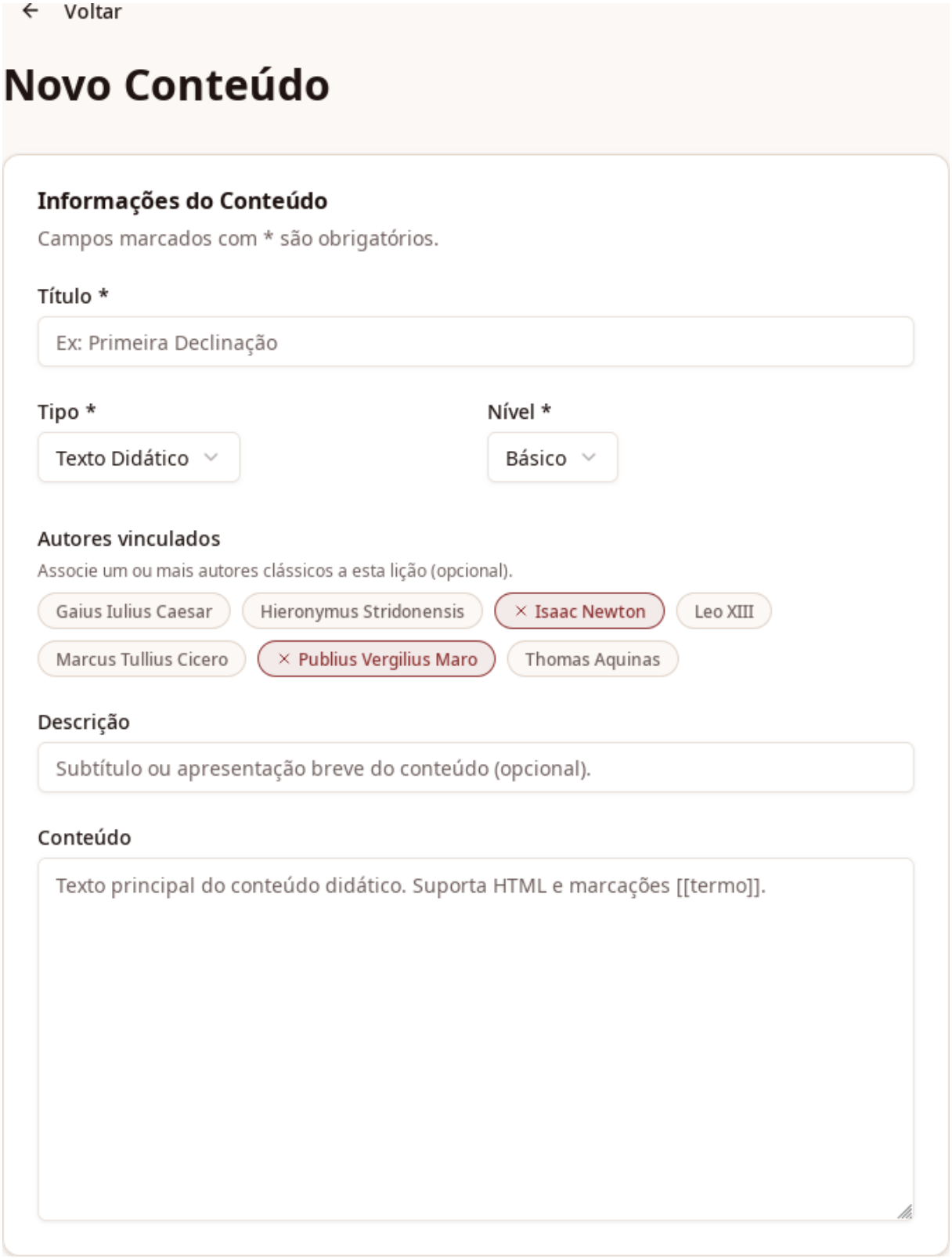
(a) Cadastro de verbete



(b) Edição de verbete

Fonte: O Autor.

Figura 38 – Cadastro de novos conteúdos — aba de metadados.

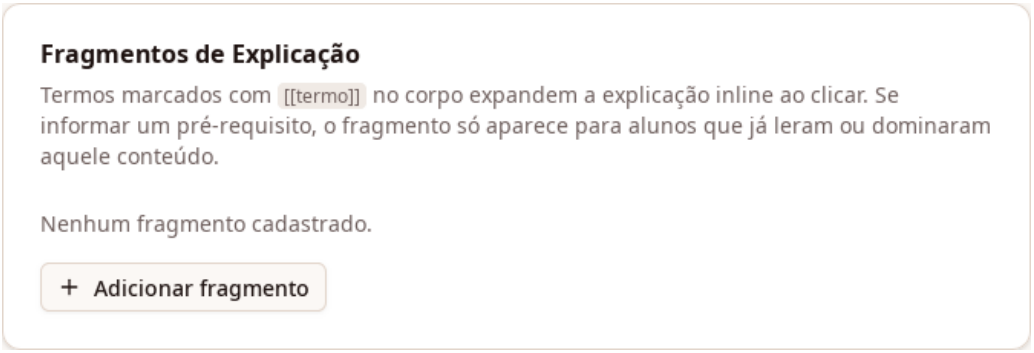


Fonte: O Autor.

Figura 39 – Cadastro de novos conteúdos — aba de seleção de pré-requisitos.

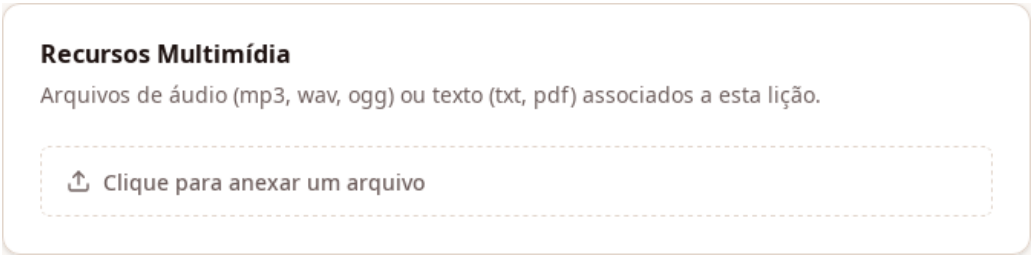
Fonte: O Autor.

Figura 40 – Cadastro de novos conteúdos — aba de questões alternativas.



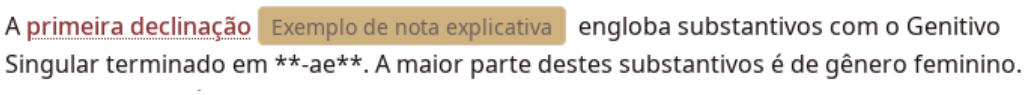
Fonte: O Autor.

Figura 41 – Cadastro de novos conteúdos — aba de estruturação de stretchtext.



Fonte: O Autor.

Figura 42 – Painel de edição detalhada de fragmento de stretchtext.

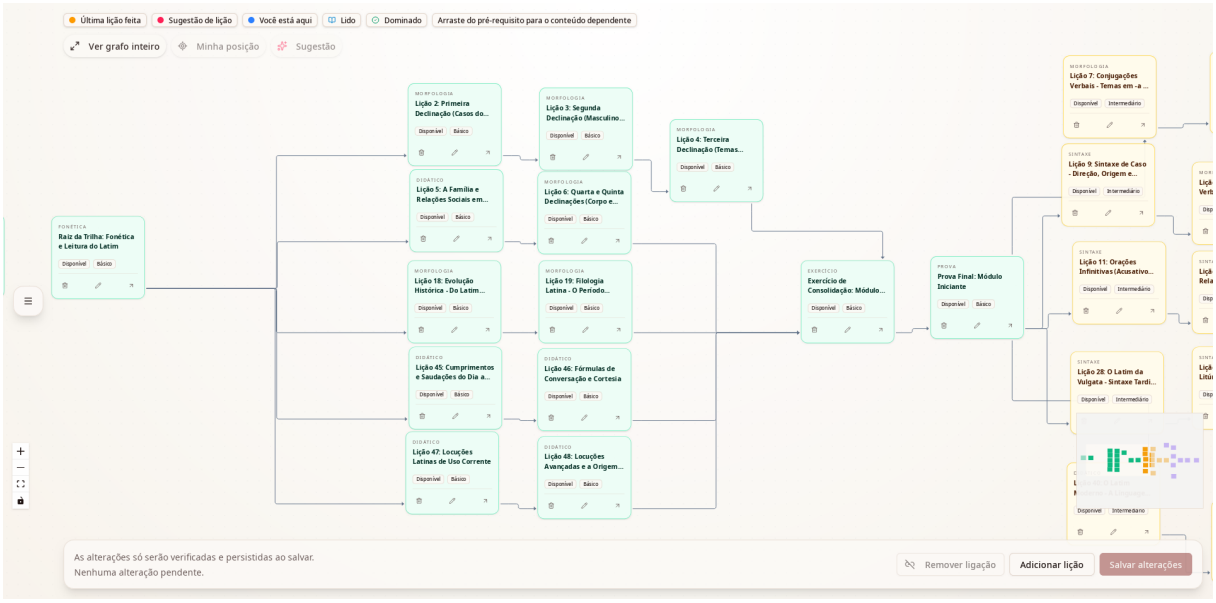


Fonte: O Autor.

5.4.7 Editor do Grafo de Trilha e Detecção de Ciclos na UI

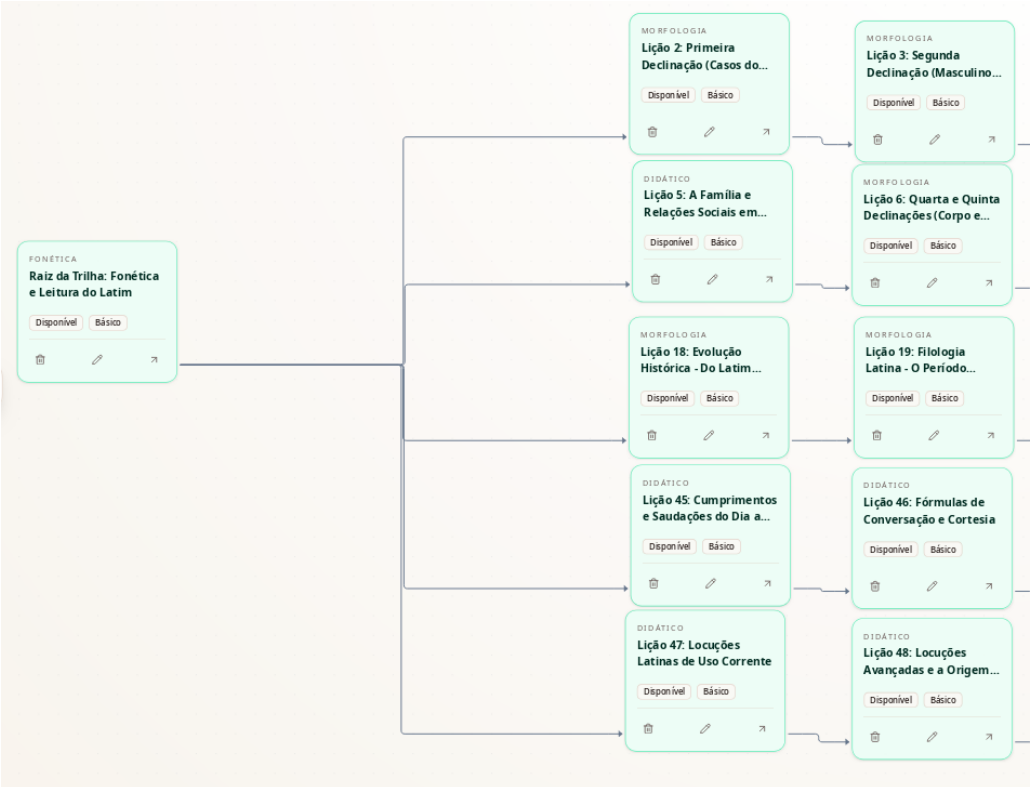
O editor visual do professor representa a consolidação interativa do grafo de aprendizagem do sistema.

Figura 43 – Editor visual do grafo de trilhas (visão do professor).



Fonte: O Autor.

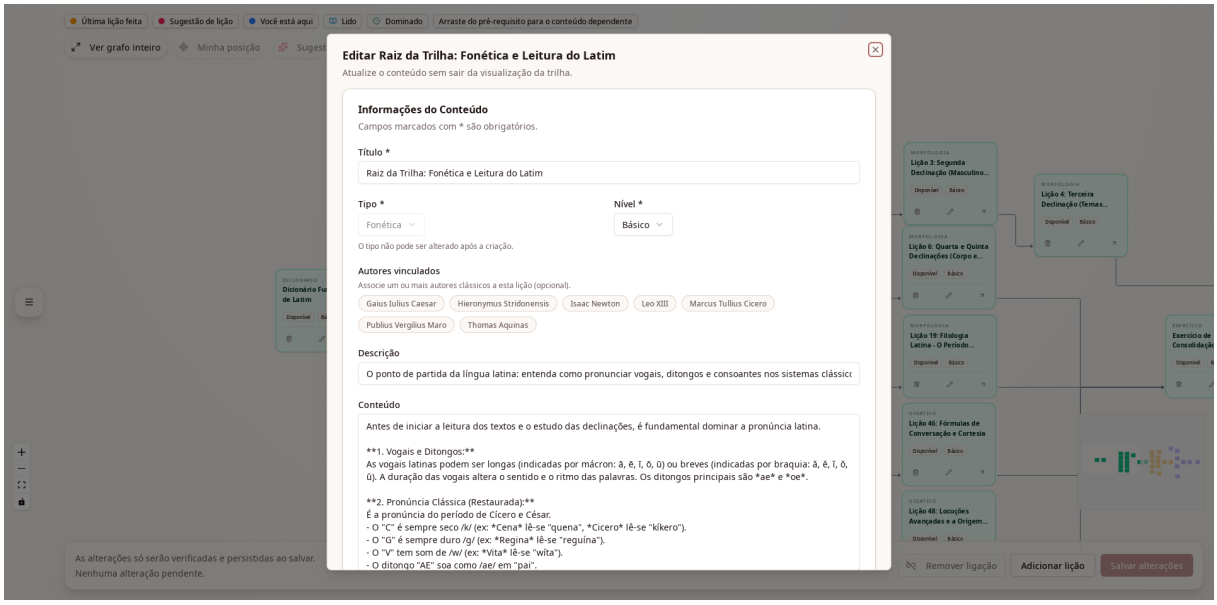
Figura 44 – Visualização de ramificações de múltiplos caminhos no grafo de trilhas.



Fonte: O Autor.

A Figura 43 representa a interface de gerenciamento visual do professor. Diferente da visão do aluno (que oculta ou desabilita links de nós bloqueados), o professor enxerga a totalidade do grafo de aprendizagem e pode arrastar os nós livremente, desenhar arestas e editar o mapeamento. A Figura 44 mostra a capacidade do grafo de ramificar-se em múltiplos caminhos paralelos a partir de um nó básico dominado pelo aluno.

Figura 45 – Menu de atalhos contextuais para edição rápida do nó (professor).



Fonte: O Autor.

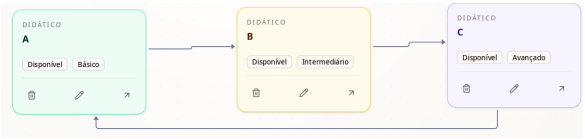
Figura 46 – Criação interativa de arestas de pré-requisitos no editor de trilha (professor).

Fonte: O Autor.

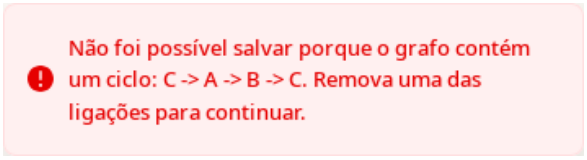
A Figura 45 detalha o menu suspenso contextual que permite ao professor editar rapidamente metadados, ir para a lição ou excluir o nó diretamente a partir do clique no grafo. A Figura 46 exibe a criação visual de uma ligação de pré-requisito arrastando a aresta a partir do nó de origem.

As Figuras 47a e 47b mostram o sistema de prevenção de ciclos em ação na interface. Se o professor tentar ligar dois nós criando um circuito fechado (ex: conectar *"Substantivos I"* a *"Casos Gramaticais"* sendo que o segundo já dependia do primeiro), a interface exibe instantaneamente um balão de alerta (Figura 47a) e o formulário de edição de conteúdo rejeita a ação com um rastro de erro detalhado (Figura 47b), impedindo a gravação de estados inconsistentes.

Figura 47 – Avisos visuais de dependência cíclica inválida na interface.



(a) Detecção de dependência circular no editor



(b) Aviso de ciclo ao salvar as propriedades do nó

Fonte: O Autor.

5.5 VERSIONAMENTO, HOSPEDAGEM E IMPLANTAÇÃO

Para assegurar o controle de versões e a rastreabilidade do protótipo desenvolvido, o repositório do projeto foi versionado com o sistema GitHub[2]. Ao concluir a implementação das funcionalidades fundamentais que validam a proposta conceitual deste trabalho, utilizou-se o comando `git tag` a fim de registrar formalmente o estado da base de código como a versão `1.0.0-mvp` do MVP.

A infraestrutura de hospedagem da aplicação full-stack foi estruturada em ambientes independentes que aproveitam as facilidades de integração contínua modernas. A interface de usuário e a camada de serviços da aplicação Next.js foram implantadas no plano gratuito da plataforma Platform as a Service (PaaS) Vercel. O processo de deploy é automatizado de maneira contínua, uma vez que a Vercel sincroniza o código e inicia novas compilações automaticamente a partir de cada push efetuado na branch `master` do repositório remoto.

De maneira análoga, a persistência de dados e o banco relacional PostgreSQL foram implantados sob o plano gratuito da plataforma Neon. Essa abordagem provê um banco de dados serverless altamente disponível que elimina a sobrecarga administrativa e de infraestrutura física, permitindo a comunicação com a camada de consultas no Next.js de forma integrada e segura.

[2] O código-fonte completo deste projeto está disponível online pelo repositório GitHub: <https://github.com/Gvstavo/tcc-nextjs>.

6 TESTES E AVALIAÇÃO DE USABILIDADE

Este capítulo apresenta a avaliação de usabilidade realizada sobre o protótipo do sistema *Lingua Latina Adaptativa*. Com o objetivo de verificar se o software atende aos requisitos de ergonomia, navegação e clareza de interface sob a perspectiva de alunos e professores, aplicou-se o método de Avaliação Heurística de Nielsen e Molich (NIELSEN; MOLICH, 1990; NIELSEN, 1994).

Descreve-se a metodologia adotada para o ensaio de usabilidade, a avaliação heurística detalhada das principais interfaces e fluxos do sistema, e as discussões conceituais acerca das melhorias propostas para a usabilidade e robustez do protótipo.

6.1 METODOLOGIA DE AVALIAÇÃO HEURÍSTICA

A Avaliação Heurística é um método de inspeção de usabilidade desenvolvido por Jakob Nielsen e Rolf Molich (NIELSEN; MOLICH, 1990). Consiste no escrutínio detalhado da interface de usuário com base em um conjunto consolidado de dez princípios gerais de usabilidade, conhecidos como as Dez Heurísticas de Nielsen (NIELSEN, 1994):

1. **H1 — Visibilidade do estado do sistema:** O sistema deve sempre manter os usuários informados sobre o que está acontecendo por meio de feedbacks apropriados em tempo razoável.
2. **H2 — Correspondência entre o sistema e o mundo real:** O sistema deve falar a linguagem do usuário, com palavras e conceitos familiares, seguindo convenções do mundo real em uma ordem lógica e natural.
3. **H3 — Controle e liberdade do usuário:** O sistema deve fornecer mecanismos de "saída de emergência" para ações acidentais, suportando as funções de desfazer e refazer sem forçar o usuário a fluxos rígidos.
4. **H4 — Consistência e padronização:** Os usuários não devem ter que adivinhar se palavras, situações ou ações diferentes significam a mesma coisa. Devem ser seguidas as convenções da plataforma.
5. **H5 — Prevenção de erros:** Mais do que boas mensagens de erro, o sistema deve evitar a ocorrência de problemas, desabilitando ações inválidas ou exigindo confirmações para ações destrutivas.
6. **H6 — Reconhecimento em vez de memorização:** O usuário não deve ter que memorizar informações de uma parte do fluxo para outra. Objetos, ações e opções devem estar visíveis e acessíveis.

7. **H7 — Flexibilidade e eficiência de uso:** Atalhos e aceleradores devem ser fornecidos para permitir que usuários experientes naveguem mais rapidamente, sem prejudicar os usuários inexperientes.

8. **H8 — Projeto estético e minimalista:** As interfaces devem ser limpas e equilibradas, não contendo informações irrelevantes ou raramente necessárias que compitam com o conteúdo principal.

9. **H9 — Ajude os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros:** As mensagens de erro devem ser expressas em linguagem clara, sem códigos técnicos, indicar o problema com precisão e sugerir uma solução construtiva.

10. **H10 — Ajuda e documentação:** Embora o sistema deva ser intuitivo, pode ser necessário fornecer documentação focada na tarefa do usuário, fácil de consultar e de tamanho moderado.

Para cada tela do protótipo, o analista verificou a aderência da interface a essas heurísticas. Diante de descumprimentos, registrou-se a localização, a heurística violada, a justificativa e o grau de severidade do problema, graduado em uma escala de 1 a 4 (NIELSEN, 1994):

- **Gravidade 1 (Cosmético):** Problema superficial que não afeta a usabilidade, mas prejudica a aparência ou acabamento estético da interface.

- **Gravidade 2 (Leve):** Problema de usabilidade que causa pequenos inconvenientes ao usuário, mas não o impede de completar a tarefa.

- **Gravidade 3 (Grave):** Problema de usabilidade significativo que dificulta a realização das tarefas, frustrando o usuário e podendo levar a desistências.

- **Gravidade 4 (Catastrófico):** Problema crítico que impede o usuário de realizar as tarefas propostas no sistema, exigindo correção obrigatória antes da liberação.

As próximas seções detalham a inspeção heurística executada nas interfaces do aluno e do professor no protótipo desenvolvido.

6.2 AVALIAÇÃO DO PROTÓTIPO POR MÓDULOS E TELAS

Esta seção agrupa as inspeções de usabilidade do sistema nos três principais contextos de utilização: a área geral de estudo do aluno, o fluxo de exames e avaliações, e o ambiente administrativo do professor.

6.2.1 Área do Aluno: Trilha de Aprendizagem, Onboarding e Lições

Este módulo compreende o primeiro contato do estudante com o sistema, a visualização gráfica das lições e a leitura interativa dos conteúdos. O Quadro 8 consolida os resultados da avaliação heurística desse subconjunto de telas.

Quadro 8 – Avaliação heurística do onboarding, trilha e lições do aluno.

Heurística	**Respeitada?**	**Caso não, por quê?**	**Sugestão de melhoria**	**Gravidade**
H1 — Status	Sim	N/A	N/A	N/A
H2 — Mundo Real	Sim	N/A	N/A	N/A
H3 — Liberdade	Não	O aluno não possui um atalho para redefinir sua proficiência ou limpar seu progresso sem intervenção do administrador.	Adicionar botão de "redefinir progresso" nas configurações do aluno.	2
H4 — Consistência	Sim	N/A	N/A	N/A
H5 — Prevenção	Sim	N/A	N/A	N/A
H6 — Reconhecimento	Sim	N/A	N/A	N/A
H7 — Eficiência	Não	Falta suporte a teclas de atalho de navegação no teclado para saltar lições ou expandir verbetes do dicionário.	Implementar atalhos rápidos (ex: `Ctrl + D` para focar busca do dicionário).	2
H8 — Estética	Sim	N/A	N/A	N/A
H9 — Erros	Sim	N/A	N/A	N/A
H10 — Ajuda	Não	O protótipo carece de um tutorial rápido ou documentação de ajuda sobre o funcionamento do grafo adaptativo.	Criar tour interativo (*onboarding walkthrough*) na home do aluno.	2

Fonte: Elaborado pelo autor.

A análise revela que o sistema atende satisfatoriamente à maior parte dos quesitos ergonômicos na trilha. A técnica de *stretchtext* aliada ao dicionário dinâmico atua fortemente a favor da heurística H6 (Reconhecimento em vez de memorização), reduzindo a carga cognitiva sobre o vocabulário clássico. Contudo, há oportunidades de melhorias quanto ao controle (H3) e atalhos de eficiência (H7) para acelerar a navegação.

6.2.2 Módulo do Aluno: Realização de Provas e Feedback

Este fluxo compreende as telas de resolução de exames avaliativos e a leitura dos feedbacks automáticos e manuais de desempenho. O Quadro 9 consolida os resultados obtidos.

Quadro 9 – Avaliação heurística de provas e feedback (aluno).

Heurística	**Respeitada?**	**Caso não, por quê?**	**Sugestão de melhoria**	**Gravidade**
H1 — Status	Sim	N/A	N/A	N/A
H2 — Mundo Real	Sim	N/A	N/A	N/A
H3 — Liberdade	Não	Não há revisão de respostas após submissão enquanto pendente.	Permitir retificação das respostas.	2
H4 — Consistência	Sim	N/A	N/A	N/A
H5 — Prevenção	Sim	N/A	N/A	N/A
H6 — Reconhecimento	Não	Na tela de prova, o aluno não tem acesso rápido ao dicionário integrado, forçando a decoreba dos verbetes.	Habilitar barra lateral do dicionário em formato restrito durante a prova.	3
H7 — Eficiência	Não	Falta atalhos no teclado para selecionar as opções das questões objetivas (ex: teclas 1 a 4).	Mapear teclas numéricas para selecionar alternativas do formulário.	1
H8 — Estética	Sim	N/A	N/A	N/A
H9 — Erros	Sim	N/A	N/A	N/A
H10 — Ajuda	Sim	N/A	N/A	N/A

Fonte: Elaborado pelo autor.

Identificou-se uma violação de Gravidade 3 (Grave) na heurística H6 (Reconhecimento em vez de memorização): a ocultação deliberada do dicionário durante as provas força o aluno a memorizar verbetes, o que pode chocar-se com abordagens pedagógicas de leitura focada. Sugere-se uma melhoria para permitir acesso parcial ou controlado a verbetes estudados na lição correspondente durante a execução do teste.

6.2.3 Área do Professor: Gerenciador de Trilhas e Cadastros

Este subconjunto compreende as telas administrativas de controle do grafo de pré-requisitos, visualização de notificações, correção de avaliações e formulários de cadastro. O Quadro 10 sintetiza a inspeção das interfaces docentes.

Quadro 10 – Avaliação heurística das telas de administração e editor de trilhas (professor).

Heurística	**Respeitada?**	**Caso não, por quê?**	**Sugestão de melhoria**	**Gravidade**
H1 — Status	Sim	N/A	N/A	N/A
H2 — Mundo Real	Sim	N/A	N/A	N/A
H3 — Liberdade	Sim	N/A	N/A	N/A
H4 — Consistência	Sim	N/A	N/A	N/A
H5 — Prevenção	Sim	N/A	N/A	N/A
H6 — Reconhecimento	Sim	N/A	N/A	N/A
H7 — Eficiência	Sim	N/A	N/A	N/A
H8 — Estética	Sim	N/A	N/A	N/A
H9 — Erros	Sim	N/A	N/A	N/A
H10 — Ajuda	Não	Ausência de documentação ou guia explicativo de atalhos rápidos e gestos para manipular o editor do grafo.	Adicionar tela de ajuda flutuante explicando atalhos de mouse/teclado do grafo.	2

Fonte: Elaborado pelo autor.

A área do professor apresentou os melhores resultados de usabilidade ergonômica, impulsionada pelas heurísticas H1, H5 e H9. A interface interativa de grafos simplificou substancialmente as tarefas complexas de gerenciamento curricular. O único ponto deficitário (Gravidade 2) refere-se à carência de ajuda e documentação detalhada sobre gestos de manipulação do grafo (H10).

6.3 ANÁLISE CONSOLIDADA DE USABILIDADE E PREVENÇÃO DE ERROS

O protótipo do sistema *Lingua Latina Adaptativa* demonstrou um elevado nível de robustez ergonômica na inspeção heurística. Um dos destaques analíticos reside na aplicação conjunta das heurísticas **H5 — Prevenção de erros** e **H9 — Recuperação de erros** no gerenciamento do grafo de pré-requisitos curriculares.

Em sistemas tradicionais de gestão de trilha de aprendizagem adaptativa, a modelagem de pré-requisitos baseada em listas textuais ou inputs sequenciais frequentemente induz os administradores a cadastrarem dependências recursivas cíclicas de maneira inadvertida. A inclusão do algoritmo de busca em profundidade de Tarjan para a detecção de Componentes Fortemente Conectados (CFC) atuou como o motor central de usabilidade nas seguintes dimensões:

- **Prevenção ativa de falhas críticas (H5):** O protótipo intercepta qualquer tentativa de persistir uma nova ligação que viole a aciclicidade do Grafo Dirigido Acíclico (DAG). A transação de persistência é bloqueada imediatamente, impedindo que o banco de dados armazene um estado de deadlock de conteúdo. Isso elimina a necessidade de processos complexos de depuração e recuperação pós-gravação de dados no PostgreSQL.
- **Feedback construtivo e diagnóstico do ciclo (H9):** Ao barrar a persistência do ciclo, a mensagem de erro retornada ao usuário não se limita a um código HTTP genérico ou um alerta genérico de falha. A interface renderiza o rastro exato do ciclo detectado (exemplo: “Esta alteração criaria uma dependência cíclica: Substantivos I -> Casos Gramaticais -> Substantivos I”), apontando com precisão onde reside a inconsistência pedagógica e ensinando o professor a diagnosticar e restabelecer a consistência do grafo imediatamente.

Para as etapas futuras de refinamento do sistema para o MVP definitivo, as deficiências identificadas com Gravidade 2 e 3 (sobretudo a indisponibilidade do dicionário integrado durante testes no módulo do aluno e a documentação escassa de controle no editor do professor) serão resolvidas a partir dos planos gerados pelos agentes inteligentes Codex CLI na etapa seguinte de engenharia de software, garantindo um produto estável, inclusivo e pedagogicamente rigoroso.

6.4 AVALIAÇÃO POR USUÁRIOS E RELATO DE EXPERIÊNCIA

Além da inspeção sistemática de usabilidade conduzida com base nas heurísticas de Nielsen, o protótipo do sistema *Lingua Latina Adaptativa* foi submetido à avaliação subjetiva de um usuário — neste caso, uma estudante de Letras Inglês e professora na área de fonoaudio-

logia. O ensaio coletou percepções críticas quanto à facilidade de uso, clareza das interfaces e barreiras operacionais identificadas na prática.

A análise do relato indica que a interface geral do sistema é limpa e intuitiva. O maior destaque reside no editor gráfico de trilha — o fluxograma de pré-requisitos —, que oferece valor bifronte. Sob a ótica docente, o fluxograma atua como uma ferramenta altamente eficaz para planejar e estruturar disciplinas extensas compostas por múltiplos conteúdos. Sob o ponto de vista do discente, o grafo serve como um mapa cognitivo claro que torna a visibilidade do progresso (H1) imediata e de fácil compreensão.

Contudo, observou-se uma desvantagem de escalabilidade do grafo: o crescimento substancial do número de lições tende a expandir excessivamente a representação visual, dificultando a gestão docente e a navegação do aluno. No que tange ao ambiente de leitura das lições, a simplicidade e a ausência de elementos concorrentes de atenção favorecem o foco instrucional, eliminando distrações e facilitando a aquisição de competências linguísticas. Desse modo, para o perfil do estudante, o protótipo cumpre satisfatoriamente o seu propósito pedagógico.

Em contrapartida, as principais dificuldades operacionais concentram-se na área administrativa do professor. O processo de autoria e cadastramento de conteúdos didáticos carece de agilidade, exigindo do docente certos conhecimentos técnicos de Tecnologia da Informação (TI) que extrapolam o domínio de usuários comuns. Essa dependência atua como uma barreira de entrada, demandando refinamento na usabilidade e na automação dos fluxos de carga de dados. Em linhas gerais, a experiência de uso foi considerada satisfatória e interessante, com pendências associadas a ajustes ergonômicos de refino no fluxo docente. O depoimento integral da avaliadora está registrado no Apêndice C.

7 CONSIDERAÇÕES FINAIS

Este trabalho propôs a concepção e modelagem de uma solução de software para uma das mais antigas e complexas disciplinas do humanismo ocidental: o ensino da língua latina. O objetivo central foi demonstrar como uma metodologia estruturada de Engenharia de Software e Hipermídia Adaptativa poderia ser aplicada na modelagem e implementação de um protótipo funcional para esse domínio.

O percurso deste trabalho iniciou-se com a exploração da fundamentação teórica dos Sistemas Hipermídia Adaptativos, analisando os métodos e técnicas de adaptação, bem como as arquiteturas de referência formais, como o Modelo Dexter, AHAM e a metodologia de engenharia de software baseada em UML do Modelo de Munique.

Em seguida, investigou-se o domínio do problema: o ensino de Latim no contexto brasileiro. Foi demonstrado que o ensino da disciplina vive uma tensão histórica entre a abordagem de Gramática-Tradução – focada no domínio exaustivo da estrutura morfológica e sintática como um fim em si mesmo – e os métodos de Leitura ou Direto, que priorizam a aquisição de vocabulário e a compreensão de textos pelo contexto. Concluiu-se que ambas as abordagens, em suas formas tradicionais, sofrem da limitação *one-size-fits-all*, sendo incapazes de se ajustar aos diferentes ritmos e necessidades de aprendizado dos alunos.

Após a consideração da bibliografia fundamental na área de HA e a definição clara do problema pedagógico, foi possível modelar um sistema adaptativo que atende às necessidades pedagógicas envolvidas. O Capítulo 4 apresentou esta modelagem, aplicando a metodologia UWE.

A solução proposta, *Lingua Latina Adaptativa*, organiza a dicotomia pedagógica em termos de modelagem. Através do design conceitual, o Modelo de Domínio foi estruturado para respeitar a complexidade da língua, separando `ConceitoGramatical` (Morfologia, Sintaxe, etc.) de `Texto` (Didático, De Autoria), e ligando-os através da relação `é_prerequisito_de`.

Esta estrutura permite ao Modelo de Adaptação aplicar diferentes estratégias:

1. Para o aluno que prefere o rigor do método de Gramática-Tradução, o sistema pode usar a Navegação Adaptativa para garantir que nenhum texto ou prova seja acessado antes que seus pré-requisitos gramaticais sejam marcados como "dominados"no Modelo de Usuário.

2. Para o aluno que prefere o método de Leitura, o sistema pode usar a Apresentação Adaptativa para exibir textos imediatamente, mas inserindo dinamicamente explicações ou glossários apenas para as estruturas gramaticais e vocábulos que o Modelo de Usuário identifica como desconhecidos.

Desta forma, a modelagem proposta evidencia a adequação da Hipermídia Adaptativa como base para estruturar percursos de estudo personalizados, sem que isso represente, neste momento, uma validação pedagógica conclusiva em contexto real de uso.

Com a conclusão deste trabalho, o desenvolvimento do protótipo funcional demonstrou de forma prática o mérito da metodologia adotada e da forma como a especificação foi convertida em software. Sob a perspectiva da modelagem, a representação conceitual baseada na metodologia UWE foi traduzida para a arquitetura da aplicação, evidenciando a consistência dos Modelos de Domínio, Usuário e Adaptação para sustentar tanto a apresentação adaptativa quanto a navegação adaptativa.

No âmbito da implementação, a materialização do sistema como uma aplicação *full-stack* em Next.js, integrada ao banco de dados relacional PostgreSQL e hospedada na plataforma Vercel, mostrou a viabilidade técnica das escolhas realizadas para sustentar o processamento dinâmico exigido pelo protótipo. A simplicidade e agilidade do processo de implantação contínua a partir do repositório Git reforçaram a adequação de uma arquitetura orientada à modularidade no ecossistema do React.

Por fim, a etapa de testes e avaliação heurística consolidou a análise da qualidade ergonômica do sistema, apontando aspectos de usabilidade e prevenção de falhas relevantes para a evolução do protótipo. A utilização do algoritmo de Tarjan para a detecção de ciclos no grafo de pré-requisitos evidenciou como a lógica algorítmica contribui para a consistência operacional do ecossistema e para a redução de erros críticos (H5) e falhas de diagnóstico (H9). Assim, o MVP cumpre com êxito a proposta desta etapa do trabalho, consolidando a base metodológica e técnica necessária para investigações futuras.

7.1 TRABALHOS FUTUROS

A partir dos resultados obtidos com o desenvolvimento e avaliação do protótipo, abrem-se diversas perspectivas para investigações e expansões futuras da plataforma. Destacam-se as seguintes propostas de continuidade de pesquisa:

- **Migração para Banco de Dados de Grafos**: Embora o banco de dados relacional PostgreSQL tenha atendido de forma satisfatória aos requisitos do MVP, a estrutura inerente do currículo adaptativo e da trilha de aprendizagem é essencialmente um grafo direcionado de dependências. Como trabalho futuro, propõe-se a substituição do paradigma relacional por um banco de dados de grafos nativo (como Neo4j ou similar). Essa mudança simplificaria substancialmente as consultas recursivas de caminhos e pré-requisitos, otimizando o desempenho de algoritmos como o de Tarjan diretamente sobre a persistência dos nós e arestas do grafo.
- **Integração de Ferramenta de RAG (*Retrieval-Augmented Generation*)**: Para expandir

as capacidades de ajuda e tutoria inteligente da plataforma (heurística H10), propõe-se a integração de um sistema de Geração Aumentada de Recuperação (RAG). Essa ferramenta poderia conectar um modelo de linguagem de grande porte (LLM) a uma base de conhecimento curada contendo gramáticas de latim, verbetes dicionários extensivos e textos clássicos traduzidos. Com isso, os alunos poderiam tirar dúvidas específicas sobre morfologia e sintaxe clássica em linguagem natural de forma dinâmica e contextualizada, enriquecendo a experiência de aprendizado autodidata.

- **Validação pedagógica com alunos e professores**: Realização de estudos com participantes reais para avaliar, em contexto educacional, a percepção de utilidade, a adequação pedagógica e o impacto da solução adaptativa no processo de ensino e aprendizagem.

REFERÊNCIAS

ALMEIDA, N. M. d. **Gramática Latina**. [S.l.: s.n.], 2011.

BRA, P. D.; HOUBEN, G.-J.; WU, H. **AHAM: a Dexter-based reference model for adaptive hypermedia**. [S.l.: s.n.], 1999. 147–156 p.

BRUSILOVSKY, P. **Methods and techniques of adaptive hypermedia**. [S.l.]: Springer, 1996. v. 6. 87–129 p.

____. **Adaptive hypermedia**. [S.l.]: Springer, 2001. v. 11. 87–110 p.

BRUSILOVSKY, P.; MILLÁN, E. **User models for adaptive hypermedia and adaptive educational systems**. [S.l.]: Springer, 2007. 3–53 p.

BUGAY, E. L. *et al.* O modelo aham-mi: modelo de hipermídia adaptativa utilizando inteligências múltiplas. Florianópolis, SC, 2006.

BUSH, V. As we may think (como podemos pensar). **Hipertextus Revista Digital**, v. 14, 2016.

BUSH, V. *et al.* **As we may think**. [S.l.]: [S. l.], 1945. v. 176. 101–108 p.

DeusData. **Codebase-Memory: High-Performance Code Intelligence Knowledge Graph MCP Server**. 2026. Disponível em: <https://github.com/DeusData/codebase-memory-mcp>. Acesso em: 20 jun. 2026.

ERNOUT, A.; THOMAS, F. **Syntaxe Latine**. [S.l.]: Klincksieck, 2008.

FERNANDES, T. *et al.* A tradução e o ensino do latim. 2012.

GARLATTI, S.; PRIE, Y. **Adaptation et personnalisation dans le Web sémantique**. [S.l.: s.n.], 2004.

GitHub Spec Kit. **Spec Kit — Workflows Reference**. 2026. Disponível em: <https://github.github.com/spec-kit/reference/workflows.html>. Acesso em: 08 maio 2026.

GRAVELLE, J. **jCodeMunch: Token-Efficient MCP Source Code Explorer via tree-sitter AST**. 2026. Disponível em: <https://github.com/jgravelle/jcodemunch-mcp>. Acesso em: 20 jun. 2026.

GROSS, C.; Akşimşek, D.; STEPINSKI, A. **Hypermedia Systems**. [S.l.]: INDEPENDENTLY PUBLISHED, 2023.

HALASZ, F.; SCHWARTZ, M. The dexter hypertext reference model. **Communications of the ACM**, ACM New York, NY, USA, v. 37, n. 2, p. 30–39, 1994.

HALL, W.; DAVIS, H.; HUTCHINGS, G. **Rethinking hypermedia: the Microcosm approach**. [S.l.]: Springer Science & Business Media, 2012. v. 4.

HECK, M. R. D. O ensino do latim no brasil: objetivos, método e tradição. 2013.

HEVNER, A. R. *et al.* Design science in information systems research. **MIS Quarterly**, Management Information Systems Research Center, University of Minnesota, v. 28, n. 1, p. 75–105, 2004.

JONES, P. V.; SIDWELL, K. C. **Aprendendo Latim: Gramática, Vocabulário, Exercícios e Textos**. 1. ed. São Paulo: Odysseus Editora, 2012. Tradução e supervisão da edição brasileira. Título original: Reading Latin (1986). ISBN 978-85-7876-019-9.

KOCH, N. **A comparative study of methods for hypermedia development**. [S.l.], 1999.

KOCH, N.; MANDEL, L. Using uml to design hypermedia applications. **Ludwig-Maximilians-University Munich, Institute of Computer Science**, 1999.

KOCH, N.; WIRSING, M. Software engineering for adaptive hypermedia applications. In: **8th International Conference on User Modeling, Sonthofen, Germany**. [S.l.: s.n.], 2001.

____. The munich reference model for adaptive hypermedia applications. In: SPRINGER. **International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems**. [S.l.], 2002. p. 213–222.

LEITE, L. R.; BARBOSA, M. *et al.* O ensino de língua latina no brasil: percursos e perspectivas. **Classica: Revista Brasileira de Estudos Clássicos**, Sociedade Brasileira de Estudos Clássico (SBEC), v. 27, n. 2, p. 53–77, 2014.

LOURENÇO, F. **Nova gramática do Latim**. [S.l.: s.n.], 2021.

NELSON, T. H. **Complex information processing: a file structure for the complex, the changing and the indeterminate**. [S.l.: s.n.], 1965. 84–100 p.

NIELSEN, J. Heuristic evaluation. In: ____. **Usability Inspection Methods**. USA: John Wiley & Sons, Inc., 1994. p. 25–62. ISBN 0471018775.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 1990. (CHI '90), p. 249–256. ISBN 0201509326. Disponível em: <https://doi.org/10.1145/97243.97281>.

Object Management Group. **Object Constraint Language, Version 2.4**. [S.l.], 2014. Disponível em: <http://www.omg.org/spec/OCL/2.4>. Acesso em: 22 abr. 2026.

OpenAI. **Codex | Parceiro de programação com IA da OpenAI | OpenAI**. 2026. Disponível em: <https://openai.com/pt-BR/codex/>. Acesso em: 20 jun. 2026.

Oraios AI. **Serena: Semantic Retrieval and Editing MCP Toolkit for Coding Agents**. 2026. Disponível em: <https://github.com/oraios/serena>. Acesso em: 20 jun. 2026.

ORBERG, H. H. **Lingua Latina - Latine Disco, Student's Manual**. Newburyport, MA: Focus Publishing/R Pullins, 1999. (Lingua Latina).

____. **Lingua Latina - Familia Romana**. Newburyport, MA: Focus Publishing/R Pullins, 2011. (Lingua Latina).

PALAZZO, L. A. M. **Modelos proativos para hipermídia adaptativa**. [S.l.: s.n.], 2000.

PISKALA, D. B. **Spec-Driven Development: From Code to Contract in the Age of AI Coding Assistants**. 2026. ArXiv:2602.00180 [cs.SE]. Submetido ao AIWare 2026. Disponível em: <https://arxiv.org/abs/2602.00180>. Acesso em: 08 maio 2026.

PUGA, S. G. **Sistemas Hipermídia adaptativos para a educação baseada na web: uma visão semiótica.** Tese (Doutorado) — Universidade de São Paulo, 2008.

REGEŠ, J. **AI Distiller: Ultra-Fast Code Distillation for Intelligent Context Extraction**. 2026. Disponível em: <https://github.com/janreges/ai-distiller>. Acesso em: 20 jun. 2026.

RICHARDS, M.; FORD, N. **Fundamentals of Software Architecture: A Modern Engineering Approach**. 2nd. ed. Sebastopol, CA: O'Reilly Media, Inc., 2025. ISBN 978-1-098-17551-1.

SZYMKOWIAK, P. **RTK: CLI Proxy for Reducing LLM Token Consumption in Dev Workflows**. 2026. Disponível em: <https://github.com/rtk-ai/rtk>. Acesso em: 20 jun. 2026.

TAGHAVI, P.; BHAVANI, S. **Spec Kit Agents: Context-Grounded Agentic Workflows**. 2026. ArXiv:2604.05278 [cs.SE]. Disponível em: <https://arxiv.org/abs/2604.05278>. Acesso em: 08 maio 2026.

TAKIKAWA, F. K. **Arquitetura de sistemas hipermídia adaptativos baseada em atributos de qualidade.** Tese (Doutorado) — Universidade de São Paulo, 2010.

TARJAN, R. Depth-first search and linear graph algorithms. **SIAM journal on computing**, SIAM, v. 1, n. 2, p. 146–160, 1972.

The PostgreSQL Global Development Group. **PostgreSQL: The World's Most Advanced Open Source Relational Database**. 2026. Disponível em: <https://www.postgresql.org/>. Acesso em: 20 jun. 2026.

Vercel. **Next.js: The React Framework for the Web**. 2026. Disponível em: <https://nextjs.org/>. Acesso em: 20 jun. 2026.

WU, H. *et al.* Adaptation control in adaptive hypermedia systems. In: SPRINGER. **International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems**. [S.l.], 2000. p. 250–259.

APÊNDICE A – MODELO DE DECLARAÇÃO DE USO DE IA

Durante a preparação deste trabalho, o autor utilizou o Codex CLI versão 0.141.0 para geração de parte do código. Após o uso desta ferramenta, o autor revisou e editou o conteúdo em conformidade com o método científico e assume total responsabilidade pelo conteúdo da publicação.

APÊNDICE B – OTIMIZAÇÃO DE CONTEXTO E EFICIÊNCIA DE TOKENS

A utilização de modelos de linguagem em larga escala (LLM) no desenvolvimento de software acarreta custos computacionais e financeiros relacionados diretamente ao consumo de tokens no contexto da conversação. O Codex CLI, operando como interface textual (TUI) integrada de desenvolvimento agêntico, estabelece estratégias para minimizar esse desperdício por meio do protocolo Model Context Protocol (MCP) e de utilitários especializados. Neste apêndice, descrevem-se as ferramentas adotadas para reduzir a carga de leitura de arquivos e saídas de terminal:

1. **jCodeMunch:** O *server* MCP jCodeMunch (GRAVELLE, 2026) realiza a indexação estrutural da base de código via árvores de sintaxe abstrata (AST) com o analisador *tree-sitter*. Em vez de submeter arquivos íntegros para extrair um escopo limitado de código, a ferramenta possibilita ao agente consultar e recuperar apenas os símbolos e declarações de funções exatos necessários, resultando em reduções superiores a 95% no consumo de tokens em tarefas de leitura e exploração.

2. **Serena:** Focada na navegação semântica de código, a ferramenta Serena (Oraios AI, 2026) expõe recursos baseados no protocolo Language Server Protocol (LSP) para as linguagens do projeto. Permite ao agente de desenvolvimento realizar consultas do tipo *Go-to-Definition* (ir para a definição) e *Find-All-References* (encontrar todas as referências) em nível de símbolo. Isso elimina a necessidade de buscas manuais por varredura textual (*Grep*) ao longo de múltiplos arquivos colaterais.

3. **Codebase-Memory:** O Codebase-Memory (DeusData, 2026) opera como um grafo de conhecimento relacional de alta performance e baixa latência que mapeia a arquitetura e dependências do repositório em mais de 150 linguagens. Ao estruturar relacionamentos como chamadas, importações e caminhos de API, permite ao agente deduzir dependências arquiteturais por meio de consultas conceituais (inclusive Cypher), reduzindo em até 99% o volume de contexto transferido em relação à exploração tradicional arquivo-por-arquivo.

4. **IA Distiller:** O utilitário IA Distiller (REGEŠ, 2026) atua na destilação semântica de código-fonte de forma ultra-rápida. Ele extrai de maneira inteligente apenas assinaturas de API públicas, tipos de dados e cabeçalhos de classes, ignorando implementações privadas e corpos de métodos. Isso gera um sumário comprimido que reduz o volume do repositório em 90% a 98% para contextualização global do agente sem estourar o limite de tokens.

5. **RTK (Rust Token Killer):** Desenvolvido como um binário nativo em Rust, o RTK (SZYMKOWIAK, 2026) opera como um *CLI proxy* transparente que intercepta a saída de comandos de terminal executados pelo agente (como compilações, testes e listagens de diretórios). A ferramenta comprime e simplifica os registros de saída eliminando redundâncias, omitindo dados de sucesso e isolando erros específicos antes que esses alcancem a janela de contexto do modelo, reduzindo o consumo de tokens na execução de comandos em 60% a 90%.

APÊNDICE C – DEPOIMENTO DE AVALIAÇÃO DE USABILIDADE POR USUÁRIO

A seguir, apresenta-se o relato integral e sem modificações fornecido pela testadora Letícia Silva (estudante de Letras Inglês e professora na área de fonoaudiologia), após interagir com os fluxos do aluno e do professor no protótipo desenvolvido:

> O sistema contém uma interface limpa e intuitiva. O maior destaque é a trilha, um fluxograma customizável que permite a conexão entre várias lições. Para um professor, é útil especialmente para a criação e organização de disciplinas muito extensas, com várias lições. Para o aluno, é uma representação gráfica de fácil entendimento do próprio progresso. A grande desvantagem da trilha parece ser que, quanto mais cresce o número de lições, mais o fluxograma se expande, e um fluxograma muito grande pode ser difícil para um professor administrar, e pode fazer com que o aluno se perca. A interface das lições é muito simples, sem rodeios, o que faz sentido para um sistema tão focado em leitura, tornando o aprendizado mais fácil, pois não há distrações. Referente ao aluno, o sistema cumpre seu propósito.
>
> As maiores dificuldades do sistema estão na interface do professor: falta maior refino na criação dos conteúdos didáticos, pois o processo atualmente é pouco ágil e dependente de certos conhecimentos de TI que o usuário comum talvez não possua.
>
> No geral, o sistema fornece uma experiência satisfatória e interessante, necessitando apenas de mais alguns refinamentos referentes à usabilidade.