NIVERSIDADE FEDERAL DE ALAGOAS-UFAL
CAMPUS DE ARAPIRACA
CIÊNCIA DA COMPUTAÇÃO

ANTONIO ANDRADE GOMES JÚNIOR

UMA API PARA PROVER SERVIÇOS DE VISUALIZAÇÃO DE DADOS PARA VISUALIZAÇÃO DA CARGA DE TRABALHO EM UM CONTEXTO ACADÊMICO

ARAPIRACA
2022

Antonio Andrade Gomes Júnior

Uma API para prover serviços de visualização de dados para visualização da carga de trabalho em um contexto acadêmico

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus de Arapiraca.

Orientador: Prof. Dr. Alexandre de Andrade Barbosa

Arapiraca
2022

Universidade Federal de Alagoas – UFAL
Campus Arapiraca
Biblioteca *Campus* Arapiraca - BCA

G633a Gomes Júnior, Antonio Andrade
Uma API para prover serviços de visualização de dados para visualização da carga de trabalho em um contexto acadêmico / Antonio Andrade Gomes Júnior. – Arapiraca, 2022.
28 f.: il.

Orientador: Prof. Dr. Alexandre de Andrade Barbosa.
Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação.) - Universidade Federal de Alagoas, *Campus* Arapiraca, Arapiraca, 2022.
Disponível em: Universidade Digital (UD) – UFAL (*Campus* Arapiraca).
Referências: f. 28.

1. Programação (Computadores) 2. Programação de sistemas (Computação) 3. Software - Desenvolvimento 4. Carga de trabalho I. Barbosa, Alexandre de Andrade II. Título.

CDU 004

Bibliotecário responsável: Gerlane Costa Silva de Farias
CRB - 4 / 1802

Antonio Andrade Gomes Júnior

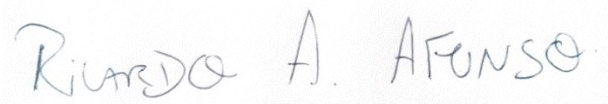
Uma API para prover serviços de visualização de dados para visualização da carga de trabalho em um contexto acadêmico

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus de Arapiraca.

Data de aprovação: 15 de julho de 2022.

Banca Examinadora

Prof. Dr. Alexandre de Andrade Barbosa Universidade Federal de Alagoas Campus Arapiraca
Orientador



Prof. Dr. Ricardo Alexandre Afonso
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

Prof. Me. Rômulo Nunes de Oliveira
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

Dedico este trabalho a minha família, a todos os amigos quefiz ao longo da vida e, por último e não menos importante, à mim mesmo.

AGRADECIMENTOS

Agradeço, primeiramente, à Deus por possibilitar que meus objetivos sejam alcançados, durante todos meus anos de estudos.

Agradeço aos meus pais, Quitéria Cristina Pereira e Antonio Andrade Gomes, e a minha avó Lusinete Daniel Pereira; por serem as pessoas que me deram apoio, carinho, compreensão e me deram força durante minha jornada na graduação. Amo muito cada um!

Gostaria de agradecer ao meu orientador, Alexandre de Andrade Barbosa; pela dedicação, paciência e aprendizado que adquiri através de sua orientação. Muito obrigado!

Agradeço também ao corpo docente do curso, constituído por pessoas fantásticas que inspiram os discentes à busca pelo novo na área da Computação. Muito obrigado!

Agradeço aos meus grandes amigos de faculdade: Júlio Miguel, Jonathas Thiago e Lucas Ferreira. Foram fundamentais para minha vida profissional e este projeto. Compartilhamos conhecimento, trabalhamos em conjunto e com isso cada um evoluiu. Obrigado caras!

Por último; sou grato a todos os familiares, amigos e colegas que não foram citados aqui mas contribuíram na minha jornada na graduação. Obrigado!

É mais fácil ser o primeiro, do que
continuar a ser o primeiro.

Bill Gates

RESUMO

As instituições de ensino superior usam diversos sistemas para promover o acesso à informações acadêmicas de discentes, docentes, turmas e disciplinas. No entanto, percebeu-se que tais sistemas não disponibilizam recursos voltados à visualização de carga de trabalho. Um recurso de visualização de carga de trabalho é importante para exibir, de forma rápida e intuitiva, um panorama geral de atividades de um usuário, onde ele pode utilizar essa informação para gerenciar sua agenda pessoal de forma mais eficiente. Diante disso, este trabalho propõe uma API com o foco em disponibilizar serviços de visualização de dados para visualização da carga de trabalho em um contexto acadêmico.

Palavras-chave: API; carga de trabalho; acesso.

ABSTRACT

Higher education institutions use several systems to promote access to academic information for students, professors, classes and subjects. However, it was noticed that such systems do not provide resources for the visualization of workload. A workload view feature is important for quickly and intuitively displaying an overview of a user's activities, where they can use this information to manage their personal schedule more efficiently. Therefore, this work proposes an API focused on providing data visualization services for workload visualization in an academic context.

Keywords: API; workload; access.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface (Interface de Programação de Aplicações)
REST	Representational State Transfer (Transferência de Estado Representacional)
HTTP	Hyoertext Transfer Protocol (Protocolo de Transferência de Hipertexto) RBAC Role-Based Access Control (Controle de Acesso Baseado na Função)
SA	Sistema Acadêmico
UFAL	Universidade Federal de Alagoas IFAL Instituto Federal de Alagoas
RU	Restaurante Universitário
UFRGS	Universidade Federal do Rio Grande do Sul
URI	Uniform Resource Identifier (Identificador de Recursos Universal) URL Uniform Resource Locator (Localizador de Recursos Universal) JSON JavaScript Object Notation
XML	Extensible Markup Language
VD	Visualização de Dados
CA	Centro Acadêmico
MVP	Minimum Viable Product (Produto Mínimo Viável)
UML	Unified Modeling Language (Linguagem de Modelagem Unificada) SGBD Data Base Management System (Sistema de Gerenciamento de Banco de Dados)
ORM	Object-relational Mapping (Mapeamento Objeto-relacional) CRUDCreate, Read, Update and Delete (Criar, Ler, Atualizar e Deletar)

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos têm sido comum as instituições de ensino superior adotarem diversos sistemas para promover o acesso a dados de docentes, discentes, turmas e disciplinas.

Nesse contexto, em cada semestre letivo são ofertadas turmas de diferentes discipli- nas. Um docente é alocado como responsável para cada turma, onde são cadastradas diversas atividades. Um conjunto de discentes participantes de uma turma deve então realizar as atividades cadastradas pelo docente responsável. Tais sistemas visam facilitar ofluxo de trabalho relacionado a gerenciamentos de turmas, discentes e atividades.

Dessa forma, durante o semestre letivo, possuir um sistema de software que possibilite a visualização da carga de trabalho das turmas, com base nas atividades ministradas pelos docentes, de forma rápida e intuitiva, e com uma listagem geral das atividades, possibilitará uma melhora na negociação para agendamento de futuras atividades e ajudará os usuários em sua organização pessoal.

Nesse contexto, em momentos do semestre em que existe uma alta frequência de atividades, nota-se uma sobrecarga nos discentes, que tem que se dedicar as atividades, e nos docentes que precisam avaliar e reportar os resultados dos trabalhos.

Conhecer o planejamento de atividades que cada docente irá ministrar, se torna importante para o discente estabelecer um cronograma de estudos e gerenciar seu tempo durante o semestre letivo. Pois é por meio do planejamento que o ambiente a sua volta é controlado consonante com seus anseios e necessidades (PURCELL, 2020).

Desse modo, neste trabalho é apresentado o desenvolvimento de uma API para gerenciamento acadêmico que tem como foco fornecer recursos para visualização da carga de trabalho, que será calculada de acordo com as atividades cadastradas pelos docentes de cada turma. Essa ferramenta se faz útil a trabalhos futuros que visam a construção de interfaces que irão consumir a API e exibir esses dados aos usuáriosfinais, como também para melhorias que se utilizem de outras técnicas para calcular a carga de trabalho.

Diante disso, o presente trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados os trabalhos relacionados; no Capítulo 3 é descrita a fundamentação teórica; no Capítulo 4 é apresentado o processo de desenvolvimento do software Atena e de suas funcionalidades; efinalmente no Capítulo 5 são apresentadas as consideraçõesfinais e as próximas etapas a serem executadas

2 TRABALHOS RELACIONADOS

Nesta seção temos por objetivo mostrar alguns trabalhos encontrados na literatura que se assemelham com uma API para prover serviços de visualização de dados para visualização de carga de trabalho em um contexto acadêmico.

2.1 ENTREGANDO RECURSOS PARA APLICAÇÕES MULTIPLATAFORMA COM A API UFRGS

No artigo (DIAS *et al.*, 2019) é tido por objetivo apresentar uma API para integração com aplicativos da Universidade Federal do Rio Grande do Sul (UFRGS). A API provê recursos que facilitam o uso da biblioteca, do Restaurante Universitário (RU) e de outras aplicações *frontend*.

A API foi projetada usando o estilo arquitetural REST com a linguagem PHP e utiliza

RBAC integrado ao *framework* OAuth 2.0 para assegurar políticas de acesso diferenciadas a cada perfil de usuário (DIAS *et al.*, 2019).

No artigo é apresentado, em suas considerações finais, que a separação de deveres do *frontend* e *backend* aprimorou o fluxo de trabalho e que o acesso a API deverá ser ampliado para novas aplicações *frontend* visando aprimorar os processos de trabalho na Universidade Federal do Rio Grande do Sul. De acordo com o que foi apresentado, não há citações sobre visualização de carga de trabalho dos discentes ou docentes.

2.2 DESENVOLVIMENTO DE UMA API REST PARA UM SISTEMA ACADÊMICO DE TERCEIROS

No artigo (FERREIRA *et al.*, 2018) é tido por objetivo demonstrar as motivações e definições arquiteturais e tecnológicas para implementação de uma API, criada após o Instituto Federal de Alagoas (IFAL) implantar um Sistema Acadêmico (SA) desenvolvido por terceiros.

A principal razão para o desenvolvimento da API é a grade curva de aprendizagem para o desenvolvimento de novas aplicações com os dados do Sistema Acadêmico adotado. Com o uso da API foram obtidos dados Padronizados e Uniformizados.

A API foi projetada usando o estilo arquitetural REST e implementada com o uso da linguagem Java. O comparativo para se analisar o processo de desenvolvimento de uma mesma aplicação de dados acadêmicos com e sem a utilização da API é dado como trabalho futuro

neste arigo.

2.3 COMPARATIVO ENTRE OS TRABALHOS RELACIONADOS

Nas subseções anteriores foram apresentados dois trabalhos relacionados, onde em ambos foi descrito o desenvolvimento de uma API para prover serviços para aplicações futuras ou já desenvolvidas para duas instituições de ensino.

Sendo que no primeiro artigo, a API desenvolvida deve possibilitar a comunicação entre os aplicativos e a base de dados da Universidade e com potencial para ser usada em futuras aplicações de uso corrriqueiro da Universidade, como o acompanhamento de frequência dos alunos. Já o segundo artigo, a API deve possibilitar a consulta de dados de atividades acadêmicas gerenciados pelo SA do IFAL para produção de projetos de pesquisa, desenvolvimenbto e inovação de maneira mais eficiente e produtiva.

O grande diferencial entre os trabalhos relacionados e o trabalho proposto é a proposta apresentada em cada um. Nofim as APIs desenvolvidas servirão para entregar rescursos para outras aplicações*frontend*, mas no trabalho proposto o foco é fornecer recursos para visualização da carga de trabalho e listagem de eventos acadêmicos das turmas.

Ambos os artigos possuem semelhanças com o trabalho proposto, pois além de possuírem comofim entregar dados para diferentes aplicações acadêmicas e podendo ajudar na produção de projetos de pesquisa, utilizam estilo arquitetural REST no desenvolvimento da aplicação.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos principais que foram utilizados neste trabalho. Tal discussão é necessária para facilitar o entendimento da ferramenta proposta.

3.1 API

API é um acrônimo para*Application Programming Interface*, ou Interface de Programação de Aplicação. API é um termo para designar uma interface de comunicação que um sistema oferece para que outros acessem suas funções, dados e recursos sem que o software ou plataforma externa precise saber como eles foram implementados. Trata-se de um conjunto de rotinas para facilitar a integração entre diferentes sistemas.

O desenvolvimento de uma API e seu uso por outros sistemas, não está ligado a uma linguagem de programação específica. Uma API pode ser escrita em Javascript, e ser consumida usando outro sistema usando Dart por exemplo.

Das diversas maneiras de integração entre clientes e API, trataremos das interações usando a Internet, com o protocolo HTTP e suas requisições. Quando uma API se utiliza do protocolo HTTP estamos falando de*web services*.

Os recursos de uma API utilizam XML ou JSON como formato de comunicação. Quando um cliente realiza uma requisição, ele espera uma resposta em XML ou JSON; quando a API necessita receber dados de um cliente, o formato a ser recebido necessita ser em um destes formatos. Um exemplo de requisição HTTP com o verbo GET na URI ’http://localhost/class’ que lista todas as turmas, deverá retornar algo semelhante a Figura 1.

Figura 1 – Resposta JSON para a requisição de todas as turmas.


```
{
  "total": 1,
  "lastPage": 1,
  "prev": null,
  "next": null,
  "data": [
    {
      "id": "4ccec284-127e-4287-9689-c298a604ea8b",
      "name": "Programação 2",
      "academicYear": "2022.2",
      "period": "4º",
      "isRegularClass": true,
      "createdAt": "2022-06-03T02:09:32.925Z",
      "updatedAt": "2022-06-03T02:09:32.926Z",
      "disciplineId": "0d384811-273d-430b-96af-8e84dd17bd05",
      "academicCenterId": "2f6461d3-9ddc-487a-b4c1-74877be9ccaa",
      "professorId": "93f2cab7-66c1-4237-a3fa-1f74d7e6027c",
      "dateInitClass": "2022-02-01T00:00:00.000Z",
      "dateEndClass": "2022-07-31T00:00:00.000Z",
      "discipline": {
        "id": "0d384811-273d-430b-96af-8e84dd17bd05",
        "code": "CPTA054-B",
        "name": "Programação 2",
        "initials": "P2",
        "courseLoad": 40,
        "createdAt": "2022-05-01T22:55:27.099Z",
        "updatedAt": "2022-05-01T22:55:27.100Z",
        "academicCenterId": "2f6461d3-9ddc-487a-b4c1-74877be9ccaa"
      },
      "professor": {
        "id": "93f2cab7-66c1-4237-a3fa-1f74d7e6027c",
        "name": "Alexandre de Andrade Barbosa",
        "mail": "alexandre.barbosa@arapiraca.ufal.br",
        "roles": [
          "PROFESSOR"
        ],
        "registration": "8765432",
        "code": null,
        "caInitDate": null,
        "caEndDate": null,
        "createdAt": "2022-04-29T03:03:42.280Z",
        "updatedAt": "2022-04-29T03:03:42.283Z"
      }
    }
  ]
}
```


Fonte: Autor (2022).

3.2 MAPA DE CALOR

Exibir dados de forma visual é mais eficiente que a informação disposta em texto. De acordo com Thiel (2018), estudos recentes apontam que 90% das informações transmitidas ao cérebro são visuais e são processadas 60 mil vezes mais rápido que os textos.

Ainda mais quando se trata de um volume considerável de dados, onde as representações

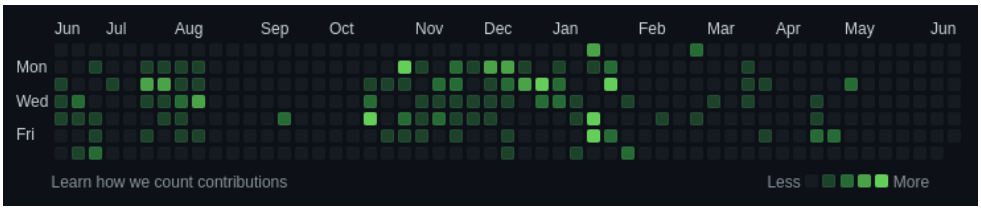
de dados visuais são mais convincentes do que uma informação textual com base em números porque o cérebro humano é capaz de processar imagens gráficas com mais facilidade do que uma tabela numérica, que requer um esforço adicional de leitura para realizar as correlações necessárias e obter o entendimento completo do que está sendo exibido (MURPHY, 2013).

Utilizando-se de técnicas de Visualização de Dados (VD) para se mapear dados em gráficos, se obtem informações mais simples de serem lidas, permitindo-se que as pessoas vejam e entendam com mais rapidez e facilidade. O propósito geral da visualização de dados não é somente exibir dados de uma forma inteligível para o ser humano, mas também é utilizado para comunicar reflexões com sucesso e uma maneira efetiva de comunicar informações (WILKE, 2019) acerca dos dados exibidos.

Dentre as técnicas e modelos de VD existentes, o mapa de calor é o que se demonstra um modelo viável a esse projeto. É um modelo que possibilita uma visão dos maiores e menores valores em uma matriz de dados, além de agrupar linhas e/ou colunas de valor semelhante (METSALU, 2016).

Um exemplo de mapa de calor pode ser encontrado nos sistemas de repositório de código de projetos de software como GitHub. O objetivo do mapa de calor nesses sistemas é exibir a frequêcia de contribuições de um usuário em projetos de software ao longo do tempo da plataforma. Na Figura 2 é apresentado um mapa de calor com a exibição da frequência de contribuições de um ano. Percebe-se que as cores vão ficando mais saturadas à medida que temos mais contribuições em um dia.

Figura 2 – Visualização de dados utilizando Mapa de Calor para exibição da frequência de contribuições de um ano



Fonte: www.github.com (2022).

4 O SOFTWARE ATENA

Neste capítulo será apresentado o desenvolvimento do software Atena, com seus processos, artefatos gerados e tecnologias utilizadas.
O processo de desenvolvimento se deu em três partes, que são:

- Levanamento e análise de requisitos do software
- Modelagem do sistema
- Codificação

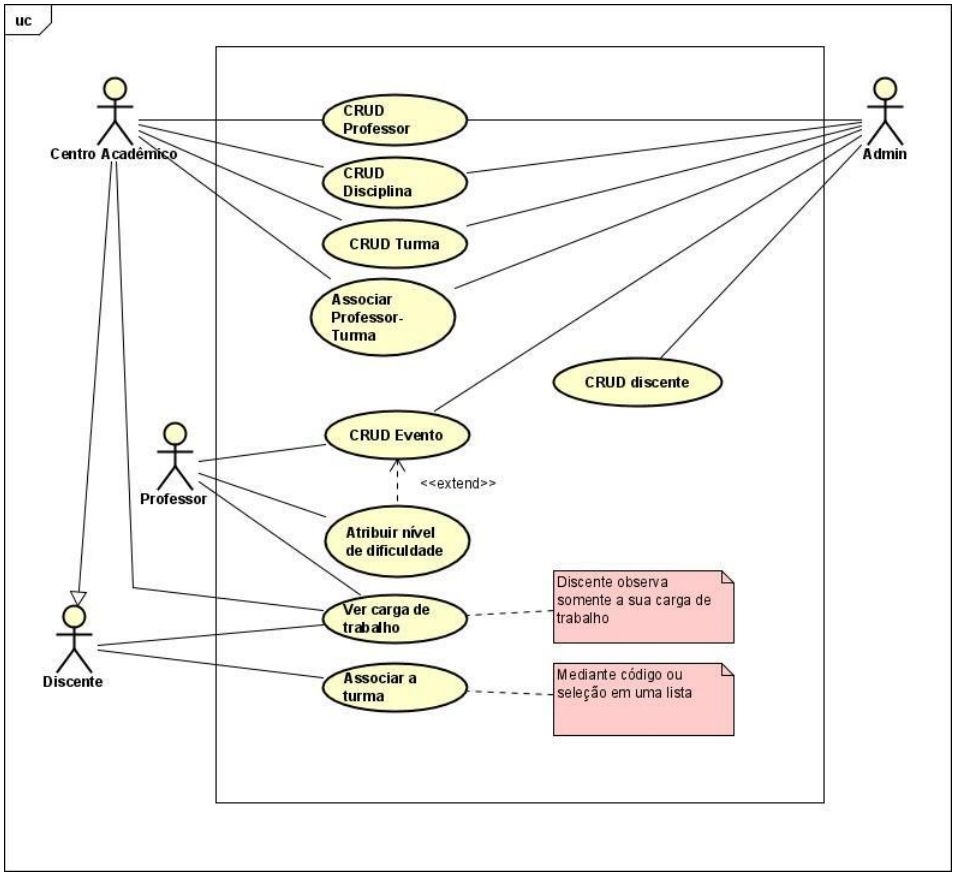
4.1 LEVANTAMENTO E ANÁLISE DE REQUISITOS DO SOFTWARE ATENA

O processo de análise e levantamento de requisitos se deu em reuniões entre as partes envolvidas: um professor, um aluno e um ex-representante do CA, com o objetivo de definir o escopo que a aplicação teria. Esta ferramenta servirá para discentes e docentes visualizarem os eventos aos quais estão associados e servirá de apoio para fundamentar as demandas entre os discentes e docentes.

Após diversos requisitos serem discutidos ao longo do processo, o escopo proposto neste trabalho é um MVP. A Figura 3 apresenta o Diagrama UML de Casos de Uso do sistema, gerado após o processo de levantamento de requisitos, ilustrando o escopo de atuação de cada ator dentro do sistema.

Os discentes que estarão como membros do CA, além do escopo de atuação de estudante, terão um escopo de atuação semelhante a um administrador do sistema, onde eles poderão gerenciar professores, turmas e disciplinas, bem como associar porfessor a uma determinada turma, uma determinada turma a uma disciplina e atender solicitações de criação de conta de professor. O professor, por sua vez, terá um escopo de atuação em que fica encarregado apenas de criar eventos para as turmas que leciona. Um discente poderá se associar a uma turma ao qual esteja matriculado e visualizar os eventos das turmas que se associou

Figura 3 – Diagrama UML de Casos de Uso do sistema.



Fonte: Autor (2022).

Sobre a Figura 3, segue uma descrição de cada requisito:

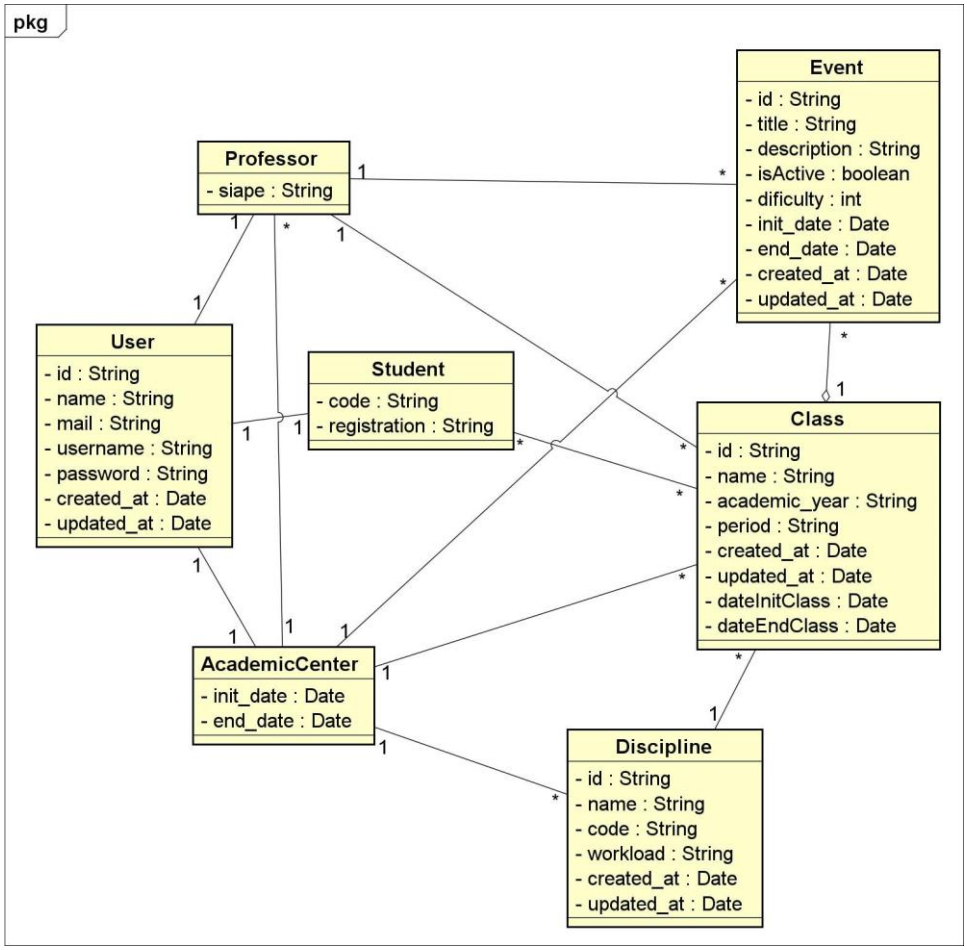
- **CRUD Professor**: Criar, Ler, Alterar e Deletar professores do sistema. Requisito de responsabilidade de usuários do tipo Centro Acadêmico e Admin;
- **CRUD Disciplina**: Criar, Ler, Alterar e Deletar disciplinas do sistema. Requisito de responsabilidade de usuários do tipo Centro Acadêmico e Admin;
- **CRUD Turma**: Criar, Ler, Alterar e Deletar disciplinas do sistema. Requisito de responsa- bilidade de usuários do tipo Centro Acadêmico e Admin;
- **Associar Professor-Turma**: Um docente é alocado para cada turma criada no sistema. Requisito de responsabilidade de usuários do tipo Centro Acadêmico e Admin;
- **CRUD Evento**: Criar, Ler, Alterar e Deletar eventos do sistema. Requisito de responsabili- dade de usuários do tipo Professor e Admin;
- **CRUD Discente**: Criar, Ler, Alterar e Deletar discentes do sistema. Requisito de responsa- bilidade de usuários do tipo Admin;

- **Atribuir nível de dificuldade**: Cada evento tem um nível de dificuldade que é um valor de inteiro de 1 à 5. Requisito de responsabilidade de usuários do tipo Professor;
- **Ver carga de trabalho**: Uma listagem de evntos próximos ou uma visualização de um mapa de calor entre datas que representa a carga de trabalho.
- **Associar a turma**: Um Discente se relacionar com uma turma, ou seja, Muitas turmas tem muitos discentes associados e o discentefica responsável por se associar as turmas de seu interesse.

4.2 MODELOS DO SISTEMA

Após o processo de análise de requisitos, o sistema foi modelado gerando como artefato o Diagrama UML de Classes. A Figura 4 apresenta o Diagrama UML de Classes, que ilustra as propriedades inerentes a cada entidade e como essas entidades se relacionam umas com as outras.

Figura 4 – Diagrama UML de Classes do sistema

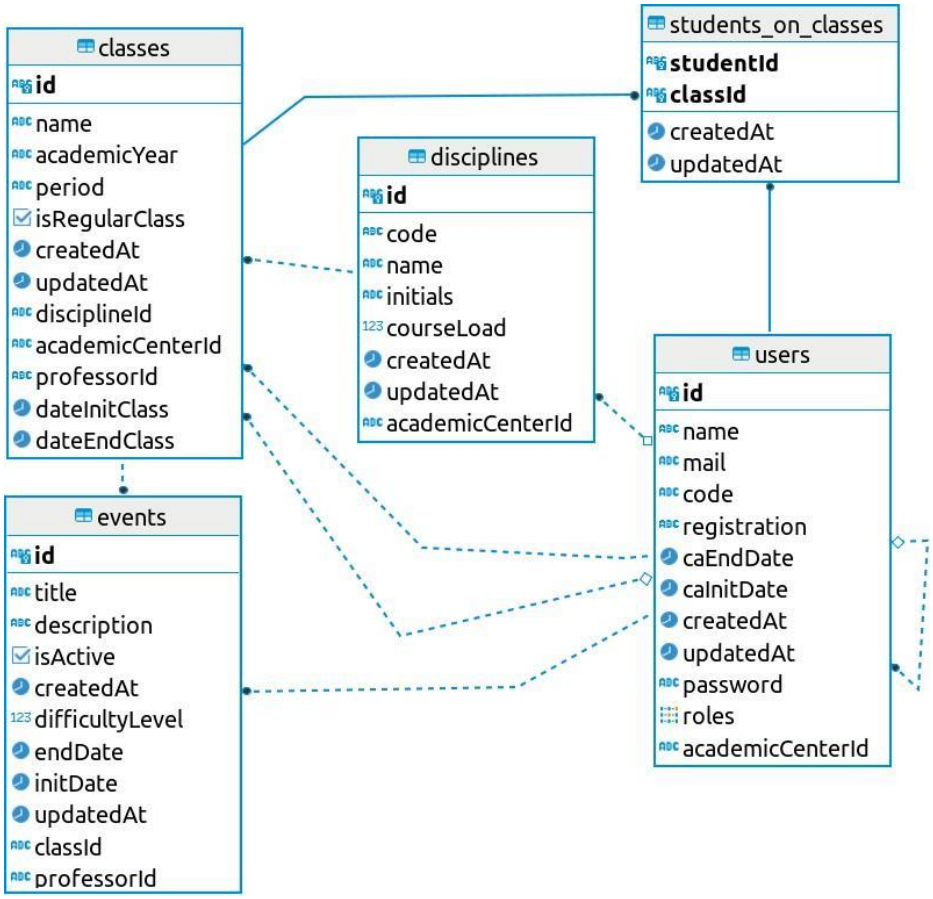


Fonte: Autor (2022).

Se usou relacionamentos 1-1 entre as classes User, Professor, Student e AcademicCenter; porque o User pode assumir tais responsabilidades no sistema. Após a observar-se que existe uma relação 1-1 entre User e Professor, User e Student e User e AcademicCenter; na criação do banco de dados se decidiu usar uma entidade única chamada User para agrupar os dados das demais e uma propriedade que é uma lista de enums com as resposabilidades que o User pode ter no sistema, sendo elas: Professor, Student, AcademicCenter e Admin. Tal decisão de projeto possibilitou se usar menos da cláusula SQL *JOIN* no projeto além de diminuir a quantidade de entidades do sistema e com isso se gerou um diagrama Entidade-Relacionamento ilustrado na Figura 5.

Sobre a Figura 4, a classe Event representa um evento cadastrado pelo professor no sistema, por isso vários eventos são associados a um objeto professor e um objeto professor é associado a vários eventos. A classe Class representa a turma ministrada por um professor e possui uma disciplina e um estudante do centro acadêmico que a criou no sistema. Como o estudante do centro acadêmico que cria no sistema professores, disciplinas e turmas, há uma associação um para muitos entre AcademicCenter e as demais entidades. Há um relacionamento N-N entre estudante e turmas, logo um estudante está em várias turmas e varias turmas tem vários estudantes.

Figura 5 – Diagrama Entidade-Relacionamento do Sitema, gerado pelo software Dbeaver.



Fonte: Autor (2022).

Na Figura 5, observa-se que classes da Figura 4 foram mapeadas em entidades que serão tabelas no banco de dados. Essas entidades são:

- **users**- É o mapeamento das classes User, Professor, Student e AcademicCenter.
- **events**- É o mapeamento da classe Event, guarda o id do professor que criou, o id da turma a qual o evento é relacionado e seus demais campos.
- **classes**- É o mapeamento da classe Class que representa a turma, guarda o id do professor responsável pela turma, do user que pertence ao centro acadêmico que criou a turma e o id da disciplina da turma, além dos demais campos inerentes a turma.
- **disciplines**- É o mapeamento da classe Disciplina, é a disciplina que corresponde a cada turma. Recebe um id do user que pertence ao centro acadêmico que criou a disciplina e seus demais campos.
- **students-on-classes**- É o mapeamento do relacionamento N-N entre Student e

Class, logo um estudante está em várias turmas e varias turmas tem vários estudantes.

4.3 CODIFICAÇÃO

Após a modelagem do sistema, foi estabelecido para este trabalho estas tecnologias para desenvolvimento da API:

- **NodeJS**- Interpretador JavaScript baseado na*Engine*V8 do Google, que permite executar código JavaScript fora do navegador web.
- **JavaScript**- Linguagem de programação de alto nível, dinâmica e não tipada, conveniente para estilos de programação orientados a objetos e funcionais (FLANAGAN, 2004). Voltada para escrita de sistemas web, foi desenvolvida para rodar do lado cliente, isto é, no navegador web; mas o NodeJs permite fazer uso fora do navegador web.
- **TypeScript**- Um*superset*para a linguagem JavaScript, que permite adicionar tipagem estática.
- **ExpressJs**- Web *framework* que atua como uma camada no topo do Node.js, facilitando e deixando o desenvolvimento de APIs em node mais prático. Além disso, com ele é mais fácil organizar as funcionalidades da aplicação usando middleware e roteamento, facilita a renderização de páginas HTML dinâmicas. Ele define um padrão de extensibilidade facilmente implementado e adiciona utilitários para os objetos HTTP do Node.js (HAHN, 2016).
- **PostgreSQL**- SBGD para bancos de dados relacionais.
- **Prima**- ORM criado para ser utilizado em aplicações*backend*escritas em TypeScript e que funciona com quaisquer tipos de bancos de dados.
- **Docker**- Plataforma que facilita a criação e administração de ambientes isolados.
- **Dart**- Linguagem de programação criada pela Google utilizada pelo Flutter.
- **Flutter**- Na documentação oficial da ferramenta dá-se a definição (em tradução direta): “Flutter é o kit de ferramentas de IU portátil do Google para criar aplicativos bonitos e compilados de forma nativa para dispositivos móveis, web e desktop a partir de uma única base de código”. Utilizado nesse projeto para criar interfaces para exemplificar o uso de alguns dos *endpoints* da API.

A visualização da carga de trabalho é um recurso visual que permite ao usuário ver um paronama de como os eventos vinculados a um usuário estão distribuídos ao longo do semestre. Assim, o usuário pode observar as possíveis frequências de alta concentração de eventos ao

longo do semestre, o que possibilita uma melhora no gerenciamento de tempo do usuário e uma melhor negociação de aplicações de eventos em determinadas datas.

Para ser possível construir a visualização da carga de trabalho, se foi criado duas requisições HTTP com o verbo GET na API, uma para os professores e turmas e uma para a visualização do aluno. Tais requisições trazem como resposta principal a lista de eventos em um certo período determinado definido pelo usuário da API, cabe ao *frontend* usar esses dados e montar uma visualização da carga de trabalho do período determinado. As requisições são:

- **http://URL-API/event/workload/general/**- recebe como parâmetro obrigatório uma data de início e como parâmetro opicional uma data de fim. Recebe também dois parâmetros de busca, o id do professor e o id da turma.

A Figura 6 mostra a requisição GET http://URL-API/event/workload/general/2022-06-01/2022-07-13?professorId=1856d813-08a6-4444-942f-1e9a0ac9f249; o objetivo é trazer todos os eventos das turmas lecionadas pelo professor, cujo id é 1856d813-08a6-4444- 942f-1e9a0ac9f249, entre as datas 01/06/2022 e 13/07/2022.

Figura 6 – Exemplo de requisição GET http://URL-API/event/workload/general/.


```
{
  "timePeriodInit": "2022-06-01T00:00:00.000Z",
  "timePeriodEnd": "2022-07-13T00:00:00.000Z",
  "classId": "",
  "professorId": "1856d813-08a6-4444-942f-1e9a0ac9f249",
  "total": 1,
  "events": [
    {
      "id": "f4bbe50d-4ab8-4f94-be3e-dc1c70fad936",
      "title": "Atividade 1",
      "description": "Atividade 1, envolvendo primeiros conceitos de Programação e Python",
      "isActive": true,
      "createdAt": "2022-07-06T22:50:22.801Z",
      "updatedAt": "2022-07-06T22:50:22.802Z",
      "difficultyLevel": 3,
      "endDate": "2022-07-12T00:00:00.000Z",
      "initDate": "2022-07-07T00:00:00.000Z",
      "classId": "a357687b-b7a3-437d-b88d-1cd6e0b85a50",
      "professorId": "1856d813-08a6-4444-942f-1e9a0ac9f249",
      "id_class": "a357687b-b7a3-437d-b88d-1cd6e0b85a50",
      "name_class": "Programação 1",
      "academic_year_class": "2022.1",
      "period_class": "3º",
      "disciplineId": "69b83216-c05d-4174-90d7-41a2276c258a",
      "name_professor": "Alexandre de Andrade Barbosa",
      "mail_professor": "alexandre.barbosa@arapiraca.ufal.br",
      "registration_professor": "2345678"
    }
  ]
}
```


Fonte: Autor (2022).

- **http://URL-API/event/workload/student/**- recebe como parâmetros obrigatórios a data de início, a data de fim e o id do estudante.

A Figura 7 mostra a requisição GET http://URL-API/event/workload/student/2022-06-01/2022-07-13/10db6aee-5993-4405-aca9-6ec59b32eacf; o objetivo é trazer todos os eventos das turmas que o estudante está associado, o estudante tem o id 10db6aee-5993- 4405-aca9-6ec59b32eac, e estes eventos estão entre as datas 01/06/2022 e 13/07/2022.

Figura 7 – Exemplo de requisição GET http://URL-API/event/workload/student/


```
{
  "timePeriodInit": "2022-06-01T00:00:00.000Z",
  "timePeriodEnd": "2022-07-13T00:00:00.000Z",
  "total": 1,
  "student": {
    "id": "10db6aee-5993-4405-aca9-6ec59b32eacf",
    "name": "Antonio Andrade Gomes Júnior",
    "mail": "gomesmax1997@gmail.com",
    "roles": [
      "ACADEMIC_CENTER",
      "STUDENT"
    ],
    "registration": "15113247",
    "code": null,
    "caInitDate": "2022-07-06T22:34:44.183Z",
    "caEndDate": "2022-12-06T22:27:34.357Z",
    "createdAt": "2022-07-06T22:23:40.596Z",
    "updatedAt": "2022-07-06T22:34:44.225Z",
    "academicCenterId": null
  },
  "events": [
    {
      "id": "f4bbe50d-4ab8-4f94-be3e-dc1c70fad936",
      "title": "Atividade 1",
      "description": "Atividade 1, envolvendo primeiros conceitos de Programação e Python",
      "isActive": true,
      "createdAt": "2022-07-06T22:50:22.801Z",
      "updatedAt": "2022-07-06T22:50:22.802Z",
      "difficultyLevel": 3,
      "endDate": "2022-07-12T00:00:00.000Z",
      "initDate": "2022-07-07T00:00:00.000Z",
      "classId": "a357687b-b7a3-437d-b88d-1cd6e0b85a50",
      "professorId": "1856d813-08a6-4444-942f-1e9a0ac9f249",
      "id_class": "a357687b-b7a3-437d-b88d-1cd6e0b85a50",
      "name_class": "Programação 1",
      "academic_year_class": "2022.1",
      "period_class": "3º",
      "disciplineId": "69b83216-c05d-4174-90d7-41a2276c258a"
    }
  ]
}
```


Fonte: Autor (2022).

Com as requisições descritas acima é possível se montar algumas visualizações de carga de trabalho. Como exemplo a Figura 8, que mostra a carga de trabalho de uma turma, e nela se vê um calendário com bolinhas coloridas em que as bolinhas verdes indicam uma baixa quantidade de eventos para serem feitos naquele dia, as amarelas indica um estágio intermediário, as vermelhas indicam uma alta quantidade de eventos para serem realizados pela turma naquele dia, e onde não tem a presença de bolinha indica que não há eventos a serem

realizados pela turma naquele dia.

Figura 8 – Visualização da carga de trabalho de uma turma chamada Programação 1.

Fonte: Autor (2022).

Na Figura 8, o mapa de calor demonstrado considera a data de início e de fim dos

eventos. Sendo assim se um evento inicia no dia 01 de agosto de 2022, e finaliza no dia 03 de agosto de 2022 ele é contabilizado em uma unidade para cada dia no intervalo, considerando todo o período em que a atividade esteve aberta para entrega. Porém o mapa de calor pode ser constuído considerando somente a data de finalização ou início e finalização.

5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi apresentada uma ferramenta voltada para prover recursos de visualização de dados de docentes, discentes, turmas e disciplinas; com o objetivo de mostrar uma visualização da carga de trabalho de docentes, discentes e disciplinas. A separação dos deveres do *frontend* e do *backend* possibilita a disponibilidade de dados para várias aplicações futuras a serem desenvolvidas usando essa ferramenta. Foi demonstrado o processo de levantamento de requisitos, modelagem e tecnologias utilizadas na codificação do sistema.

A ferramenta em seu estado atual é capaz de realizar todos os requisitos demonstrados no capítulo 4, de modo a apresentar ao*frontend*duas requisições com as quais se posibilita criar um mapa de calor demonstrando carga de trabalho.

Os proximos passos para evolução da ferramenta dizem respeito a uma melhor representação e cálculo da carga de trabalho de docentes, discentes e turmas. Possibilitando uma melhor visualização no*frontend*de um mapa de calor.

Um outro trablho que poderá ser desenvolvido é uma aplicação *frontend* web e *mobile* que utilize dados dessa API, inclusive uma aplicação para o Admin. O uso do banco de dados gerado nessa aplicação pode ser usado para estudos em áreas de inteligência artificial para geração de cálculos mais precisos de carga de trabalho.

Por fim, destaca-se que esta API pode ser expandida para uso em aplicações como o controle do RU, ou o uso na biblioteca da UFAL. Diferentes aplicações e projetos podem ser desenvolvidos, incluindo, mas não se limitando a ferramentas de grenciemento de eventos acadêmicos e visualização de carga de trabalho para tomadas de decisão.

REFERÊNCIAS

DIAS, A. C.; SANTOS, F. Á. d.; MOTTA, T. S. Entregando recursos para aplicações multiplataforma com a api ufrgs. *In:* WORKSHOP DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO DAS IFES. 13., 2019. **Anais [...]** Cuiabá: Andifes, 2019.

FERREIRA, A. S.; NICACIO, J. M.; FERREIRA, G. V.; FILHO, G. M. S.; RODRIGUES, V. S. Desenvolvimento de uma api rest para um sistema acadêmico de terceiros. **Revista de Sistemas e Computação-RSC**, v. 8, n. 2, 2018. Disponível em: https://revistas.unifacs.br/index.php/rsc/article/view/5862. Acesso em: 15 jun. 2022.

FLANAGAN, D.**JavaScript: o guia definitivo**. [S.l.]: Bookman Editora, 2004.

HAHN, E.**Express in Action:** writing, building, and testing node. js applications. [S.l.]: Simon and Schuster, 2016.

METSALU, T.**Statistical analysis of multivariate data in bioinformatics**. [S.l.]: Tartu, 2016.

MURPHY, S. A. Data visualization and rapid analytics: Applying tableau desktop to support library decision-making.**Journal of Web Librarianship**, Taylor & Francis, v. 7, n. 4, p. 465–476, 2013.

PURCELL, C.**Análise de uma Ferramenta de Gestão para o Desenvolvimento da Atividade Docente: estudo de caso em uma Instituição privada de Ensino Superior em Maceió/Brasil**. Dissertação (Mestrado em Gestão do Potencial Humano) - Instituto Superior de Gestão, Lisboa, 2020. Disponível em: https://comum.rcaap.pt/handle/10400.26/31688?locale=en. Acesso em 17 abr, 2022.

THIEL, C.**Marketing Visual: Qual a Importância das Imagens?** 2018. Disponível em: https://cristianethiel.com.br/2018/03/13/marketing-visual-qual-a-importancia-das-imagens/. Acesso em: 15 jun. 2022.

WILKE, C. O.**Fundamentals of data visualization**: a primer on making informative and compelling figures. [S.l.]: O'Reilly Media, 2019.