Desenvolvimento de uma API REST para um Sistema Acadêmico de Terceiros

Ana Silva Ferreira[1], Jalves Mendonça Nicacio[2], Glauber Ventura Ferreira[3], Greyson Mascarenhas Santos Filho[3], Vanderson Silva Rodrigues[1]

[1]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Maceió, AL – Brasil

[2]Campus São Miguel dos Campos – Instituto Federal de Alagoas (IFAL)
São Miguel dos Campos, AL – Brasil

[3]Campus Palmeira dos Índios – Instituto Federal de Alagoas (IFAL)
Palmeira dos Índios, AL – Brasil

{dasilvaferreira.af, greysonmrx, vanderson99rodrigues}@gmail.com,
{jalves.nicacio,glauber.ferreira}@ifal.edu.br

Abstract. *This paper presents the roadmap in the development of an API for searching a third party Academic System (AS) deployed on Federal Institute of Alagoas. The main motivation for this development is that although academic data is directly available through database queries, the learning curve is high for developing new projects due to the complexity of AS. Such an API will allow students, teachers, and IT system analysists to develop software that use these academic data, and then, do research, development, and innovation projects, promoting the production of innovative scientific and technological knowledge. Architectural and technological decisions are discussed, and a prototype application is presented with the objective of validating the developed API.*

Keywords: *Educational Data; Enterprise Software; SPA; JSON API*

Resumo. *Neste artigo apresentam-se os caminhos seguidos no desenvolvimento de uma API de consulta de dados no Sistema Acadêmico (SA) de terceiros implantado no Instituto Federal de Alagoas. A principal razão para o desenvolvimento desta API é que, apesar dos dados estarem acessíveis através de consultas diretas ao banco de dados, a curva de aprendizagem é grande para o desenvolvimento de novos projetos em virtude da complexidade do SA. Essa API permitirá que alunos, professores e analistas de TI possam desenvolver software que utilize esses dados e, com isso, executar projetos de Pesquisa, Desenvolvimento e Inovação em diferentes áreas, estimulando a produção de conhecimento científico e tecnológico inovadores. Decisões arquiteturais e tecnológicas são discutidas e uma aplicação protótipo é apresentada com objetivo de validar a API desenvolvida.*

Palavras-chave: *Dados Educacionais; Software Corporativo; SPA; JSON API*

1. Introdução

Há alguns anos atrás o Instituto Federal de Alagoas (IFAL) iniciou o processo de implantação de um Sistema Acadêmico (SA) desenvolvido por terceiros. Dentre as motivações para a contratação e utilização desse sistema podem ser citadas:

1. **Padronização** de uso em todos os ***campi*** do IFAL, uma vez que antes do processo de implantação diferentes sistemas que possuíam o mesmo propósito eram utilizados por diferentes *campi*.
2. **Uniformização** de uso nos diferentes **níveis de ensino** ofertados pelo IFAL, visto que anteriormente utilizavam-se sistemas diferentes para cada um dos níveis.

As diversas operações utilizadas pelos usuários diariamente ao utilizarem o SA geram uma imensa quantidade de dados, de diferentes naturezas, desde frequência e notas de alunos registradas pelos professores em suas turmas, passando por mensagens postadas por alunos em fóruns de discussão e respostas para atividades e questionários eletrônicos, até o registro de acesso dos alunos em cada uma das seções do sistema.

Esses dados são de propriedade do IFAL, possuem um imenso valor para a obtenção de novas informações e a instituição poderia utilizá-los como entrada para a realização de projetos de Pesquisa, Desenvolvimento e Inovação (PDI) em diferentes áreas. Se todos esses projetos pudessem ser desenvolvidos internamente por pesquisadores do próprio IFAL, através do corpo técnico relacionado à área de TI, haveria uma liberdade maior para analisar os dados de maneira exploratória, fazer pesquisa científica e tecnológica, permitindo desenvolver novos produtos e serviços inovadores.

É importante salientar que desenvolver projetos de PDI internamente na instituição utilizando os dados dos SA é possível atualmente com a infraestrutura de software implantada. Entretanto, é uma tarefa custosa e improdutiva, uma vez que as equipes de cada um desses projetos precisariam compreender, dominar e acessar diretamente o modelo de dados relacional do sistema, composto por inúmeras tabelas. Além disso, a equipe de TI do IFAL precisaria definir mecanismos de autenticação/autorização para delimitar o acesso aos dados para cada projeto específico.

Em face dessas dificuldades, surgiu a motivação para o desenvolvimento de *Web Services* [World Wide Web Consortium 2004], mais especificamente através de uma API REST [Fielding 2000], para possibilitar a consulta de dados de atividades acadêmicas gerenciados pelo SA. Essa API pavimentará o acesso às informações, possibilitando que diversos projetos de PDI com dados acadêmicos sejam desenvolvidos de maneira mais eficiente e produtiva.

2. Definições Arquiteturais e Tecnológicas

Uma das primeiras decisões que precisaram ser tomadas pela equipe ao iniciar o desenvolvimento da API foi a respeito do formato de arquivo utilizado para disponibilizar os dados acadêmicos. Optou-se pelo formato aberto JSON (*JavaScript Object Notation*) [Bray, T 2017; Ecma International 2017] em virtude da sua ampla utilização e consequente suporte ferramental.

Com o objetivo de padronizar a estrutura dos arquivos JSON trocados entre a API e as aplicações cliente, assim como padronizar o mecanismo utilizado para criação,

atualização, exclusão, relações, filtragem, ordenação e paginação de recursos, decidiu-se seguir a especificação JSON API [JSON API 2013]. Aderindo a esta especificação, a implementação da API é facilitada pois a equipe pode se concentrar diretamente no desenvolvimento das entidades disponibilizadas pela própria API, uma vez que várias demandas comuns a toda API já são resolvidas pela própria especificação.

Dentre as diversas implementações disponíveis da especificação JSON API, considerando que a API do SA está sendo implementada utilizando o arcabouço Java/Spring Boot [Spring 2005], após estudos e testes optou-se pela utilização da biblioteca Katharsis [Read The Docs 2017], uma vez que ela provê nativamente integração com o Spring Boot para receber/responder requisições de acordo com a especificação JSON API.

Já com relação à camada de dados, o SA utiliza o banco de dados relacional PostgreSQL. O esquema do banco de dados do SA foi então estudado e sete tabelas foram selecionadas inicialmente para consulta: `pessoa`, `discente`, `curso`, `componente_curricular`, `situacao_matricula`, `matricula_componente`, e `componente_curricular_detalhes`. A seleção dessas tabelas se deu em virtude de elas armazenarem dados relativos a uma parte do escopo definido inicialmente para o desenvolvimento da API, a saber: aluno, disciplina, curso e nota.

2.1. Integrando a Camada Web JSON com a Camada de Acesso a Dados

Levando em consideração as tabelas citadas anteriormente, realizou-se então a especificação, o projeto e a implementação da API de consulta. Na Figura 1 ilustram-se as classes e interfaces que foram implementadas para permitir a consulta via API aos dados armazenados na tabela `discente`. Ao longo desta seção descrevem-se alguns detalhes relacionados a essa implementação, deixando claro que para cada uma das outras tabelas os caminhos seguidos foram análogos.

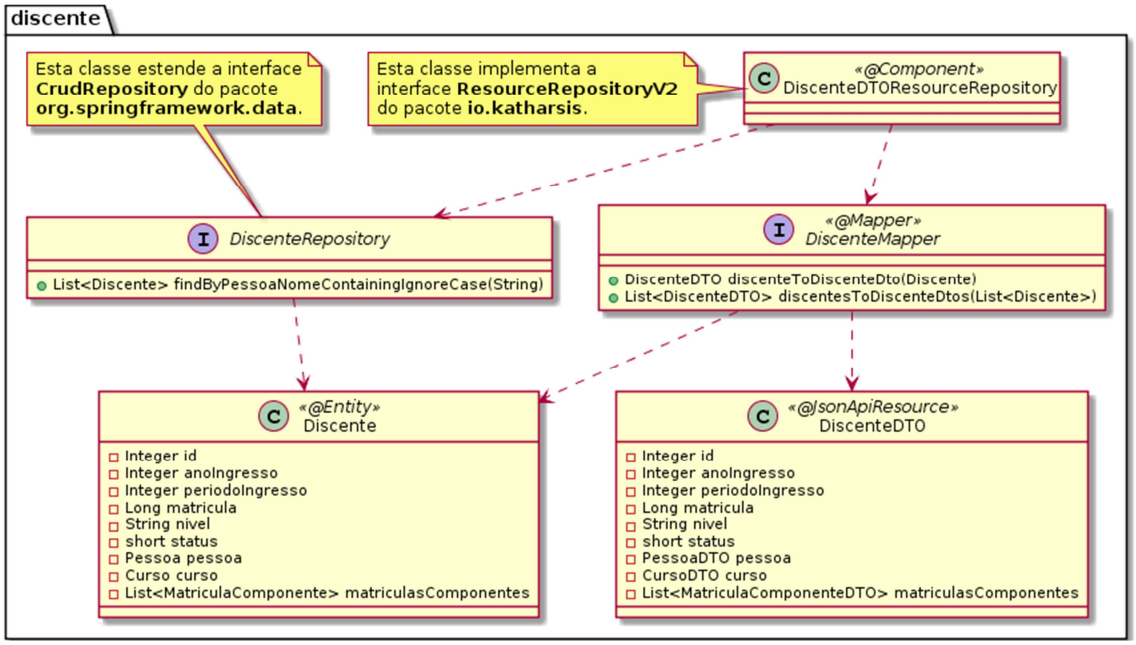

Figura 1. Diagrama de classes do pacote *discente*

A classe Discente representa um discente do IFAL que está armazenado na tabela discente do banco de dados. Essa classe possui a anotação @Entity, que faz parte do pacote de persistência de dados javax.persistence ilustrado na Figura 2, na qual são representados os pacotes de bibliotecas de terceiros utilizados no sistema. O papel dessa anotação é possibilitar que as instâncias da classe Discente sejam correlacionadas às linhas da tabela discente no banco de dados.

Já a interface `DiscenteRepository` define o comportamento relacionado à camada de acesso a dados do sistema. Através da extensão da interface `CrudRepository` do pacote `org.springframework.data` ilustrado na Figura 2, todas as funcionalidades habituais relativas às operações CRUD (*Create*, *Read*, *Update* e *Delete*) ficam disponíveis para gerenciar os dados dos discentes a partir da interface `DiscenteRepository`. Consultas específicas podem ainda ser implementadas, bastando para isso que haja apenas a declaração de métodos na interface `DiscenteRepository` utilizando um conjunto de palavras-chave de acordo com convenções de nomenclatura do Spring. Por exemplo, através da declaração do método `findByPessoaNomeContainingIgnoreCase` na interface `DiscenteRepository` é possível recuperar do banco de dados uma lista com todos os discentes cujo `nome` da `Pessoa` ao qual ele está associado contém a `String` passada como parâmetro do método, ignorando caracteres maiúsculos e minúsculos.

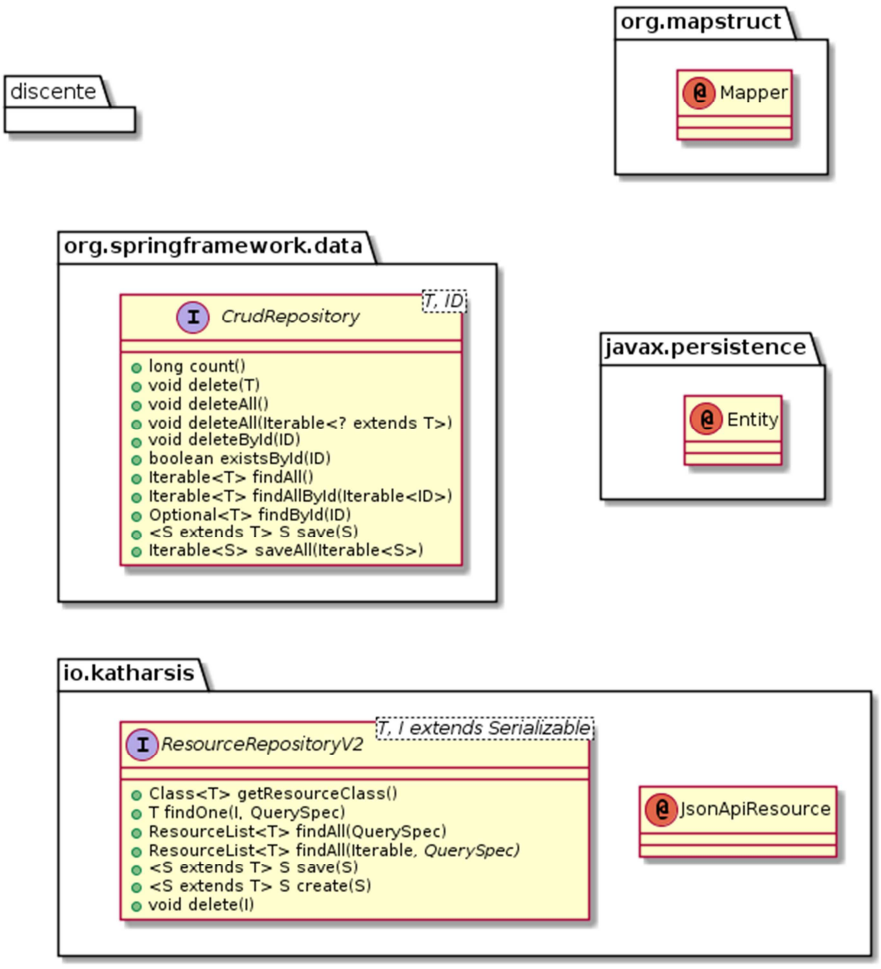

Figura 2. Diagrama de pacotes de bibliotecas de terceiros

Outro repositório implementado no pacote `discente`, ora descrito, é a classe `DiscenteDTOResourceRepository`. Essa classe também é responsável pelo comportamento relativo às operações CRUD, porém desta vez não para a interação com o banco de dados, mas sim para o gerenciamento de recursos REST a partir da disponibilização de requisições HTTP de acordo com a especificação JSON API. Ou seja, através dessa classe é possível por exemplo, acessar a seguinte URL através de um método HTTP GET para consultar seguindo a especificação o recurso `discente` que possui a `matricula` 214362: `/api/discentes/214362`. Para isso, essa classe implementa a interface `ResourceRepositoryV2` do pacote `io.katharsis`, que por sua vez especifica o comportamento necessário para realizar operações em recursos REST de acordo com a JSON API.

É importante destacar que a classe `DiscenteDTOResourceRepository` não retorna diretamente para os seus clientes instâncias da classe `Discente`. Com o objetivo de evitar um forte acoplamento entre os dados expostos pela API e a estrutura utilizada no armazenamento, optou-se por utilizar o padrão de projeto de software denominado *Data Transfer Object* (DTO) [Fowler 2002] para definir a interface dos dados que serão retornados pela API via HTTP. Esse padrão é implementado pela classe `DiscenteDTO`, que por sua vez possui a anotação `@JsonApiResource`, pertencente ao pacote `io.katharsis`. Essa anotação é responsável por marcar as classes que são recursos REST, podendo então ser alvo de consultas através da API. O valor do atributo `type` dessa anotação é utilizado para criar a URL utilizada para acessar o recurso; no caso de `DiscenteDTO` o valor desse atributo é a `String` *discentes*.

Para finalizar a implementação do padrão DTO, a classe `DiscenteDTO ResourceRepository` faz uso da interface `DiscenteMapper`. Essa interface é responsável por realizar o mapeamento de instâncias da classe `Discente` para a instâncias da classe `DiscenteDTO`. Para isso, a interface `DiscenteMapper` possui a anotação `@Mapper`, disponibilizada pela biblioteca MapStruct [Morling 2013] e presente no pacote `org.mapstruct`. Esta biblioteca é um gerador de código que simplifica a implementação de mapeamentos entre Java *beans*. Atualmente a classe `DiscenteDTO` é praticamente uma réplica da classe `Discente`, possuindo modificações apenas nas suas associações para referenciar outros DTOs implementados no sistema, simplificando, portanto, o mapeamento realizado via MapStruct. Apesar dessa semelhança, a API está preparada para evoluir sem estar acoplada ao modelo de armazenamento de dados, bastando para tanto que os DTOs sejam alterados e os Mappers atualizados.

2.2. Requisições e Respostas Utilizando JSON API

Na Figura 3 ilustra-se o conteúdo da resposta, em conformidade com a especificação JSON API, para a consulta utilizando a API REST por um determinado discente cadastrado no SA. Por questões de sigilo, são apresentados dados de teste. Entretanto, os atributos exibidos no arquivo JSON de resposta condizem com os dados obtidos do banco relacional do SA. Os dados ilustrados na Figura 3 tem sido analisados através da ferramenta Postman [Postdot Technologies 2012], utilizada durante o desenvolvimento para realizar consultas e verificar as respostas em formato JSON.

```
{
  "data": {
    "id": "214362",
    "type": "discentes",
    "attributes": {
      "anoIngresso": 2018, "matricula": 2018877872, "nivel": "T", "periodoIngresso": 1, "status": 1
    },
    "relationships": {
      "pessoa": {
        "data": { "id": "18132", "type": "pessoas" },
        "links": {…}
      },
      "matriculasComponentes": {
        "links": {…}
      },
      "curso": {
        "data": { "id": "8358486", "type": "cursos" },
        "links": {…}
      }
    },
    "links": {…}
  },
  "included": [
    {
      "id": "8358486",
      "type": "cursos",
      "attributes": {
        "codigo": "86", "ativo": true, "nome": "CURSO ABC", "nivel": "T"
      },
      "links": {…}
    },
    {
      "id": "18132",
      "type": "pessoas",
      "attributes": {
        "nome": "FULANO DE TAL"
      },
      "links": {…}
    }
  ]
}
```


Figura 3. Resposta em formato JSON para a consulta de discente do SA

O arquivo JSON ilustrado na Figura 3 pode ser dividido em duas partes principais: o intervalo das linhas 2-33 (membro "data") representa os atributos e os relacionamentos da entidade consultada (`discente` com `id` 214362), enquanto que o intervalo das linhas 34-55 (membro "included") representa um *array* com os recursos relacionados aos dados presentes no membro "data". Ou seja, enquanto a linha 10 indica que o discente com `id` 214362 está relacionado à `pessoa` com `id` 18132, as linhas 45-54 detalham os dados desse relacionamento, apresentando os seus atributos. De forma análoga, o mesmo acontece para os `cursos`: a linha 23 indica a presença do relacionamento, enquanto que as linhas 35-44 apresentam os seus atributos.

É importante ressaltar que a recuperação dos atributos dos relacionamentos não é realizada por padrão; o membro "included" não é retornado por padrão no JSON. Para que isso ocorra é necessário informar explicitamente na URL de consulta quais relacionamentos devem ser recuperados. Esse comportamento da JSON API é importante pois torna flexível para o cliente a definição dos dados necessários em cada situação. No caso dos dados ilustrados na Figura 3, a URL utilizada para a consulta foi `/api/discentes/214362?include=pessoa,curso`. Através do parâmetro da requisição `include` torna-se possível indicar o conjunto de relacionamentos que devem ser retornados. Como o relacionamento `matriculasComponentes` (linha 16) não foi repassado na URL, esse relacionamento não foi listado no membro "included" do JSON de resposta.

A URL descrita anteriormente foi requisitada ao servidor utilizando o método HTTP GET. Considerando que o objetivo da API é possibilitar a *consulta* de dados, apenas o método HTTP GET (recuperação de recursos) é de fato implementado. Os outros métodos HTTP utilizados nas APIs REST (POST para a criação de recursos, PUT/PATCH para atualização e DELETE para exclusão) não são permitidos, fazendo com que um código de erro seja retornado para a aplicação cliente.

3. Prototipagem de uma Aplicação para Validação da API

Com o objetivo de validar a API em desenvolvimento, decidiu-se implementar uma aplicação protótipo, direcionada para os Coordenadores de Curso do IFAL, que permite a visualização das notas de um determinado discente em um formato de Gráfico de Radar. A ideia é que a partir de uma única visualização gráfica o coordenador possa ter uma visão ampla do desempenho quantitativo do discente. Na Figura 4 ilustra-se um exemplo deste gráfico para um determinado discente, cuja identificação foi omitida.

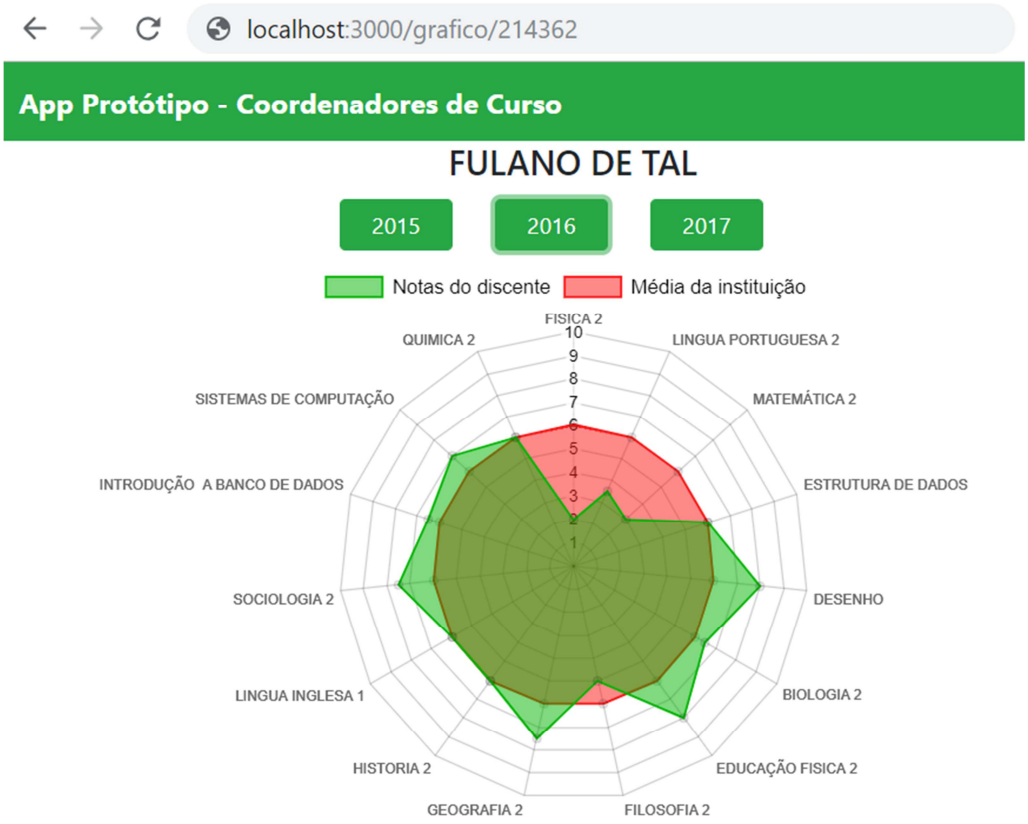

Figura 4. Gráfico de radar ilustrando as médias de um discente e da instituição em diferentes disciplinas

Após uma busca pelo discente em uma tela anterior, utilizando diferentes critérios, a aplicação protótipo exibe o gráfico de radar, no qual cada disciplina em que o discente está matriculado é visualizada em um dos seus vértices. Além destas informações, exibe-se também uma curva com as notas do discente (verde) e outra curva utilizada para comparação com a média regulamentar da instituição (rosa). É possível identificar no gráfico, por meio das situações em que, a partir do centro, a curva da média da instituição ultrapassa a curva das notas do discente, que o mesmo

obteve um desempenho abaixo da média da instituição nas disciplinas de Física 2, Língua Portuguesa 2, Matemática 2 e Filosofia 2. As disciplinas em que o discente obteve o melhor desempenho são aquelas cujos pontos na curva de notas do discente estão mais próximos das bordas do gráfico: nesse exemplo, Desenho e Educação Física 2. Através dessa análise visual é possível identificar que a maioria das notas desse discente estão próximas da média da instituição e que o mesmo está com dificuldade em quatro disciplinas. Por outro lado, ao analisar o gráfico de radar de outro discente, ilustrado na Figura 5, é possível perceber que o seu desempenho quantitativo é ótimo. Em todas as disciplinas a curva que indica a nota do discente sempre supera a curva da média da instituição e está bem próxima das bordas do gráfico.

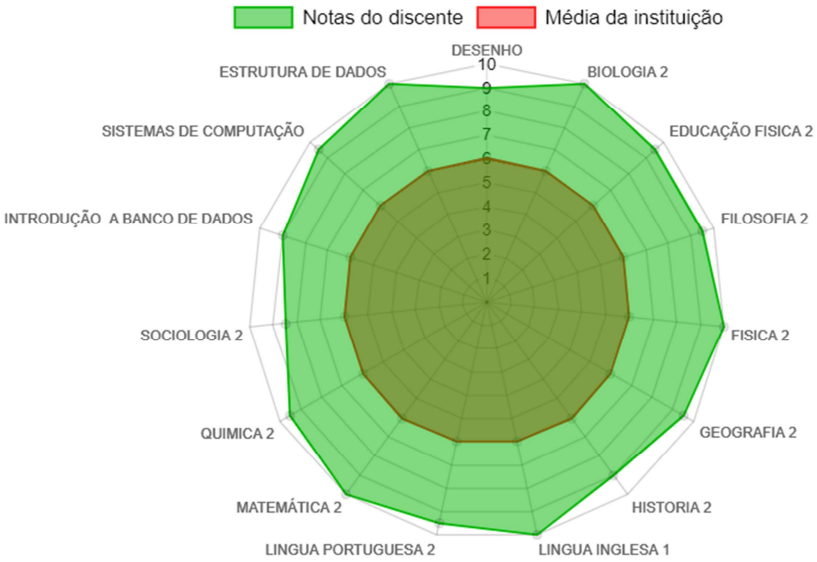

Figura 5. Gráfico de radar de um discente com ótimo desempenho quantitativo

Os gráficos ilustrados na Figura 4 e na Figura 5 foram gerados dinamicamente através da aplicação protótipo que foi desenvolvida seguindo a estrutura de uma SPA (*Single Page Application*). Para isso, utiliza-se o arcabouço JavaScript Vue.js [Vue.js 2014] no lado cliente em conjunto com a biblioteca JavaScript Chart.js [Chart.js 2013] para efetuar a geração dos gráficos. A aplicação protótipo se integra com a API REST através de requisições e respostas em formato JSON, de acordo com os exemplos descritos na Seção 2.2. Desta maneira, o conjunto de dados exibidos no gráfico relativos ao discente, às disciplinas, e às notas estão sendo recuperados dinamicamente através de consultas a API REST.

Pretende-se complementar estes dados com uma categorização das disciplinas de acordo as Áreas do Conhecimento utilizadas no ENEM (Matemática e Suas Tecnologias, Ciências Humanas e Suas Tecnologias, etc.), de maneira que o coordenador possa ter uma visão mais clara caso o discente esteja eventualmente tendo dificuldades em alguma área do conhecimento específica.

4. Considerações Finais e Perspectivas Futuras

Neste artigo apresentaram-se os caminhos tomados para a especificação, o projeto e a implementação de uma API de consulta para facilitar o acesso aos dados de atividades acadêmicas gerenciadas pelo SA de terceiros utilizado no IFAL. Pretende-se com a

disponibilização desta API que toda a comunidade da instituição seja beneficiada, uma vez que, dependendo da natureza dos projetos de PDI desenvolvidos utilizando a API, pais, alunos, professores, técnicos, gestores acadêmicos, etc. poderão usufruir das informações obtidas.

Além das decisões arquiteturais e tecnológicas tomadas ao longo do desenvolvimento, descreveu-se também o funcionamento da aplicação protótipo que está sendo implementada com o intuito de validar os artefatos de software produzidos pela API. Sua principal funcionalidade é exibir uma visão quantitativa unificada das notas de um determinado aluno para os coordenadores de curso.

Os próximos passos para evolução da API REST dizem respeito à inclusão de informações de frequência dos alunos e, em seguida, de consulta de professores. Já com relação à aplicação, além da categorização das disciplinas de acordo com as áreas de conhecimento do ENEM, pretende-se também adicionar informações de frequência na visualização gráfica para que se possa verificar a correlação entre notas e frequência, permitindo a identificação de situações que antecipem eventuais evasões.

Uma funcionalidade transversal que precisa necessariamente ser incorporada na API antes da sua efetiva disponibilização para a comunidade acadêmica é um mecanismo de autorização, uma vez que é inconcebível do ponto de vista da segurança da informação expor os dados acadêmicos sem um mecanismo robusto para garantir que apenas as pessoas autorizadas possam acessá-los. Um estudo preliminar já foi realizado acerca do arcabouço OAuth 2.0 [Eloy 2017], com um foco especial no *grant type Authorization Code*, que deve ser utilizado na API como mecanismo de autorização. Além disso, o protocolo OAuth 2.0 "obriga que todas as requisições sejam feitas utilizando uma camada de transporte segura através de *Transport Layer Security* (TLS) ou *Secure Socket Layer* (SSL)" [Eloy 2017].

Com relação à segurança da informação por parte dos desenvolvedores de software que utilizarão a API, pretende-se disponibilizar um ambiente de homologação com uma base de dados de teste para que as aplicações implementadas possam ser testadas antes de serem implantadas no ambiente de produção com os dados reais. Isto evitará que os dados reais sejam coletados de forma indevida. Uma vez em ambiente de produção, apenas os usuários autorizados (e.g. alunos, professores, coordenadores, etc.) mediante login e senha poderão acessar os seus respectivos dados.

Um outro trabalho que poderá ser desenvolvido é um estudo comparativo para analisar o processo de desenvolvimento de uma mesma aplicação de dados acadêmicos com e sem a utilização da API. Sem a API, os dados seriam acessados através de consultas ao banco de dados. O intuito do estudo é medir os ganhos de produtividade da equipe de desenvolvimento ao utilizar a API em detrimento do acesso direto ao banco.

Por fim, é importante destacar que ao tempo em que esta API estiver disponibilizada para uso, diferentes aplicações e projetos poderão ser desenvolvidos, incluindo, mas não se limitando a: painéis de indicadores gerenciais (*dashboards*); mineração de dados educacionais [Romero e Ventura 2010]; ferramentas para auxiliar gestores na tomada de decisão; integração com aplicações educacionais de terceiros, tais como Kahoot [Kahoot 2013], Socrative [Showbie 2014], Google Classroom [Google 2014], Gradepen [Gradepen Corporation 2014]; novas interfaces gráficas para acesso a funcionalidades já existentes; desenvolvimento de novas aplicações.

5. Agradecimentos

Os autores agradecem o apoio e o financiamento concedido pelo IFAL para a execução deste trabalho. Os autores Ana e Vanderson foram discentes do IFAL, enquanto que o autor Greyson é discente do IFAL e é bolsista do Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI), tendo como fonte pagadora o IFAL.

Referências

Bray, T. (2017). IEFT RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format. Disponível em: <https://tools.ietf.org/html/rfc8259>

Chart.js. (2013). Open source HTML5 Charts for your website. Disponível em: <https://www.chartjs.org/>

Ecma International. (2017). Standard ECMA-404 - The JSON Data Interchange Syntax. Disponível em: <https://www.ecma-international.org/publications/standards/Ecma-404.htm>

Eloy, A. (2017). OAuth 2.0: Proteja suas aplicações com Spring Security OAuth2. São Paulo: Casa do Código.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Irvine, CA, EUA. Tese de Doutorado.

Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.

Google. (2014). Google Classroom. Disponível em: <https://classroom.google.com/>

Gradepen Corporation. (2014). Gradepen**.** Disponível em: <https://gradepen.com/>

JSON API. (2013). JSON API - A specification for building APIs in JSON. Disponível em: <http://jsonapi.org/>

Kahoot. (2013). Kahoot! - Learning Games - Make Learning Awesome. Disponível em: <https://kahoot.it/>.

Morling, G. (2013). MapStruct - Java bean mappings, the easy way**.** Disponível em: <http://mapstruct.org/>

Postdot Technologies. (2012). Postman - API Development Environment**.** Disponível em: <https://www.getpostman.com/>

Read The Docs. (2017). Katharsis JSON-API documentation. Disponível em: <http://katharsis-jsonapi.readthedocs.io/en/latest/>

Showbie. (2014). Socrative**.** Disponível em: <https://www.socrative.com/>

Spring. (2005). Spring Boot. Disponível em: <https://spring.io/>

Romero, C.; Ventura, S. (2010). Educational Data Mining: A Review of the State of the Art. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), v. 40, n. 6, p. 601 - 618.

Vue.js. (2014). The Progressive JavaScript Framework. Disponível em: <https://vuejs.org/>

World Wide Web Consortium. (2004). Web Services Architecture. Disponível em: <https://www.w3.org/TR/ws-arch/>