API para Testes Adaptativos Computadorizados: Inovação em Avaliação Educacional com Base na TRI e MDC

Bruno Ap. Silvestre
ICMC - Universidade de São Paulo (USP)
São Carlos, São Paulo, Brasil
bruno.silvestre@usp.br

Thiago F. Miranda
ICMC - Universidade de São Paulo (USP)
São Carlos, São Paulo, Brasil
thiagofm@usp.br

Mariana Cúri
ICMC - Universidade de São Paulo (USP)
São Carlos, São Paulo, Brasil
mcuri@icmc.usp.br

Resumo

Este trabalho apresenta o desenvolvimento de uma API para a aplicação de Testes Adaptativos Computadorizados (CAT), visando suprir a escassez de soluções acessíveis e flexíveis nessa área. A API integra modelos psicométricos da Teoria de Resposta ao Item (TRI) e dos Modelos de Diagnóstico Cognitivos (MDC), permitindo a seleção dinâmica de itens e a estimação de proficiência. A arquitetura combina uma aplicação em Django à microsserviços em R com o pacote mirtCAT, garantindo modularidade, escalabilidade e integração com sistemas externos. Testes práticos em ambientes reais validaram a funcionalidade, estabilidade e precisão da solução, tanto em provas adaptativas quanto tradicionais. Os resultados demonstram o potencial da ferramenta para aplicações educacionais, com registro detalhado de dados para análises futuras.

Palavras-chave: Testes Adaptativos Computadorizados; API; Teoria da Resposta ao Item; Modelos de Diagnóstico Cognitivo.

1 Introdução

Os Testes Adaptativos Computadorizados (CAT, do inglês *Computerized Adaptive Testing*) têm se consolidado como uma abordagem inovadora para a avaliação de habilidades e conhecimentos em diversas áreas, como educação, psicologia e certificações profissionais. Diferentemente dos testes tradicionais, em que todos os participantes respondem ao mesmo conjunto de itens, os testes adaptativos selecionam questões de forma dinâmica com base no desempenho individual do examinado, proporcionando avaliações mais precisas e eficientes com menos itens. Essa metodologia, amplamente baseada na Teoria de Resposta ao Item (TRI) e nos Modelos de Diagnóstico Cognitivos (MDC), exige infraestrutura tecnológica capaz de gerenciar a seleção adaptativa dos itens e o cálculo contínuo da habilidade estimada.

Apesar do avanço teórico e das aplicações práticas de CAT, o desenvolvimento de soluções técnicas para sua implementação ainda apresenta desafios. A criação de sistemas adaptativos demanda conhecimento especializado em algoritmos de seleção de itens, métodos de estimação de habilidade e arquitetura de software capaz de garantir desempenho em tempo real. Atualmente, existem poucos recursos de código aberto ou APIs públicas que ofereçam suporte completo ao fluxo de um teste adaptativo, o que dificulta a adoção mais ampla dessa tecnologia em projetos de avaliação educacional e psicológica.

Com o objetivo de preencher essa lacuna, este trabalho apresenta o desenvolvimento de uma API (*Application Programming Interface*) dedicada à administração de Testes Adaptativos Computadorizados. A proposta é fornecer uma ferramenta flexível, extensível e de fácil integração, que permita a realização de testes adaptativos baseados em diferentes estratégias de seleção e estimação. Além da descrição da arquitetura da API, este estudo também relata um teste prático realizado para avaliar seu desempenho em cenários simulados, validando a funcionalidade e a eficiência da solução proposta.

2 Objetivo

Este trabalho tem como objetivo apresentar o desenvolvimento de uma API para a administração de Testes Adaptativos Computadorizados (CAT), já em fase avançada de implementação, e relatar os resultados obtidos em um teste prático realizado para avaliação de sua funcionalidade. A API está sendo projetada para oferecer uma solução flexível e extensível, capaz de integrar algoritmos de seleção de itens e estimação de habilidade com base na teoria de resposta ao item e modelos de diagnóstico cognitivos. Além de descrever a arquitetura e os principais recursos da API, este estudo busca analisar seu desempenho e a viabilidade de sua aplicação em ambientes de avaliação adaptativa.

3 Metodologia

Nesta seção, vamos abordar mais sobre CAT, os modelos que nossa aplicação implementa e a estrutura da solução proposta.

3.1 Teste Adaptativo Computadorizado (CAT)

O CAT foi implementado para selecionar itens dinamicamente com base nas estimativas de proficiência. O sistema segue os princípios descritos por Weiss e Sahin (2024), operacionalizado com o pacote mirtCAT (Chalmers, 2016) do software R (R Core Team, 2025).

Os principais componentes de um Teste Adaptativo Computadorizado incluem: o banco de itens, composto por questões previamente calibradas com base nos modelos TRI e MDC; o valor inicial de proficiência, que pode ser um escalar ou um vetor — contínuo nos modelos da TRI ou discreto nos modelos de diagnóstico cognitivo.; o critério de seleção de itens, que define a estratégia para escolha do próximo item, como o Critério da Máxima Informação (MI); a atualização da proficiência, realizada após cada resposta por métodos como a Média A Posteriori (EAP); e, por fim, o critério de parada, estabelecido por um número fixo de itens ou pela obtenção de um erro padrão mínimo na estimativa da proficiência.

Em termos práticos, um CAT começa com base em um valor inicial de proficiência fixado. A partir deste valor e utilizando um critério de seleção específico, um item é escolhido e apresentado ao respondente. Com base na resposta dada, a proficiência do respondente é estimada e então verificamos se o critério de parada do teste foi alcançado. Caso não tenha sido, ocorre uma nova iteração do CAT, onde outro item é selecionado e apresentado. Este processo continua até que o critério de parada seja satisfeito, momento em que o CAT é concluído. Ao final do teste, conhecemos a estimativa da proficiência do respondente juntamente com seu erro padrão.

3.2 Modelos Psicométricos

Segundo Chalmers (2016), a modelagem da proficiência no contexto das avaliações educacionais principalmente realizada com base em modelos da Teoria da Resposta ao Item, onde o principal objetivo é estimar valores contínuo de proficiências. Amplamente conhecidos, os modelos da TRI disponíveis para a criação de testes adaptativos presente neste projeto são os modelos logísticos unidimensionais e multidimensionais de 1, 2, 3 e 4 parâmetros.

Além dos modelos da TRI, este trabalho conta com a integração dos modelos de diagnóstico cognitivo. Nos últimos anos, os MDC têm ganhado destaque por possibilitarem a identificação precisa dos pontos fortes e fracos de um indivíduo em diversas áreas do conhecimento. Esses modelos estimam vetores latentes discretos, permitindo traçar um perfil cognitivo multidimensional detalhado, indicando quais habilidades o respondente domina ou não em uma avaliação. Embora compartilhem semelhanças com os modelos da teoria de resposta ao item, os MDC se distinguem por exigirem um componente previamente definido: a matriz-Q. Essa matriz codifica a estrutura de habilidades requeridas para cada item (GEORGE; ROBITZSCH, 2015; DEONOVIC et al., 2019).

Dentre os principais modelos de diagnóstico, destaca-se o modelo DINA. Conforme descrito por de la Torre (2009), o DINA — acrônimo de *Deterministic Inputs, Noisy "And" gate* — parte do pressuposto de que, para que um indivíduo responda corretamente a um item, ele deve dominar todas as habilidades exigidas por esse item.

Assim como nos modelos da TRI, a estimação das habilidades no modelo DINA pode ser realizada a partir de métodos bayesianos como o da Média da Posteriori (EAP) ou Moda da Posteriori (MAP).

3.3 Estrutura da Aplicação

A arquitetura da aplicação desenvolvida foi projetada para permitir a realização de testes adaptativos computadorizados de forma modular, escalável e de fácil integração com diferentes sistemas externos. A solução é composta por três principais componentes: um servidor principal construído com o framework Django (Gore, 2021; Duisebekova et al, 2021), um módulo estatístico externo implementado em R, e um banco de dados relacional MariaDB. Esses elementos interagem por meio de uma API RESTful que expõe os recursos necessários para a administração e aplicação dos testes.

No contexto da engenharia de software, uma API (Interface de Programação de Aplicações) é um mecanismo que permite que diferentes partes de um software, ou sistemas distintos, troquem informações e funcionalidades de forma padronizada. Ela atua como uma ponte entre aplicações, ocultando a complexidade interna e facilitando a reutilização de recursos. No caso das APIs REST (Representational State Transfer), a comunicação ocorre por meio do protocolo HTTP, utilizando formatos como JSON ou XML para acessar, modificar ou excluir dados. Esse modelo é amplamente adotado por sua simplicidade, escalabilidade e compatibilidade com arquiteturas distribuídas (Fielding, 2000; Richardson & Ruby, 2007).

Na aplicação desenvolvida, o servidor Django é implementado como uma API REST, sendo responsável pela gestão central da aplicação, incluindo o controle de usuários, o cadastro de questões, a criação de modelos de prova e a orquestração do fluxo de aplicação dos testes. A interface administrativa do Django é voltada para administradores — frequentemente profissionais da educação — que configuram as avaliações e cadastram os itens, permitindo o uso da ferramenta mesmo por usuários sem conhecimentos técnicos. Todas as informações são armazenadas no banco de dados MariaDB, que centraliza os dados relacionados às provas, usuários e respostas.

A componente adaptativa é delegada a instâncias de R, organizadas como microsserviços com o auxílio do pacote plumber (SCHLOERKE; ALLEN, 2022). Esses serviços em R utilizam o pacote mirtCAT para realizar os cálculos baseados na TRI, estimar a habilidade dos participantes e selecionar as próximas questões a serem apresentadas durante o teste. A API Django se comunica com essas instâncias por meio de requisições HTTP, enviando dados sobre os itens disponíveis e as respostas fornecidas pelo usuário, e recebendo em retorno as decisões adaptativas. Por fim, a estrutura permite que consumidores externos — como interfaces web ou outras APIs — utilizem os serviços da aplicação de forma transparente, possibilitando sua adoção em diferentes contextos educacionais. Atualmente, a equipe trabalha na adaptação desse serviço para suportar o modelo cognitivo DINA, utilizando a mesma lógica estrutural da mirtCAT, mas com o desenvolvimento de um pacote próprio.

4 Resultados

Como resultado do desenvolvimento, atualmente contamos com uma aplicação funcional para a realização de testes adaptativos computadorizados, composta por uma API REST desenvolvida em Django acoplada a um módulo estatístico externo em R com uso do pacote mirtCAT. A aplicação já permite o cadastro e gerenciamento de questões e provas por meio de uma interface administrativa acessível, bem como a aplicação automática de testes com base na teoria de resposta ao item.

Foram realizados dois testes práticos em turmas reais com sucesso: um utilizando o modelo adaptativo proposto e outro no formato clássico. Essa escolha teve como objetivo demonstrar a flexibilidade da aplicação, que pode ser utilizada tanto para avaliações adaptativas quanto para provas tradicionais. Em ambos os casos, o sistema conduziu corretamente o fluxo de aplicação, garantindo o envio das questões apropriadas e, no caso do teste adaptativo, encerrando automaticamente a prova com base nos critérios de parada definidos. A aplicação demonstrou estabilidade, coerência na seleção dos itens e boa comunicação entre os módulos implementados em Python e R.

Além da aplicação da prova, a API também registra e disponibiliza uma série de dados importantes para análise, como o histórico completo de respostas dos participantes, a evolução da proficiência estimada ao longo da prova, o tempo total da aplicação e o tempo gasto por questão. Essas informações oferecem um panorama detalhado do desempenho de cada participante e serão, futuramente, utilizadas na geração de relatórios visuais, visando fornecer feedbacks claros e acessíveis para alunos e educadores.

5 Conclusões e Considerações Finais

O desenvolvimento da API para aplicação de testes adaptativos computadorizados representa um avanço na direção de avaliações mais personalizadas, eficientes e alinhadas às necessidades individuais dos alunos. A solução proposta mostrou-se funcional e versátil, permitindo tanto a aplicação de provas adaptativas baseadas na teoria de resposta ao item quanto de provas tradicionais, com um fluxo automatizado, estável e de fácil configuração.

Os testes práticos realizados demonstraram que a arquitetura adotada — combinando Django, R e MariaDB — é adequada para aplicações educacionais, oferecendo flexibilidade e confiabilidade. A separação entre os módulos da aplicação facilita a manutenção e futuras expansões, como a integração com novas interfaces ou a adição de funcionalidades voltadas à visualização e análise dos resultados.

Como trabalhos futuros, pretende-se implementar dashboards interativos para a visualização dos dados gerados ao longo das provas, ampliar a aplicação da API com a inclusão de outros modelos de diagnóstico cognitivo, como os modelos DINO e GDINA, e realizar novas aplicações da ferramenta com turmas reais em contextos educacionais. Esses passos visam tanto o aprimoramento técnico da solução quanto sua validação prática no ambiente escolar. Atualmente, encontra-se em desenvolvimento a adaptação da API para contemplar o modelo cognitivo DINA, utilizando uma estrutura semelhante à do pacote mirtCAT, porém com a construção

de um pacote próprio, visando maior flexibilidade e adequação aos requisitos específicos do modelo.

6 Referências

CHALMERS, R. P. Generating Adaptive and Non-Adaptive Test Interfaces for Multidimensional Item Response Theory Applications. *Journal of Statistical Software*, v. 71, n. 5, 2016.

de la TORRE, J. DINA Model and Parameter Estimation: A Didactic. *Journal of Educational and Behavioral Statistics*, v. 34, n. 1, p. 115–130, mar. 2009.

DEONOVIC, B.; CHOPADE, P.; YUDELSON, M.; TORRE, J. de la; DAVIER, A. A. von. Application of cognitive diagnostic models to learning and assessment systems. In: . *Handbook of Diagnostic Classification Models: Models and Model Extensions, Applications, Software Packages*. Cham: Springer International Publishing, 2019. p. 437–460. ISBN 978-3-030-05584-4.

DUISEBEKOVA, Kulanda; KHABIROV, Roman; ZHOLZHAN, Azamat. Django as Secure Web-Framework in Practice. Вестник Казахской академии транспорта и коммуникаций им. М. Тынышпаева, n. 1, p. 275-281, 2021.

FIELDING, Roy Thomas. *Architectural styles and the design of network-based software architectures*. 2000. 1 v. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, 2000.

GEORGE, A. C.; ROBITZSCH, A. Cognitive diagnosis models in r: A didactic. *The Quantitative Methods for Psychology*, TQMP, v. 11, n. 3, p. 189–205, 2015. Disponível em: <http://www.tqmp.org/RegularArticles/vol11-3/p189/p189.pdf>.

GORE, Himanshu et al. Django: Web development simple & fast. *Annals of the Romanian Society for Cell Biology*, v. 25, n. 6, p. 4576-4585, 2021.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2024. Disponível em: <https://www.R-project.org/>.

RICHARDSON, Leonard; RUBY, Sam. *RESTful Web Services*. Sebastopol: O'Reilly Media, 2007.

SCHLOERKE, B.; ALLEN, J. *plumber: An API Generator for R*. [S.l.], 2022. R package version 1.2.1. Disponível em: <https://CRAN.R-project.org/package=plumber>.

WEISS, D. J.; SAHIN, A. *Computerized Adaptive Testing: From Concept to Implementation*. [S.l.]: Guilford Publications, 2024.