Análise de ferramentas low-code para o desenvolvimento de sistemas de recomendação: um estudo de caso para o projeto GrouPlanner

[1] **Danilo Santos Monteiro, Alberty Macedo**

[2] **Eveline de Jesus Viana Sá**

[3] **Josenildo Costa da Silva**

Instituto Federal de Educação, Ciência e Tecnologia do Maranhão – Campus Monte Castelo

Resumo

Sistemas de recomendação com Inteligência Artificial (IA) são promissores para a personalização de experiências, como rotas turísticas. No entanto, o desenvolvimento desses sistemas geralmente requer níveis avançados de habilidade de programação. O projeto de pós-doutorado, intitulado "Uma análise do uso de IA na geração de rotas turísticas personalizadas para o público idoso: um estudo de caso no Projeto The Route", teve como objetivo compreender os algoritmos inteligentes utilizados no projeto GrouPlanner e adaptá-los para o público idoso da cidade de São Luís - MA. Com base nos resultados parciais, este projeto propõe investigar o desenvolvimento deste sistema utilizando tecnologias de baixo código (*low-code*), comparando-as com o desenvolvimento tradicional (*high-code*) para avaliar sua viabilidade e eficiência.

1. Introdução

Os sistemas de recomendação representam uma área de pesquisa em constante crescimento, tendo o objetivo de sugerir soluções adequadas ao perfil do usuário. Tradicionalmente, esses sistemas são desenvolvidos utilizando abordagens de alto nível (*high-code*), que envolvem a escrita de código fonte detalhado em linguagens como Python, Java e

[1] Acadêmicos do Curso de Sistemas de Informação, Instituto Federal do Maranhão – Campus MTC. E-mail: danilom@acad.ifma.edu.br e alberty.macedo@acad.ifma.edu.br.

[2] Drª Professora Orientadora do Curso de Sistemas de Informação, Instituto Federal do Maranhão – Campus MTC. E-mail: eveline@ifma.edu.br

[3] Dr. Professor Orientador do Curso de Sistemas de Informação, Instituto Federal do Maranhão – Campus MTC. E-mail: jcsilva@ifma.edu.br

C++. No entanto, essa abordagem pode ser lenta e exige dos pesquisadores um conhecimento avançado de desenvolvimento de software.

Diante desse cenário, surge uma questão de pesquisa relevante: como acelerar o desenvolvimento de sistemas de recomendação? Uma abordagem promissora é o uso de tecnologias low-code, que permitem a criação de aplicações através de interfaces gráficas, reduzindo a necessidade de programação tradicional. Segundo a TOTVS (2023), a abordagem low-code acelera o desenvolvimento de software, diminuindo a dependência de desenvolvedores especializados e democratiza o processo de criação de sistemas permitindo que os próprios pesquisadores se tornem, também, os desenvolvedores.

Este projeto propõe investigar como as tecnologias low-code podem ser utilizadas para o desenvolvimento de sistemas de recomendação de rotas turísticas. Baseando-se em resultados parciais de uma pesquisa anterior sobre o uso de IA na geração de rotas turísticas personalizadas, o objetivo é explorar ferramentas *low-code*, avaliando seu alcance e potencial para acelerar o desenvolvimento e permitir uma maior participação de diferentes perfis de desenvolvedores. O desenvolvimento do projeto será focado em ferramentas *open source*, mas, caso não estejam disponíveis, serão utilizadas aquelas que oferecem planos iniciais gratuitos.

2. Metodologia

A metodologia para a realização deste projeto está estruturada em três fases principais. O processo visa, primeiramente, estabelecer uma base teórica sólida em Sistemas de Recomendação (SR), utilizando como referência o projeto Grouplanner. Em seguida, foca-se na seleção da ferramenta de desenvolvimento mais adequada, especificamente uma plataforma Low-Code. Por fim, a metodologia culmina no desenvolvimento de um Produto Mínimo Viável (MVP) do sistema de recomendação turística proposto.
Abaixo, detalhamos cada etapa do processo:

Fase 1: Estudo sobre Sistemas de Recomendação (SR) e o Grouplanner

Esta fase inicial dedica-se ao estudo aprofundado da temática de Sistemas de Recomendação. O projeto GrouPlanner é utilizado como principal base de referência, dada a sua consolidada base de dados sobre turistas e a aplicação prática de SR.

Os objetivos desta análise são:

- Compreender o processo de construção de um SR.
- Adquirir conhecimento sobre a aplicação de SR no contexto turístico.
- Obter insights para a modelagem do MVP, incluindo a definição de informações cruciais, o tratamento e a apresentação dos dados, e os resultados esperados.

Fase 2: Seleção de Plataformas de *Low-Code*

Devido à intenção de utilizar um ambiente de desenvolvimento ágil, esta fase consiste na seleção da plataforma low-code mais apropriada para hospedar o sistema de recomendação.

O processo de filtragem das plataformas disponíveis no mercado considerou critérios como:

- Capacidade de Desenvolvimento: O nível de poder e flexibilidade da plataforma para comportar um SR.
- Acessibilidade/Custo: A disponibilidade de uma versão gratuita (open-source ou de acesso livre) para o desenvolvimento do projeto.

Com base nesses critérios, as plataformas Power Apps e OutSystems foram selecionadas como as mais adequadas.

Fase 3: MVP

Após a definição da base teórica e a seleção da plataforma, esta fase envolve o desenvolvimento prático do Sistema de Recomendação Turística através da ferramenta no Power Apps. O MVP será construído contendo as operações mínimas essenciais para validação do conceito:

- Cadastro de Usuários.
- Geração de Recomendações: A funcionalidade será implementada através da associação de categorias de pontos turísticos, com base nos dados e mapeamentos estabelecidos durante o estudo do projeto GrouPlanner.

3. Resultados e Discussão

Após a construção do MVP, esta seção apresenta os resultados alcançados em conformidade com a metodologia descrita anteriormente. Também são feitas comparações

com o método de desenvolvimento tradicional (*high-code*), destacando a eficácia e as vantagens do desenvolvimento *low-code* para cenários que envolvem complexidade, como um Sistema de Recomendação.

3.1. Desenvolvimento

O desenvolvimento do MVP focou-se primariamente na implementação do método de geração de recomendações, que representou a etapa de maior complexidade e consumo de tempo. Esta fase exigiu a adaptação do modelo proposto no projeto GrouPlanner por (Alves, P. R. J, 2024).

É importante notar que, em ambientes *low-code*, a maior parte do esforço se concentra no *Backend* (lógica da aplicação). Embora algumas plataformas, como o *Power Apps*, possuam sua própria linguagem de programação, a integração com linguagens tradicionais (como Python e Java) geralmente requer o uso de recursos e serviços externos.

A primeira etapa do desenvolvimento consistiu na modelagem do sistema, conforme ilustrado na Figura 3.1. Esta modelagem foi fundamental para a definição das entidades essenciais e suas relações, incluindo: Usuários; Pontos Turísticos; Informações de Avaliação; Categorias dos Pontos.

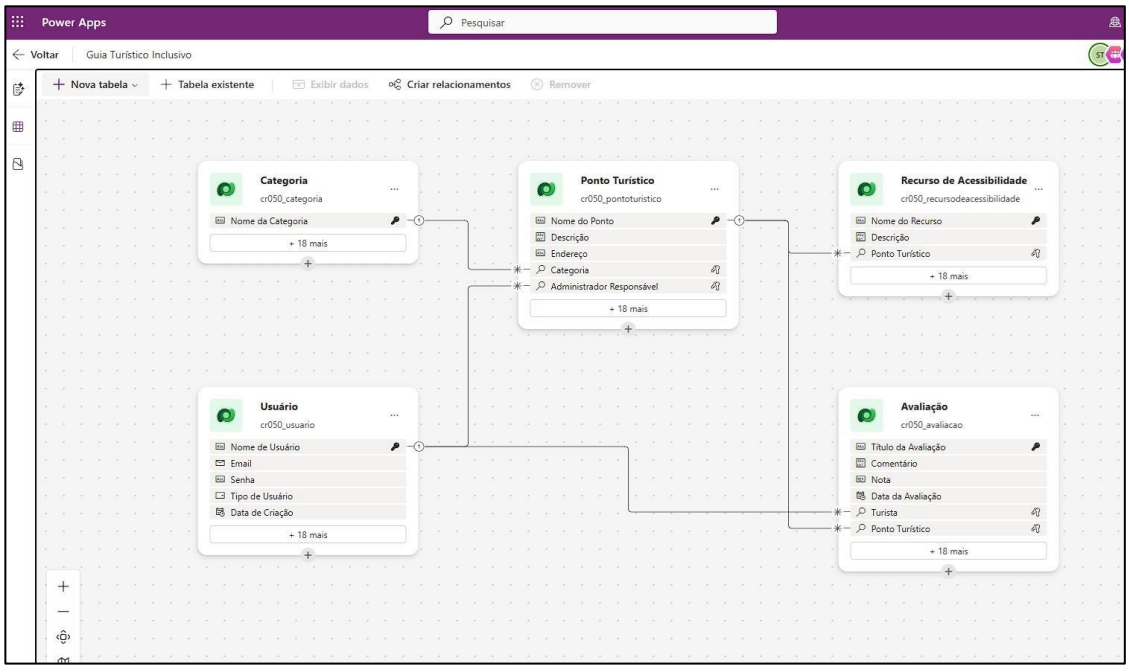

Figura 3.1: Modelagem do sistema no ambiente do Power Apps

Embora a lógica (backend) seja o coração do sistema, o desenvolvimento *low-code* proporcionou um aumento significativo na velocidade de construção da interface visual (*frontend*). Os ambientes low-code se beneficiam da técnica *Drag and Drop* e de componentes visuais pré-prontos. Isso permite que o desenvolvedor selecione e organize rapidamente os elementos necessários para a visualização, resultando em uma produtividade muito maior em comparação com o desenvolvimento tradicional, onde a codificação da interface pode consumir um tempo considerável. Com isso, a construção das telas apresentadas na Figura 3.2 levou um tempo muito menor do que poderiam levar em outros casos.

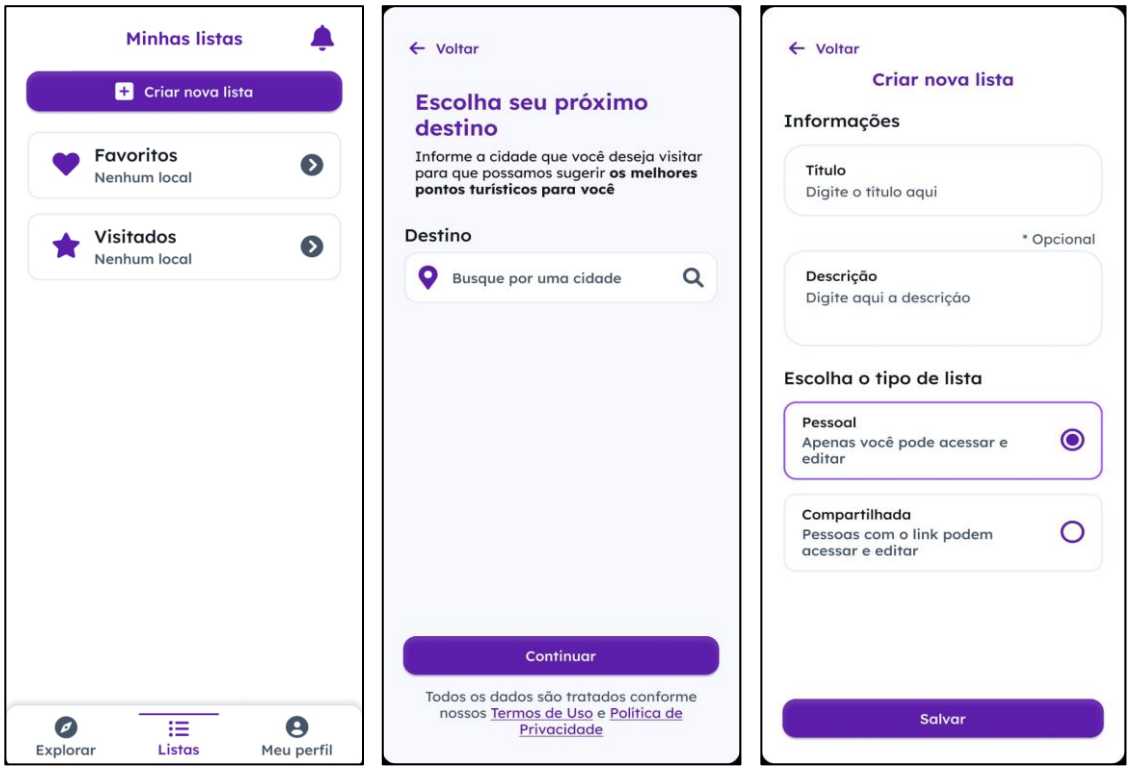

Figura 3.2: Telas do MVP

3.2. Low-Code x High-Code

A escolha da metodologia de desenvolvimento é crucial para o sucesso e a longevidade de um projeto de software, especialmente em cenários que exigem agilidade, como o desenvolvimento de um produto ou serviço. Na Tabela 1 apresentamos uma comparação estruturada entre o Desenvolvimento Low-Code e o Desenvolvimento Tradicional (*High-Code*), com base em critérios de impacto técnico e estratégico.

Tabela 1: Quadro comparativo das características entre Low-Code e High-Code

Característica	Low-Code	High-Code
Produtividade e Velocidade	Acelera o tempo de colocação no mercado (Time-to-Market) por meio de componentes visuais e drag-and-drop (LINKIT, s.d.).	Requer codificação manual, resultando em ciclos de desenvolvimento mais longos (Trigo et al., 2022).
Nível de Abstração	Foca na lógica de negócios, abstraindo a complexidade da infraestrutura e do código-fonte (Evidensys, 2022).	Exige controle total e profundo conhecimento das linguagens de programação e frameworks.
Customização e Flexibilidade	A customização é restrita aos recursos e bibliotecas da plataforma (InnSpire.dev, s.d.).	Permite a criação de funcionalidades únicas e alto grau de personalização (Mind Consulting, s.d.).
Custo de Propriedade (TCO)	Envolve custos de licenciamento da plataforma, que podem se elevar com o aumento de usuários e funcionalidades (IT Forum, 2020).	Os custos iniciais são focados na mão de obra especializada, com menor dependência de licenças de terceiros.
Aderência a Padrões de Engenharia	Muitas plataformas não oferecem suporte adequado para práticas como controle de versão, testes automatizados e Clean Code (ResearchGate, 2021).	Permite a aplicação irrestrita das melhores práticas de Engenharia de Software para alta manutenibilidade.
Risco de Vendor Lock-in	Alta dependência do fornecedor da plataforma, o que dificulta a migração para outras tecnologias (IT Forum, 2020).	O código é totalmente proprietário do cliente, facilitando a portabilidade e a longevidade do projeto.

3.3. Limitações

Embora o *Low-Code* tenha proporcionado uma rapidez para o desenvolvimento do MVP, sua aplicação em um Sistema de Recomendação (SR) complexo como esse mostrou algumas limitações significativas, pois o objetivo era explorar se as atuais plataformas teriam capacidades e ferramentas para desenvolver sistemas de alta complexidade.

As principais limitações observadas foram:

- **Restrição Algorítmica e de Otimização:** Plataformas *Low-Code* limitam a customização fina dos algoritmos essenciais para a eficácia de um SR (como filtragem colaborativa). A implementação de bibliotecas externas otimizadas (e.g., Python) requer o uso de conectores ou serviços externos, restringindo a liberdade de design e a otimização de performance.
- **Risco de Vendor Lock-in:** A dependência de uma plataforma Low-Code cria um forte aprisionamento tecnológico (vendor *lock-in*). Isso torna a futura migração para

outras soluções ou para o desenvolvimento *High-Code* significativamente mais complexa e custosa (IT FORUM, 2020).

- **Desafios de Escalabilidade e Performance:** Embora o Low-Code suporte o MVP, a escalabilidade de alto nível necessária para lidar com um volume crescente de dados turísticos e de usuários é dificultada. A otimização de performance em larga escala, crucial para um SR robusto, é mais difícil de controlar e refinar, pois está limitada às configurações de runtime da plataforma (TRIGO et al., 2022).

4. Conclusão

O desenvolvimento deste projeto confirmou que a abordagem Low-Code é uma estratégia eficaz e ágil para sistemas de pequeno e médio porte com complexidade moderada. No contexto de um Sistema de Recomendação (SR), que exige recursos algorítmicos avançados, o Low-Code foi excelente para o rápido desenvolvimento de protótipos e MVPs. Contudo, ele impôs limitações significativas em relação à escalabilidade, otimização de performance e personalização algorítmica. Essas restrições se tornaram evidentes porque apenas as versões gratuitas das plataformas foram utilizadas.

Em última análise, as plataformas Low-Code, apesar de serem catalisadores de produtividade, ainda não suportam de forma ideal sistemas com a alta complexidade e os requisitos de escalabilidade de um SR em nível de produção. A solução completa e robusta para um SR deve residir em um modelo híbrido, que utiliza a agilidade do Low-Code para o frontend e a flexibilidade do *High-Code* para a lógica central e otimização. Isso sugere que as versões premium das plataformas *Low-Code* podem ser capazes de contornar algumas dessas limitações.

5. Agradecimentos

Ao IFMA pelo apoio e suporte ao desenvolvimento da pesquisa.

6. Referências

TOTVS. Low code e no-code: guia completo. Blog TOTVS, São Paulo, 19 out. 2022. Disponível em: https://www.totvs.com/blog/negocios/no-code-low-code/

TRIGO, Antonio; VARAJÃO, João; ALMEIDA, Miguel. Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game? IEEE Software, 2022. Disponível em: https://www.researchgate.net/publication/365873896_Low-Code_Versus_Code-Based_Software_Development_Which_Wins_the_Productivity_Game.

RESEARCHGATE. LETHBRIDGE, Timothy. Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. ResearchGate, 2021. Disponível em: https://www.researchgate.net/publication/355182082_Low-Code_Is_Often_High-Code_So_We_Must_Design_Low-Code_Platforms_to_Enable_Proper_Software_Engineering.

IT FORUM. Low-code: as vantagens e desvantagens do tal código baixo. IT Forum, [S. l.], 10 jan. 2020. Disponível em: https://itforum.com.br/low-code-as-vantagens-e-desvantagens-do-tal-codigo-baixo/. Acesso em: 29 set. 2025.

INN SPIRE. High-Code vs Low-Code: qual escolher para o seu projeto de software? Inn Spire.dev, [S. l.], [2025]. Disponível em: https://innspire.dev/blog/high-code-vs-low-code-qual-escolher/. Acesso em: 29 set. 2025.

MIND CONSULTING. Low Code vs High Code: Por que o Desenvolvimento do zero é a Melhor Escolha. Mind Consulting, [S. l.], 27 fev. 2025. Disponível em: https://mindconsulting.com.br/2025/02/low-code-vs-high-code-por-que-escolher-desenvolvimento-tradicional/

APPIAN. Low-Code vs. No-Code vs. High-Code: Which Will Serve You Best? Appian Blog, [S. l.], 22 fev. 2023. Disponível em: https://appian.com/blog/acp/low-code/low-code-vs-no-code-vs-high-code

NINJAONE. High Code vs. Low Code vs. No Code: Navigating the Best Coding Solutions for Your Needs. NinjaOne Blog, [S. l.], [2025]. Disponível em: https://www.ninjaone.com/blog/high-code-vs-low-code-vs-no-code