Aurora v2: Reestruturação Arquitetural e Fomento de um Ecossistema Colaborativo

Proposta de Trabalho de Conclusão de Curso

1. Introdução

O Aurora é uma plataforma de código aberto voltada para o gerenciamento das disciplinas cursadas por discentes de graduação. Lançada em 27 de junho de 2025 e direcionada inicialmente aos alunos do Bacharelado em Ciência da Computação (BCC) do Instituto de Matemática Estatística e Ciência da Computação da Universidade de São Paulo (IME-USP), a ferramenta obteve rápida adoção. Em 14 de maio de 2026, a ferramenta possuía 308 usuários registrados no total, o que incluía 54% do corpo discente matriculado no curso.

O desenvolvimento inicial ocorreu ao longo de 2024 e meados de 2025, conduzido por uma equipe multidisciplinar de 11 estudantes da universidade. Pelo ponto de vista técnico, a primeira versão foi concebida sob uma arquitetura monolítica baseada no framework Laravel, integrada a um frontend em React por meio da biblioteca Inertia. Contudo, devido à curva de aprendizado da equipe com projetos de tal escala, decisões arquiteturais equivocadas comprometeram a escalabilidade, a estabilidade e, sobretudo, a manutenibilidade do código-fonte. Esse acúmulo de dívida técnica elevou consideravelmente a barreira de entrada para novos contribuidores, resultando na estagnação das atualizações da plataforma desde o seu lançamento.

Diante dessa problemática, este Trabalho de Conclusão de Curso propõe o desenvolvimento da segunda versão do Aurora. O escopo principal concentra-se na reescrita estrutural da aplicação, aplicando princípios e padrões de projeto que garantam maior qualidade de software. Com essa nova fundação tecnológica, o trabalho busca viabilizar a evolução contínua da plataforma, a introdução de novas funcionalidades e, a partir do novo sistema, fomentar um ecossistema colaborativo de desenvolvimento de software livre.

2. Motivação e Justificativa

Uma ferramenta capaz de auxiliar os discentes da Universidade de São Paulo no planejamento estratégico de suas trajetórias acadêmicas tem um caso de uso claro e relevante dentro da instituição. Em cursos com estruturas curriculares altamente flexíveis, o Aurora permite a visualização a longo prazo das possibilidades de grade, o acompanhamento de trilhas de especialização e o mapeamento de pré-requisitos. Dentre os benefícios concretos dessa abordagem, destacam-se a viabilização de semestres atípicos, para a realização de estágios ou intercâmbios, sem prejuízo à data de conclusão do curso, a orientação na escolha de disciplinas optativas para

a formação de perfis específicos, e a estruturação de planos de estudos rigorosos para alunos próximos ao prazo máximo de integralização ou em processo de reingresso.

A aplicabilidade de uma solução com esse escopo transcende o Bacharelado em Ciência da Computação. No próprio IME-USP, bem como em outras unidades da universidade, existem diversos cursos caracterizados por grades adaptáveis e pela presença de habilitações específicas. O Bacharelado em Matemática Aplicada e Computacional (BMAC), por exemplo, possui habilitações que alteram consideravelmente a estrutura das disciplinas da graduação. Além disso, há diretrizes institucionais para a ampliação do sistema de especializações no curso de Estatística [1]. Nesses cenários, a expansão do Aurora representaria um recurso valioso para o planejamento acadêmico de um número ainda maior de estudantes.

Adicionalmente, o acompanhamento preciso dos créditos necessários para a colação de grau é um desafio histórico para o corpo discente. Atualmente, no IME-USP, o processo de conferência para formatura possui etapas significativamente manuais, nas quais o aluno preenche um formulário indicando a suposta integralização dos créditos, que são então validados pelo Serviço de Graduação [2]. Embora existam ferramentas institucionais no sistema JúpiterWeb para esse acompanhamento [3], estas carecem de clareza quanto aos critérios atendidos em sua totalidade e não oferecem suporte nativo a especializações internas, como as trilhas do BCC. Nesse cenário, ferramentas estudantis foram construídas no passado para sanar esse problema, como o Yggdrasil2 [4], mas pela ausência de uma comunidade para manter a solução a longo prazo e pela falta de conexão institucional para garantir os dados da plataforma, ela se tornou obsoleta com o tempo.

Diante da evidente utilidade acadêmica do Aurora e de seu potencial de expansão, é imperativa a manutenção de seu funcionamento e a garantia de sua evolução contínua. Contudo, a ausência de atualizações no projeto nos últimos seis meses levanta preocupações quanto à sua longevidade. Essa estagnação decorre, majoritariamente, das limitações arquiteturais da sua versão inicial. Dessa forma, a principal motivação deste projeto é reconstruir a base tecnológica do Aurora, proporcionando a infraestrutura e a manutenibilidade necessárias não apenas para a retomada do desenvolvimento de novas funcionalidades, mas também para reengajar e consolidar uma comunidade ativa de alunos contribuidores.

3. Objetivos

3.1. Objetivo Geral

O objetivo principal deste Trabalho de Conclusão de Curso é desenvolver a segunda versão do sistema Aurora, migrando sua base tecnológica para uma arquitetura moderna, modular e de fácil manutenção, além de estruturar iniciativas que fomentem a criação de uma comunidade ativa de contribuidores para o projeto entre os alunos do IME-USP.

3.2. Objetivos Específicos

Para alcançar o objetivo geral, definem-se as seguintes metas específicas:

Reestruturação Tecnológica : Reescrever o sistema utilizando linguagens modernas, mais comumente conhecidas por alunos de graduação em Ciência da Computação e que promovam maior legibilidade. Para o backend, prevê-se a linguagem Python e o framework FastAPI. A base de dados será migrada para PostgreSQL, justificando-se por seu robusto suporte a consultas complexas (necessárias para modelagem de currículos e pré-requisitos) e sua excelente integração com as bibliotecas ORM adotadas. Para o frontend, prevê-se a adoção de TypeScript em todo o código, assim como a utilização de bibliotecas como TanStack Query, Redux e React Router para simplificar o código.

Garantia de Qualidade e Manutenibilidade : Adotar o paradigma de tipagem estática em toda a aplicação. Essa prática reduz a incidência de bugs [5] e diminui a barreira de entrada para novos desenvolvedores, pois facilita a compreensão de APIs [6]. Além disso, o projeto incluirá a implementação de testes automatizados e a configuração de uma esteira de Integração Contínua (CI) para garantir a integridade do código a cada nova contribuição.

Desenvolvimento de Biblioteca Base (pyreplicado) : Desenvolver uma biblioteca modular e independente, denominada pyreplicado, para abstrair e facilitar o consumo de dados do banco de dados replicado do Júpiter ("Replicado") disponibilizado pelo Apoio Institucional do IME-USP. Essa biblioteca visa não apenas atender ao Aurora, mas servir como infraestrutura para futuros sistemas acadêmicos.

Incentivo a novos contribuidores : Promover iniciativas para disseminar o conhecimento sobre o sistema e atrair novos desenvolvedores. Com inspiração em eventos como hackathons, prevê-se a organização de competições e atividades *gamificadas* para engajar a comunidade e impulsionar as contribuições ao projeto.

Inclusão de Novo Curso De maneira a validar a nova arquitetura construída e funcionalidade do sistema, planeja-se realizar a adição da grade de um segundo curso do IME USP ao Aurora.

4. Delimitações e Trabalhos Futuros

Tendo em vista que o foco primordial desta proposta é estritamente estrutural, algumas frentes de desenvolvimento foram deliberadamente reservadas para etapas posteriores. Como perspectivas de trabalhos futuros, vislumbra-se a atualização e a refatoração da atual interface gráfica, bem como a implementação final de novas funcionalidades voltadas ao usuário. Além disso, a arquitetura modular concebida e a construção da biblioteca pyreplicado servirão de alicerce para a expansão do ecossistema, possibilitando a futura reconstrução de outros sistemas, como o MatrUSP [7]. Por fim, no âmbito das operações de software, prevê-se que o projeto possa evoluir futuramente com a implementação de uma esteira de Entrega Contínua (CD) para deploy automatizado.

5. Metodologia

5.1. Fundamentação Teórica e Definição Arquitetural

Revisão bibliográfica acerca de padrões de projeto (Design Patterns), arquiteturas de software focadas em manutenibilidade (como Arquitetura Hexagonal ou Clean Architecture) e estratégias de refatoração. Com base nesse estudo, será elaborada e documentada a arquitetura inicial do novo sistema e da biblioteca pyreplicado, definindo os limites de contexto e a comunicação com o banco de dados PostgreSQL.

5.2. Análise de Código e Modelagem

Antes da implementação, será realizado o mapeamento das funcionalidades e endpoints da versão atual (monolito em Laravel). Em paralelo, ocorrerá a modelagem do novo banco de dados relacional, estruturando os domínios de cursos, disciplinas e trilhas curriculares para suportar a complexidade dos currículos da USP, mas também criar uma abstração que ultrapasse os limites de um único sistema acadêmico.

5.3. Implementação e Garantia de Qualidade

O desenvolvimento do sistema utilizará as linguagens Python e TypeScript. O backend será construído com o framework FastAPI, enquanto o frontend empregará a biblioteca React.

Com o intuito de assegurar a qualidade do software e mitigar a inserção de falhas, o processo de desenvolvimento adotará as seguintes práticas:

Análise Estática Utilização de ferramentas para validação de tipagem estática e padronização de formatação do código (como linters e formatadores), garantindo a conformidade com as diretrizes de estilo adotadas.

Testes Automatizados Ampla adoção de testes de unidade e de integração, com a meta de alcançar uma cobertura de código de pelo menos 80%.

Integração Contínua (CI) Configuração de pipelines automatizados por meio do GitHub Actions. Essas esteiras executarão as suítes de testes e as checagens estáticas a cada nova submissão de código, validando as alterações antes de serem incorporadas à versão principal.

5.4. Fomento à Comunidade e Validação

Para viabilizar a entrada de novos desenvolvedores, será produzida documentação técnica detalhada, incluindo guias de contribuição, configuração de ambiente de desenvolvimento contêinerizado (Docker) e documentação interativa da API via Swagger/OpenAPI. Como método de validação prática da barreira de entrada da nova arquitetura, planeja-se a organização de atividades de incentivo a contribuição direcionadas aos alunos do IME-USP. Tal iniciativa permitirá avaliar qualitativamente o processo de integração ao projeto, por meio da coleta de feedback dos participantes acerca da clareza do código-fonte e da documentação disponibilizada.

6. Cronograma

Atividade	Mai.	Jun.	Jul.	Ago.	Set.	Out.	Nov.
Fundamentação Teórica e Definição Arquitetural	•	•					
Engenharia Reversa e Modelagem	•	•					
Desenvolvimento da biblioteca pyreplicado	•	•					
Implementação do novo backend		•	•	•			
Implementação do novo frontend			•	•	•		
Atividades de fomento à comunidade					•	•	
Análise de Resultados						•	
Conclusão do TCC							•

Tabela 1: Cronograma previsto de atividades.

7. Referências

Bibliografia

[1] Instituto de Matemática e Estatística da Universidade de São Paulo, "Projeto Acadêmico 2023–2027", 2023. Acesso em: 1º de junho de 2026. [Online]. Disponível em: https://www.ime.usp.br/media/academica/projeto-academico/projeto_academico_2023-2027.pdf

[2] Instituto de Matemática e Estatística da Universidade de São Paulo, "Formatura — Graduação". Acesso em: 1º de junho de 2026. [Online]. Disponível em: https://www.ime.usp.br/graduacao/formatura/

[3] Pró-Reitoria de Graduação da Universidade de São Paulo, "Júpiter Web em vídeos: acompanhamentos, dados do programa, resumo, evolução no curso e rendimento acadêmico". Acesso em: 1º de junho de 2026. [Online]. Disponível em: https://prg.usp.br/alunos-2/jupiter-web-em-videos/acompanhamentos-dados-do-programa-resumo-evolucao-no-curso-e-rendimento-academico/

[4] BCC Dev — Instituto de Matemática e Estatística da Universidade de São Paulo, "Yggdrasil2". Acesso em: 1º de junho de 2026. [Online]. Disponível em: https://bccdev.ime.usp.br/principal/news/2018/09/28/yggdrasil2.html

[5] Z. Gao, C. Bird, e E. T. Barr, "To Type or Not to Type: Quantifying Detectable Bugs in JavaScript", em *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, p. 758–769.

[6] P. Petersen, S. Hanenberg, e R. Robbes, "An empirical comparison of static and dynamic type systems on API usage in the presence of an IDE: Java vs. groovy with eclipse", em *Proceedings of the 22nd International Conference on Program Comprehension*, Association

for Computing Machinery, 2014, p. 212–222. [Online]. Disponível em: https://doi.org/10.1145/2597008.2597152

[7] BCC Dev — Instituto de Matemática e Estatística da Universidade de São Paulo, "MatrUSP". Acesso em: 1º de junho de 2026. [Online]. Disponível em: https://bccdev.ime.usp.br/matrusp/