Cardery, aplicação mobile para estudo da língua inglesa: relatório de desenvolvimento do back-end.

Fábio Rodrigo Lima Martins

Orientador: Ivo Félix Gualberto de Sá

Resumo

Este relatório tem como objetivo mostrar o pensamento utilizado para desenvolver o back-end de uma aplicação mobile para estudo da língua inglesa através de flashcards. Para tanto, explica, primeiramente, por que aprender inglês é relevante para comunicação e conhecimento científico. Discute, ainda, por que dispositivos móveis estão sendo usados como uma ferramenta de aprendizado e como esses dois tópicos foram fundidos para dar início ao desenvolvimento do back-end do aplicativo. Em seguida, define-se o que é e do que se compõe o back-end e são descritas as tecnologias utilizadas no sistema da aplicação desenvolvida, algumas delas sendo: NestJS, TypeScript e PostgreSQL como um banco de dados, logo após um breve texto descrevendo detalhes relacionados a estas tecnologias. Por fim, apresentam-se as funcionalidades do sistema e alguns detalhes relevantes para a sua implementação, indicando, por exemplo, que a documentação que explica como se comunicar com o servidor foi gerada automaticamente via código.

Palavras-chave: Back-end. Desenvolvimento de software. Servidor. Aplicação mobile. Aprendizagem de língua inglesa.

Abstract

This report aims to show the thinking process to develop the back-end of a mobile application whose goal is to help its user to study the English language through flashcards. To do so, this work explains, firstly, why learning English is relevant for communication and scientific knowledge. It also discusses why mobile devices are being used as a learning tool and how these two topics were merged to start the development of the back-end of the application. Then, what the back-end is and what it is made of is defined and the technologies used in the developed application system are described. Some of them are: NestJS, TypeScript and PostgreSQL as a database. Finally, the system functionalities and some relevant details for its

implementation are presented, indicating, for example, that the documentation that explains how to communicate with the server was automatically generated via code.

Keywords: Back-end. Software development. Server. Mobile application. English language learning.

1. Introdução

Falando sobre literatura, Wallis (1969) pontua que “todo tipo de literatura está amplamente disponível em edições inglesas, e, sem ser arrogante, pode ser dito que existe de forma escassa qualquer pedaço de conhecimento que não foi registrado hoje, adequadamente na língua inglesa.”[1]

Inglês é, hoje, a língua mais falada do mundo, usada internacionalmente para compartilhar conhecimento e fechar negócios. Aprender inglês torna mais fácil atingir uma população maior para o compartilhamento de pesquisas, trabalhar fora e aprender os últimos avanços científicos em múltiplos tópicos.

Muitas são as maneiras utilizadas para se estudar a língua inglesa, podendo-se destacar dentre elas o uso de ferramentas acessíveis, como telefones, tablets e tecnologias mobile em geral, que estão sendo utilizadas ao redor do mundo para aprender idiomas.

Sobre usar essas ferramentas para aprendizados, Kukulsa-Hulme e Traxler (2005, p. 10) dizem que “utilizar essas ferramentas móveis traz uma série de vantagens que contribuem para a sua definição, podendo ser espontâneo, pessoal, informal, contextual, portátil e disponível em qualquer lugar.”[2] Integrados em nossas vidas, estes dispositivos estão disponíveis em todo lugar ao redor do globo.

Usando dispositivos móveis, é fácil decidir quando aprender; e aplicativos feitos para isso conseguem personalizar a experiência com base na interação do usuário com o sistema. Dúvidas estão no máximo a três ou quatro cliques de distância de uma pesquisa na internet. Comunicar-se e expressar seus pensamentos são formas vitais para adquirir novas conexões e conhecimento.

---

[1] Tradução nossa.
[2] Tradução nossa.

Com a relevância da língua inglesa cada vez mais em evidência e um país que possui mais de um dispositivo móvel por habitante no brasil (FGV, 2021), é realmente importante que aplicativos educativos e gratuitos estejam disponíveis neste tipo de ambiente fértil.

É nesse contexto que este trabalho surge com o objetivo geral de desenvolver o back-end de uma aplicação mobile de flashcards para auxiliar no aprendizado da língua inglesa. Seus objetivos específicos são:

- Definir as tecnologias a serem utilizadas quanto ao servidor e ao banco que persistirá os dados;
- Definir os requisitos e funcionalidades do software;
- Criar os módulos responsáveis pela regra de negócio do software.

Este relatório está dividido da seguinte forma: inicialmente, serão apresentados todos os elementos relacionados ao desenvolvimento do back-end do Cardery, tratando do conceito de back-end adotado pelo trabalho, passando pelas tecnologias utilizadas durante o desenvolvimento do sistema e finalizando com as funcionalidades disponíveis na aplicação. Por fim, apresentam-se as considerações gerais, em que se faz uma breve avaliação do percurso, além de tratar de perspectivas futuras quanto ao aplicativo desenvolvido.

2 O back-end do Cardery.

Nesta seção, será apresentada a definição do que é back-end, como ele se relaciona com a web e sua composição e responsabilidades.

2.1 Conceito de back-end

Sobre aplicações web distribuídas, Abdullah e Zeki (2014, p. 1) pontuam que “serviços web estão se tornando tecnologias chaves para a implementação de aplicações integradas e distribuídas.” O quão rápido e responsivo é uma aplicação é determinante para o seu sucesso, tendo um impacto crítico na aceitação daqueles que a utilizam. Aplicações, em um contexto geral, necessitam de um serviço back-end que lidará com a execução da parte mais pesada e custosa do código. O

back-end de uma aplicação é composto por geralmente três partes: um servidor, a lógica da aplicação e um banco de dados. Além disso, geralmente tem sua estrutura hospedada na nuvem (ABDULLAH & ZEKI, 2014, p. 1).

Servidores são uma das bases da web nos dias de hoje; Oluwatosin (2014, p. 67) diz que, “no mundo de hoje, o sistema cliente-servidor se tornou tão popular porque está sendo utilizado virtualmente todos os dias para diferentes aplicações”. Alguns protocolos que se encaixam nesse padrão cliente-servidor são por exemplo: Protocolo de Transferência de Arquivos (FTP), Protocolo de Transferência de Correio Simples (SMTP) e o Protocolo de Transferência de Hipertexto (HTTP). Um sistema cliente-servidor pode ser definido como uma arquitetura de software que é composta por ambos, clientes enviam requisições e os servidores que estão escutando-as respondem ao que foi requisitado.” (OLUWATOSIN, 2014).

Sobre a lógica da aplicação, também conhecida como lógica de negócio, Evans e Evans (2004, p. 2) afirmam que “todo programa de software se relaciona a alguma atividade ou interesse de seus usuários. O assunto no qual os usuários aplicam o programa é a lógica de negócio da aplicação.” Esse assunto é chamado de domínio. Alguns domínios envolvem o mundo físico, como, por exemplo, um programa de reservas de passagens de uma companhia aérea que envolve pessoas de verdade entrando em aviões de verdade. Já alguns desses assuntos são intangíveis, como, por exemplo, o domínio de uma aplicação de contabilidade envolve diretamente dinheiro e finanças. Na maior parte dos casos, domínios de aplicações têm pouco a ver com as aplicações em si. (EVANS & EVANS, 2004)

Millett e Tune (2015, p. 7), por sua vez, detalham ainda mais sobre o desenvolvimento de software a partir da lógica de domínio, definindo o que é o domínio núcleo, aos pontuarem que “desenvolvedores e experts na lógica de domínio utilizam padrões para analisar e destilar um domínio grande em subgrupos de domínios. A partir do que foi filtrado, o domínio núcleo de uma aplicação é a razão principal pela qual o software está sendo desenvolvido”[3]. A lógica da aplicação refere-se, então, ao escopo do código responsável por atender diretamente às necessidades dos clientes.

Quanto ao banco de dados, Harrington (2016, p. 5) traz o seguinte: “a ideia de um banco de dados é que um usuário ou uma aplicação não precisem se preocupar sobre a forma como os dados são guardados fisicamente no disco: existe um

[3] Tradução nossa.

conceito principal por trás de todos os bancos de dados". Há casos em um ambiente de negócio onde precisamos guardar dados e realizar operações em cima deles; e esses dados se relacionam com outros dados de formas variadas. Para ser considerado um banco de dados, o lugar em que estes dados estão guardados devem não só conter os dados em si, mas também informações sobre os seus relacionamentos. Podemos, por exemplo, precisar relacionar os pedidos da nossa loja a clientes específicos e os nossos itens no estoque a essas vendas. Um software conhecido como Sistema gerenciador de banco de dados (SGBD), então, traduz a requisição do usuário para um endereço físico dos dados no disco. (HARRINGTON, 2016)

O back-end, então, refere-se, no geral, à parte do software na qual o usuário final interage de forma indireta. O back-end é responsável por criar as lógicas do negócio e fornecê-las via uma interface padrão, um modo de acesso a essas lógicas, que possuem a capacidade de persistir, alterar e remover dados de usuários de um determinado sistema. Lidar com performance e arquitetura geral da aplicação é também responsabilidade do back-end, assim como verificar e autorizar o acesso de usuários a essas interfaces. O back-end será criado a partir de bibliotecas de código aberto e será multiplataformas.

2.2 Tecnologias do sistema

NestJS é um framework baseado em Node.js, bem documentado, voltado para o desenvolvimento de APIs server-side. Este framework tem, como conceitos importantes, a relação com a organização arquitetural de projetos e a injeção de dependência.

Sobre a documentação de software em um contexto de engenharia de software, artigo da AltexSoft (2018) traz que "a documentação é um termo guarda-chuva que cobre todos os documentos escritos e materiais que lidam com o crescimento e uso de um produto de software". O objetivo principal de uma documentação é clarificar os módulos de uma aplicação, unificar dados e conhecimentos relacionados ao projeto, a fim de permitir discussões de questões importantes que possam ser levantadas por desenvolvedores e stakeholders. (ALTEXSOFT, 2018.) A documentação do NestJs é extensa e com vários exemplos, desde a parte de iniciar uma aplicação até a parte de aprimorá-la e de estendê-la.

Artigo da RedHat (2020) apresenta arquitetura da seguinte forma:

> A arquitetura de uma aplicação descreve os padrões e técnicas utilizadas para desenhar e construir uma aplicação. A arquitetura te dá um roadmap e melhores práticas a se seguir quando construindo uma aplicação, de forma que você acabe com um aplicativo bem estruturado[4].

Sobre a arquitetura pré-definida do NestJS, Rahman (2019) pontua que “esse framework guia fortemente os desenvolvedores para usar certas ferramentas e codificar de certa forma”[5]. Os padrões definidos pelo framework facilitam a modificação e expansão do código, além de facilitar a entrada de novos desenvolvedores no projeto que já estão acostumados com esse padrão.

A estrutura organizacional do projeto é composta a partir destes tipos principais:

- Controllers - unidades responsáveis por lidar com as requisições e respostas para os clientes;
- Providers - unidades que podem ser injetadas como dependência em outras classes;
- Repositories - unidades que possuem lógica para entrar salvar/consultar dados dentro do banco de dados;
- Guards - unidades responsáveis por garantir/negar a autorização do usuário para o recurso que ele está tentando acessar. Ex: Usuário tentando pedir flashcards privados de outro usuário.
- Modules - agrupadores de funções relacionadas a um mesmo propósito. Ex: Módulo de autenticação.

O Express é um framework minimalista utilizado pelo NestJS para criar rotas, que serão chamadas pelo front-end para executar regras de negócio.

O código do projeto é escrito em Typescript. Desenvolvido e mantido pela Microsoft, o Typescript é definido por Pham (2020, p. 19) como a “versão tipada do Javascript que ajuda desenvolvedores nos tempos de compilação e execução. Essa

[4] Tradução nossa.
[5] Tradução nossa.

linguagem empodera desenvolvedores com a simplicidade do Javascript e a robustez e segurança de uma linguagem tipada."[6] À medida que uma aplicação começa a escalar um código bem definido e seguro, torna-se muito importante para a manutenibilidade do projeto.

A documentação do NestJS (2023) define o que é considerado a raiz de um aplicativo da seguinte forma: "cada aplicação no NestJS tem ao menos um módulo, o raiz. O módulo raiz é o ponto inicial que o framework utiliza para construir o gráfico da aplicação, uma estrutura interna que resolve os relacionamentos de módulos, providers e suas dependências."[7]

PostgreSQL é gerenciador de banco de dados relacional, com uma série de funções relacionadas à manipulação de diversos tipos de dados, incluindo JSONs e sistemas de transações.

O projeto também conta com uma documentação gerada de forma automática, que documenta o uso de todas as rotas via uma integração do NestJS e Swagger.

2.3 Funcionalidades do Sistema

O módulo do usuário conta com funções de registro e informações de perfil. A função de registro encripta a senha digitada pelo usuário utilizando o argon2, um algoritmo para proteger a senha dos usuários, que, segundo Biryukov *et al.* (2016), é um "algoritmo de hash com alta performance, que utiliza uma quantidade configurável de memória, a fim de aumentar o custo por tentativa para sistemas que utilizam FPGAs, GPUS e módulos ASIC".[8]

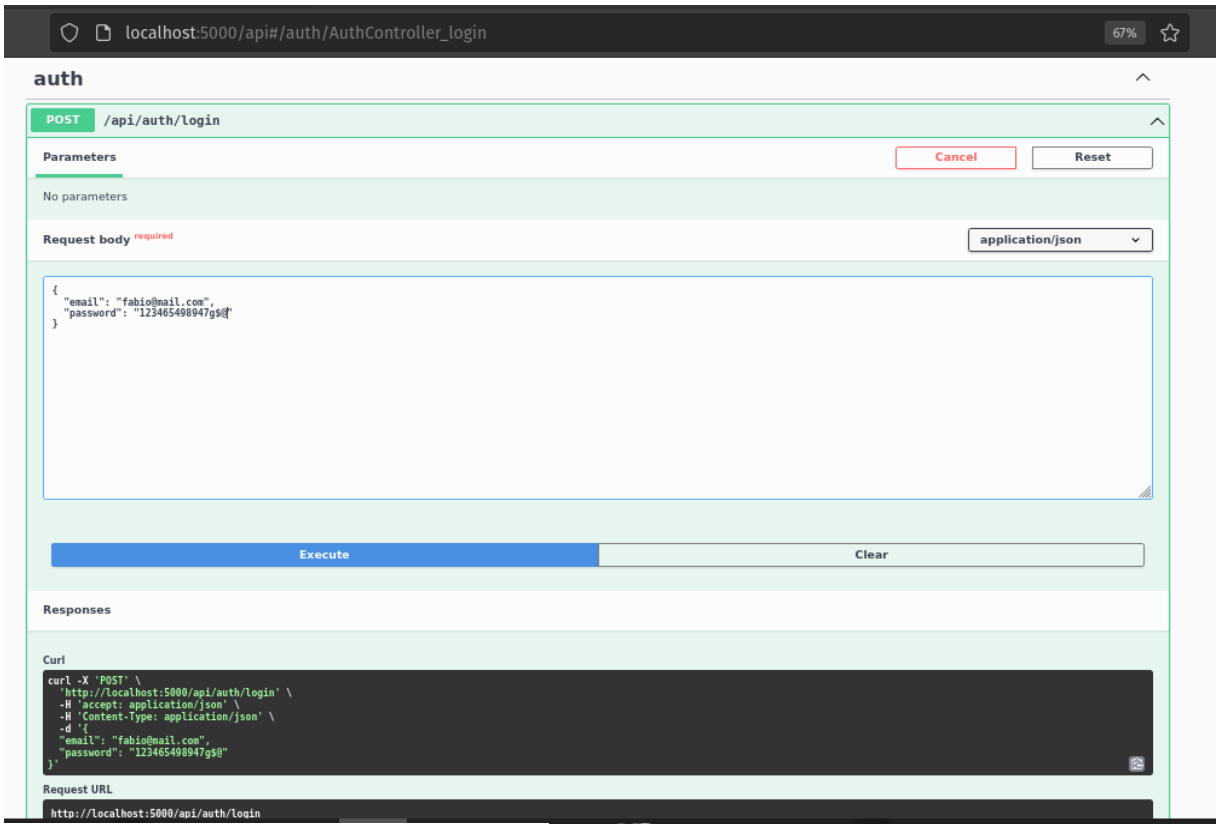
O módulo de autenticação é responsável por negar/autenticar usuários baseado nas suas credenciais. Este módulo utiliza a biblioteca Passport para lidar com os tokens de autorização. Quando emitido pelo servidor, esses tokens são do tipo JSON Web Token (JWT).

JWT é um padrão aberto de autenticação utilizado para emitir tokens que contêm informações autenticadas pelo servidor. Esses tokens são gerados utilizando o padrão de criptografia assimétrica, em que uma chave privada é guardada dentro

[6] Tradução nossa.
[7] Tradução nossa.
[8] Tradução nossa.

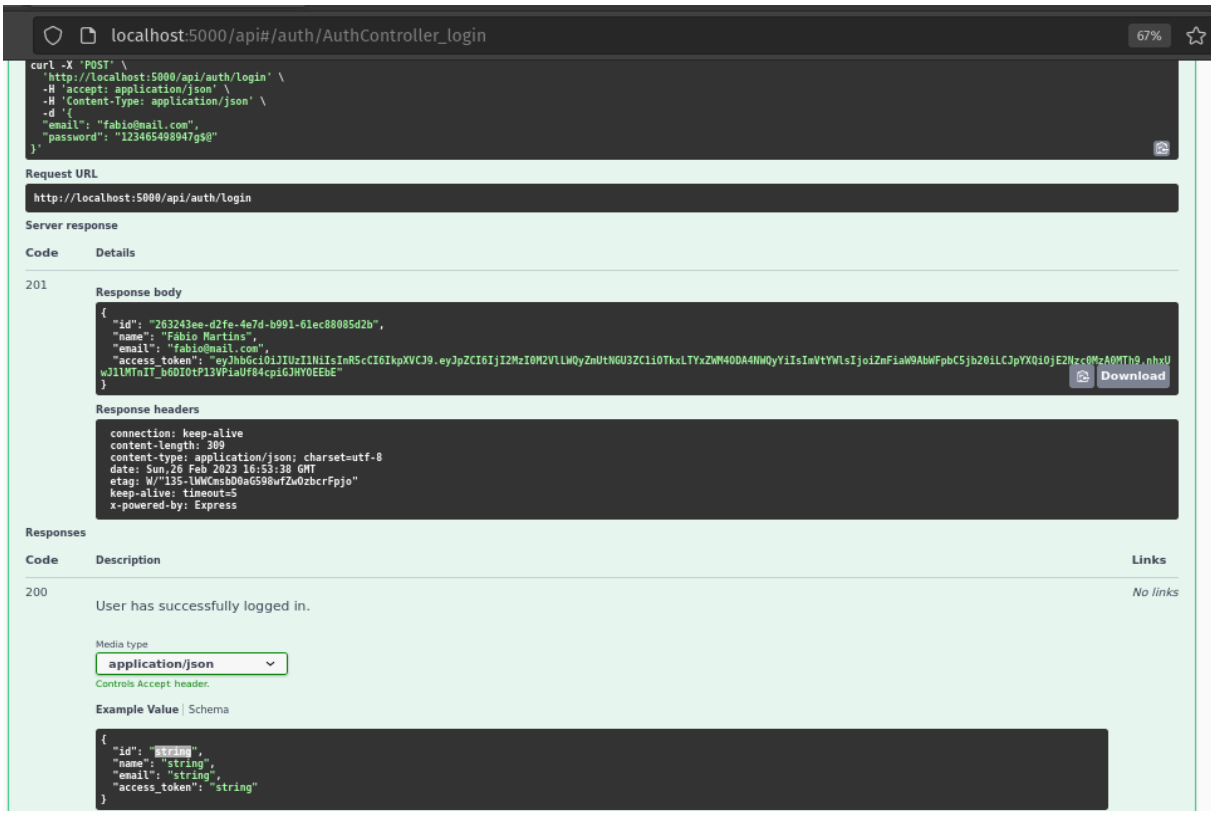
das configurações do servidor. A estrutura de um JWT é composta por uma cadeia de caracteres separados por pontos em 3 partes principais: cabeçalho, corpo e uma assinatura. O cabeçalho e o corpo são codificados utilizando base64. A garantia de que aquele token foi gerado de fato pelo servidor está na terceira parte do token, que é gerada pelo servidor a partir de sua chave privada, assinando os dados do cabeçalho e do corpo.

Figura 1 - Requisição de login documentada via Swagger (parte 1).



Fonte: captura de tela de computador pessoal

Figura 2 - Requisição de login documentada via Swagger (parte 2).



Fonte: captura de tela de computador pessoal

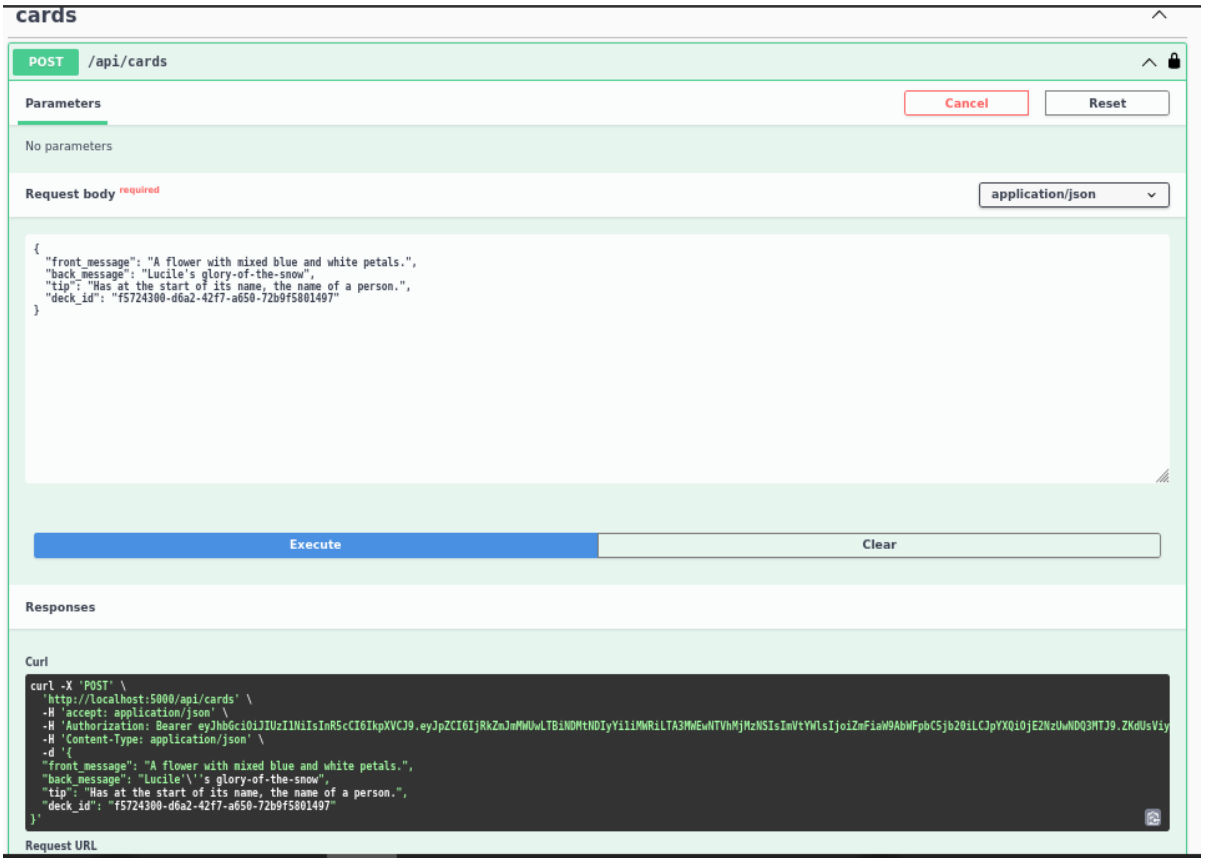
Todos os módulos, com exceção do de login e de criação de usuário, contam com um guarda que verifica a existência e validade do token, podendo levar em consideração parâmetros de emissão, como, por exemplo, data e hora de emissão, acesso a rotas específicas etc. A documentação do NestJS [20--?] define a responsabilidade dos guardas assim: “Guardas têm apenas uma responsabilidade: Eles determinam se uma devida requisição poderá ser atendida por uma rota ou não, dependendo de certas condições (como permissão, papéis, listas de controle e etc.).”

Diferente de middlewares, os guardas têm acesso ao seu contexto de execução, tornando possível saberem exatamente o que será executado depois. Eles foram desenhados de forma a serem bem parecidos com filtros de exceção, pipes e interceptores, permitindo que se possa interpor lógica de processo no momento exato dentro do ciclo de requisição/resposta, escrito de forma declarativa, ajudando a manter o código DRY[9].

[9] DRY é um acrônimo para “Don’t repeat yourself” (em português, não repita a si mesmo), um princípio de software que tenta evitar a repetição desnecessária de códigos

O sistema conta com um módulo de criação de flashcards, em que o usuário pode especificar uma mensagem principal a ser lembrada, uma dica e um deck.

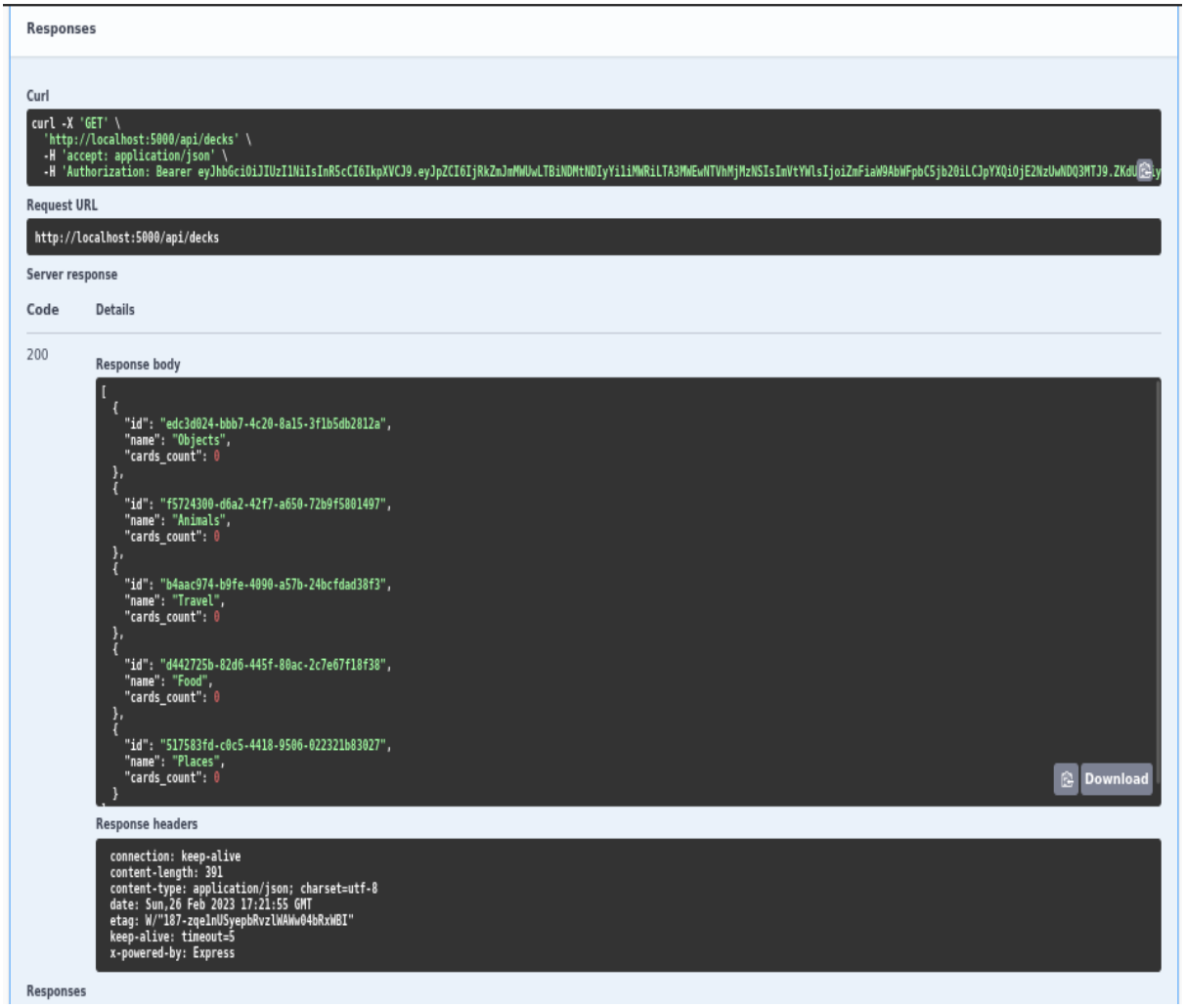
Figura 3 - Requisição de criação de um flashcard documentada via Swagger.



Fonte: captura de tela de computador pessoal

O sistema também conta um módulo pré-definido de decks, para que o usuário possa incluir e agrupar suas próprias cartas dentro desses decks para futura revisão.

Figura 4 - Requisição para pegar os decks padrões do sistema.



Fonte: captura de tela de computador pessoal

O sistema também possui um módulo de revisões de carta com tempos configuráveis. Ao cadastrar cartas no sistema, essas cartas tornam-se disponíveis para revisão com o objetivo de ajudar o usuário a aprender por repetição. Quando uma carta está sendo revisada, o usuário pode avaliá-la como “easy”, “medium” ou “hard”. Isso influencia diretamente em quanto tempo as cartas demorarão a aparecer novamente: as avaliadas como “easy” demoram 3 dias para aparecer novamente; as como “medium” demoram 8 horas aparecer novamente; e as avaliadas como “hard” demoram apenas 15 minutos.

3. Considerações finais

Logo no começo, pesquisar por conteúdos específicos de desenvolvimento mostrou-se um desafio. Já dados relacionados a pesquisas educativas e educação no geral possuem conteúdos ricos e foram encontrados com certa facilidade, o que contribuiu para a melhor compreensão do eixo temático em que o aplicativo se encaixa, levando à construção de ferramentas que têm mais possibilidade de sucesso no cumprimento de seus objetivos.

Quanto às perspectivas futuras da aplicação, temos, também, o objetivo de implementar testes e features que permitam ao usuário gravar sua pronúncia da palavra estudada e receber um feedback quanto a acertos e possíveis melhorias nessa produção oral

Por fim, a ideia do Cardery é que ele possa ser relevante para pessoas no geral e que consiga de fato ajudá-las a melhorar suas habilidades em língua inglesa; essa seria uma grande recompensa.

Referências

ABDULLAH, H; ZEKI, A; "Frontend and back-end Web Technologies in Social Networking Sites: Facebook as an Example". *In*: I**nternational Conference on Advanced Computer Science Applications and Technologies**, nº 3, 2014, Amã. Amã, 2014, p 85-89.

ALTEXSOFT. **Software Documentation Types and Best Practices.** 16 jan. 2018. Disponível em: https://blog.prototypr.io/software-documentation-types-and-best-practices-1726ca595c7f. Acesso em: 28 fev. 2023.

BIRYUKOV, Alex; DINU, Daniel; KHOVRATOVICH, Dmitry. Argon2: new generation of memory-hard functions for password hashing and other applications. In: **2016 IEEE European Symposium on Security and Privacy (EuroS&P)**. IEEE, 2016. p. 292-302.

EVANS, Eric; EVANS, Eric J. **Domain-driven design: tackling complexity in the heart of software**. Addison-Wesley Professional, 2004.

FGV – FUNDAÇÃO GETÚLIO VARGAS. **Retrospectiva 2021**: Brasil tem dois dispositivos digitais por habitante, revela pesquisa da FGV. 21 dez. 2021. Disponível em: https://portal.fgv.br/noticias/retrospectiva-2021-brasil-tem-dois-dispositivos-digitais-habitante-revela-pesquisa-fgv. Acesso em: 28 fev. 2023.

HARRINGTON, Jan L. **Relational database design and implementation**. Burlington: Morgan Kaufmann, 2016.

KUKULSKA-HULME, Agnes; TRAXLER, John; **Mobile learning**: a handbook for educator and trainers. New York: Routledge, 2005.

MILLETT, Scott; TUNE, Nick. **Patterns, principles, and practices of domain-driven design**. New York: John Wiley & Sons, 2015.

NESTJS. **Modules**. Disponível em: https://docs.nestjs.com/modules Acesso em: 12 de fev. de 2023.

OLUWATOSIN, Haroon Shakirat. Client-server model. In: **IOSR Journal of Computer Engineering**, v. 16, n. 1, p. 67-71, 2014.

PHAM, Anh Duc. **Developing back-end of a web application with NestJS framework**: Case: Integrify Oy's student management system. Bachelor's thesis, 2020.

Rahman, S. **Why I choose NestJS over other Node JS frameworks**. 20 ago. 2019. Disponível em: https://medium.com/monstar-lab-bangladesh-engineering/why-i-choose-nestjs-over-other-node-js-frameworks-6cdbd083ae67. Acesso em: 28 fev. 2023.

REDHAT. **What is an application architecture?** 09 mar. 2020. Disponível em: https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture. Acesso em: 28 fev. 2023.

WALLIS, John. **Grammatica linguae Anglicanae**. Menston: Scolar Pr., 1969.