UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS
CURSO DE BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

LUAN GUILHERME DANTAS

UM PROTÓTIPO DE UM SISTEMA PARA FORNECER DICAS PARA TAREFAS DE PROGRAMAÇÃO EM DISCIPLINAS DE PROGRAMAÇÃO INTRODUTÓRIA

PAU DOS FERROS - RN
2020

LUAN GUILHERME DANTAS

UM PROTÓTIPO DE UM SISTEMA PARA FORNECER DICAS PARA TAREFAS DE PROGRAMAÇÃO EM DISCIPLINAS DE PROGRAMAÇÃO INTRODUTÓRIA

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Tecnologia da Informação.

Orientador: Reudismam Rolim de Sousa, Prof. Dr.

PAU DOS FERROS - RN

2020

D192p Dantas, Luan Guilherme.
Um protótipo de um sistema para fornecer dicas para tarefas de programação em disciplinas de programação introdutória / Luan Guilherme Dantas. - 2020.
62 f. : il.

Orientador: Reudismam Rolim de Sousa.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2020.

1. Programação. 2. Aprendizagem de programação. 3. Projeto de sistema web. 4. Sistema de dicas. I. Sousa, Reudismam Rolim de , orient. II. Título.

LUAN GUILHERME DANTAS

UM PROTÓTIPO DE UM SISTEMA PARA FORNECER DICAS PARA TAREFAS DE PROGRAMAÇÃO EM DISCIPLINAS DE PROGRAMAÇÃO INTRODUTÓRIA

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Tecnologia da informação.

Defendida em: 07 / 12 / 2020.

BANCA EXAMINADORA

REUDISMAM ROLIM DE SOUSA:08392843495
Assinado de forma digital por REUDISMAM ROLIM DE SOUSA:08392843495
Dados: 2020.12.08 16:47:15 -03'00'

Reudismam Rolim de Sousa, Prof. Dr. (UFERSA)

Presidente

LAYSA MABEL DE OLIVEIRA FONTES:07092274427
Assinado de forma digital por LAYSA MABEL DE OLIVEIRA FONTES:07092274427
Dados: 2020.12.09 07:52:49 -03'00'

Laysa Mabel de Oliveira Fontes, Profa. Dra. (UFERSA)

Membro Examinador

VERONICA MARIA LIMA SILVA:02558212397
Assinado de forma digital por VERONICA MARIA LIMA SILVA:02558212397
Dados: 2020.12.09 12:09:03 -03'00'

Verônica Maria Lima Silva, Profa. Dra. (UFERSA)

Membro Examinador

Dedico esse trabalho aos meus pais, Jacinta e Inácio, e a meus irmãos Lucas e Leonardo. Também dedico a todos os meus amigos que contribuem em minha jornada.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me proporcionar chegar até aqui; me abençoou trazendo energia e força de vontade fazendo com que me mantenha firme para superar todos os desafios durante esse ciclo.

A minha família, principalmente, aos meus pais, minha mãe Jacinta Guilherme e meu pai Inácio Dantas, aos meus irmãos Lucas Guilherme e Leonardo Inácio, e a minha namorada Madleide Dantas por todo o apoio, incentivos, pela paciência e compreensão pelas ausências nas muitas horas dedicada ao estudo e a este trabalho.

Agradeço a todos meus amigos que se tornaram presentes em minha vida, em especial: Maria Adriana, Felipe Cardoso, Elizieb Luiz, Rosendo Lucas, Sebastião Costa, Igor Sthaynny, Allan Miqueias; que me ajudaram muito nessa jornada. Agradeço ao grupo de pesquisa GPSiCS por serem tão receptivos, são como uma família para mim; a todos os membros, em especial, ao coordenado do grupo Professor Dr. Lenardo Chaves e ao Professor Dr. Reudismam Rolim, por serem grandes exemplos de pessoas e profissionais.

A UFERSA, na qual me trouxe oportunidades que me fizeram obter um amadurecimento pessoal e profissional; aos meus professores pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho durante minha jornada, sendo grande fonte de inspiração.

Meu muito obrigado para meu orientador Reudismam Rolim, que me fez acreditar e me permitiu finalizar esse trabalho, oferecendo dedicação a mim e todo seu conhecimento e inteligência, sendo grande fonte de inspiração e exemplo de profissional.

Agradeço a todos esses citados, mas também agradeço aqueles que de alguma forma contribuíram para minha jornada acadêmica.

“Temos de fazer o melhor que podemos. Esta é a nossa sagrada responsabilidade humana.”

(Albert Einstein)

RESUMO

Dificuldades com a resolução de problemas por meio de uma linguagem de programação é um desafio comum entre os cursos que envolvem programação de computadores. Essa característica pode ser explicada por alguns fatores, tais como a dificuldade da abordagem desses tópicos no ensino básico, sendo em escolas públicas ou privadas. Esses problemas causam várias dificuldades para esses cursos, dentre essas o aumento da evasão, retenção e a necessidade constante de alocar novos docentes para ministrar essas componentes. Neste sentido, várias abordagens foram propostas para melhorar as taxas de sucessos, dentre elas a abordagem discutida em Freitas (2020) para fornecer dicas de programação para auxiliar os estudantes na resolução de problemas. No entanto, essa abordagem atualmente não dispõe de um sistema para auxiliar os estudantes no consumo de dicas e para os professores melhor intervirem baseado nesses dados. Dessa forma, este trabalho buscou desenvolver, utilizando a metodologia de desenvolvimento de projeto (*design*) de interfaces guiada pelo usuário proposta Garrett (2011) e a ferramenta Figma, um protótipo de um sistema para automatizar o sistema de dicas avaliado em Freitas (2020), com objetivo de tornar este processo de intervenção mais eficaz e eficiente. Como resultado, espera-se que o projeto e protótipo desenvolvido possa guiar o desenvolvimento e implementação desse sistema que informatizará a intervenção metodológica proposta por Freitas (2020).

Palavras-chave: Programação. Aprendizagem de programação. Projeto de sistema web. Sistemas de dicas.

ABSTRACT

Difficulties with solving problems using a programming language is a common challenge among courses that involve computer programming. This characteristic can be explained by some factors, such as the difficulty of addressing these topics in basic education, both in public or private schools. These problems cause several difficulties for these courses, among them the increase in dropout, retention, and the constant need to allocate new teachers to these components. In this sense, several approaches have been proposed to improve success rates, among them the approach discussed in Freitas (2020) to provide programming tips to assist students in problem-solving. However, this approach currently does not have a system to assist students in consuming tips and for teachers to better intervene based on these data. Thus, this work aims to develop, using the development methodology (design) of the project of interfaces guided by users proposed by Garrett (2011) and the Figma tool, a prototype of a system to automate the system of tips evaluated in Freitas (2020), in order to make this intervention process more effective and efficient. As a result, it is expected that the project and prototype developed can guide the development and implementation of this system that will automate the methodological intervention proposed by Freitas (2020).

Keywords: *Programming. Programming learning. Web system design. Tip systems.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
AVAs	Ambientes Virtuais de Aprendizagem
BTI	Bacharelado em Tecnologia da Informação
CSU	Casos de Uso
EaD	Educação a Distância
ICPC	*International Collegiate Programming Contest*
PDF	Pau dos Ferros
TCC	Trabalho de Conclusão de Curso
UFERSA	Universidade Federal Rural do Semi-Árido
UML	*Unified Modeling Language*
URI	Universidade Regional Integrada

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta uma visão geral deste trabalho e está organizado da seguinte forma: a Seção 1.1 apresenta uma contextualização; a Seção 1.2 apresenta a problematização; a Seção 1.3 apresenta a justificativa; a Seção 1.4 apresenta os objetivos deste trabalho; e, por fim, a Seção 1.5 apresenta como está organizado este trabalho.

1.1 CONTEXTUALIZAÇÃO

Os discentes do ensino superior na área de computação costumam apresentar dificuldades em disciplinas voltadas à programação, principalmente, nos períodos iniciais dos cursos no Brasil (ROLIM 2020; SOUSA et al., 2020). Esta dificuldade pode estar relacionada ao baixo ou contato inexistente dos estudantes com temas dessa área (ROLIM 2020; SOUSA et al., 2020; QUEIROZ et al., 2018; MOREIRA et al., 2018).

Especialmente, o componente curricular de introdução à programação básica, Algoritmos, do curso de Bacharelado em Tecnologia da Informação (BTI) da Universidade Federal Rural do Semi-Árido do Centro Multidisciplinar de Pau dos Ferros (UFERSA-PDF), tende a apresentar elevados graus de insucessos, chegando a 87% no semestre 2016.2 e 70% no semestre 2017.1 (SOUSA et al. 2020).

Neste sentido, se faz necessário buscar alternativas para reduzir esses graus de insucessos (ROLIM 2020; SOUSA et al., 2020). Uma das formas que pode motivar os estudantes para o aprendizado de programação é o acompanhamento constante para tirar pequenas dúvidas que ocorram durante o momento de aprendizagem, desde um erro de sintaxe da linguagem ao entendimento do enunciado da questão. No entanto, fornecer esse *feedback* de forma imediata pode se tornar inviável devido a necessidade de uma comunicação constante com os mediadores do aprendizado, sejam eles o próprio docente ou monitores da disciplina. Atualmente, Algoritmos apresenta um número elevado de discentes, incluindo os ingressantes no curso, ao todo o curso de BTI recebe 80 alunos por semestre (UFERSA, 2014), somado aos discentes retidos. No ensino remoto ou à distância essas dificuldades são ainda maiores devido à distância física entre o docente e o aluno.

Com o intuito de colaborar com esta problemática, neste trabalho é proposto um protótipo de um sistema de dicas para aprendizagem de Algoritmos. A proposta do sistema a ser projetado permite aos discentes receber dicas personalizadas, voltadas para o programa em

discussão em sala de aula, que podem auxiliá-los a superar as suas dificuldades durante a resolução de problemas sugeridos pelo docente. Esta proposta busca informatizar a intervenção metodológica baseada em dicas de programação proposta por Freitas (2020). Ela consiste em aplicar, como autoavaliações, problemas de programação que referenciam conteúdos ministrados na disciplina de Algoritmos nas turmas, denominados como cenários, definir dicas para auxiliar os alunos no entendimento e na resolução de tais problemas e apresentar os resultados obtidos da resolução dos problemas por cada aluno aos docentes com o objetivo de apontar as dificuldades identificadas da turma ou específicas de cada aluno.

1.2 PROBLEMATIZAÇÃO

Existem diversos fatores que podem influenciar nas dificuldades encontradas pelos alunos no aprendizado de programação introdutória que, por sua vez, contribuem para os insucessos nos componentes curriculares que envolvem programação. Como discutido por Freitas (2020) e Rodrigues (2002), dentre esses fatores de dificuldade, inclui-se à dificuldade do desenvolvimento do raciocínio lógico. Pereira Junior et al. (2005) elencam fatores como dificuldades dos alunos em entender o problema a ser resolvido, em identificar os elementos necessários para solução, em aplicar os conhecimentos obtidos durante as aulas, entre outros fatores que elevam as taxas de insucessos nessas componentes. Outro fator que causa dificuldade é que esses componentes curriculares são o primeiro contato dos alunos com a área de programação, como afirmam Soares e Carvalho (2017).

As dificuldades com disciplinas voltadas à programação podem levar a vários problemas para as instituições de ensino superior. Uma delas é a retenção dos discentes nos períodos iniciais do curso, o que pode levar à necessidade de alocação de novos docentes para atender as demandas crescentes dos discentes nestas componentes.

Adicionalmente, as elevadas taxas de insucessos podem levar os discentes a se evadirem dos cursos, o que contribui para uma visão negativa dos discentes para cursos com componentes voltados para a programação de computadores. Além disso, o curso em que a disciplina se encontra alocada é um bacharelado interdisciplinar de primeiro ciclo. Nesse sentido, o discente ao completar o curso pode optar por realizar um curso de segundo ciclo no tema de seu interesse. A UFERSA-PDF dispõe de dois cursos de segundo ciclo, são eles Engenharia de Computação e Engenharia de *Software* (UFERSA, 2014). Dessa forma, a retenção no primeiro ciclo reduz o

quantitativo de ingressantes nos ciclos posteriores, o que pode comprometer a manutenção desses cursos.

Além disso, como mostrado por Freitas (2020), e evidenciado por Raabe e Silva (2005), esses problemas se intensificam devido a lentidão em se identificar as dificuldades de aprendizagem, bem como em definir estratégias ou métodos para melhorar o ensino de programação introdutória.

Geralmente, muitos dos métodos e estratégias para auxiliar a aprendizagem de programação são feitos de forma manual. Assim, todos os passos são feitos manualmente, desde a fase de aplicação até a analise dos resultados, em alguns casos, utilizando formulários *online*, que podem não se adequar a metodologia em uso, como é caso da intervenção metodológica aplicada por Freitas (2020) nas disciplinas de programação introdutória da UFERSA, *Campus* Pau dos Ferros. Nesse sentido, emerge o principal objetivo desse trabalho, que é desenvolver o projeto de um sistema para informatizar a intervenção metodológica proposta por Freitas (2020), de forma a torná-la mais eficiente.

1.3 JUSTIFICATIVA

A capacidade de auxiliar os estudantes na resolução de tarefas de programação no momento exato em que o problema ocorre pode ajudar os estudantes a desenvolver a motivação pelo aprendizado de programação, pois muitos estudantes se frustram por ficarem presos e não conseguirem desenvolver uma solução para determinado problema (QUEIROZ et al., 2018; MOREIRA et al., 2018). Esse auxílio pode ajudar os alunos com problemas que, muitas das vezes, são facilmente contornados, mas por questão de interpretação do enunciado ou não lembrar de determinada sintaxe de programação, eles não conseguem resolver; ao obter essa ajuda, os estudantes poderão ver a sua evolução, não ficando presos à resolução de uma questão e poderão direcionar o seu tempo para questões que possam contribuir para o aprendizado do assunto abordado.

Da mesma forma, o docente não fica restrito a problemas menores e podem focar o ensino em direcionamento que farão o estudante adquirir as competências necessárias objetivadas pelo componente curricular ou mesmo superá-los.

Além disso, a redução do número de insucessos nessas componentes iniciais permite que os cursos organizem melhor o quadro de docentes para atender mais adequadamente às demandas do curso nesta área.

1.4 OBJETIVOS

Neste capítulo apresenta-se o objetivo geral e específicos que foram desenvolvidos neste trabalho de conclusão de curso.

1.4.1 Objetivo Geral

O objetivo geral do trabalho é o desenvolvimento de um protótipo de um sistema, com o propósito de auxiliar estudantes de programação, fornecendo *feedback* imediato (dicas) sobre a interpretação do problema, sintaxe e lógica de programação, na realização de tarefas no contexto de programação introdutória.

1.4.2 Objetivos Específicos

Para alcançar o objetivo geral destacado, os seguintes objetivos específicos foram realizados:

- Revisão da literatura para entender os principais problemas relacionados à resolução de problemas de programação;
- Levantamento de requisitos do sistema;
- Projeto de um sistema para fornecer dicas para problemas de programação;
- Prototipagem do sistema.

1.5 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado da seguinte forma: no Capítulo 2 é apresentada a fundamentação teórica utilizada para o desenvolvimento deste trabalho; no Capítulo 3 é apresentada a metodologia utilizada para prototipagem e desenvolvimento do projeto do sistema; no Capítulo 4 são apresentados os trabalhos relacionados; e, por fim, no Capítulo 5 é apresentado e descrito o resultado final do projeto (*design*) de interface.

2 REFERENCIAL TEÓRICO

Neste capítulo, apresenta-se alguns aspectos relacionados ao tema da proposta e a base teórica que fundamenta este trabalho, de forma a auxiliar a uma melhor compreensão do sistema proposto, estando organizado da seguinte forma: na Seção 2.1 é discutido o tema algoritmos; na Seção 2.2 são apresentados os ambientes ou sistemas de apoio ao ensino de algoritmos; e na Seção 2.3 é apresentado o trabalho tomado com base para o protótipo proposto.

2.1 ALGORITMOS

Como é definido por Dicio (2020) em informática algoritmos são “um conjunto de regras que fornecem uma sequência de operações capazes de resolver um problema específico”, ou seja, uma série de instruções, um código de tamanho finito que é lido de forma sistemática para executar uma tarefa ou resolver um problema.

Para Cormen et al. (2002), um algoritmo é qualquer procedimento computacional eficaz que para determinado valor ou conjunto de valores como entrada produz algum valor ou conjunto de valores como saída. Cormen et al. (2002) ainda completam que um algoritmo pode ser visto como uma ferramenta para resolver um problema computacional bem definido.

No ambiente acadêmico de um curso superior na área de computação, os alunos necessitam utilizar os algoritmos para resolver problemas. Muito das vezes, é a primeira vez do aluno no ambiente de programação, principalmente, os alunos que tiveram uma formação de educação básica que não abordavam o tema (ROLIM 2020; SOUSA et al., 2020).

2.2 AMBIENTES OU SISTEMAS DE APOIO AO ENSINO DE ALGORITMOS DE PROGRAMAÇÃO

Com o avanço da tecnologia e, principalmente, da internet, assim como da sociedade, novas formas de aprender e ensinar determinado assunto, seja ele qual for, foram criadas. Principalmente, no mundo virtual e digital, sendo o caso dos Ambientes Virtuais de Aprendizagem (AVAs). Hoje os AVAs apoiam diversas disciplinas seja ela matemática, geografia, ciência ou línguas estrangeiras.

Para Almeida (2003), os AVAs são sistemas ou ambientes virtuais disponíveis na internet utilizados para dar suporte as atividades de ensino/aprendizagem a distância, para apoio

às atividades de sala de aula, permitindo expandir as interações da aula além daquele momento presencial e possibilitando integrar múltiplos recursos, mídias e linguagens, expor informações de maneira organizada, elaborar e compartilhar conteúdos, pretendendo alcançar determinados objetivos. Dessa forma, seu objetivo é ajudar os professores no gerenciamento de conteúdos para seus alunos, além de possibilitar acompanhar o desenvolvimento do processo de ensino/aprendizagem dos estudantes.

Como discutido por Heming (2018), a aprendizagem mediante um AVA promove uma interatividade entre os envolvidos no ensino e aprendizado, como também proporciona facilidades em diversas tarefas como disponibilização de materiais (matérias teóricos, exercícios, entre outros arquivos), em acompanhar melhor o desempenho dos alunos, entre outras tarefas por meio da ferramenta. Com isso, o conhecimento é obtido de uma maneira eficaz, devido aos alunos tenderem a ser também protagonistas do processo de aprendizagem.

A utilização destas ferramentas trouxe à EaD (Educação a Distância), além de autonomia e a construção coletiva do conhecimento, também a possibilidade dos alunos, professores e tutores participarem ativamente (PROEC UFABC, 2014).

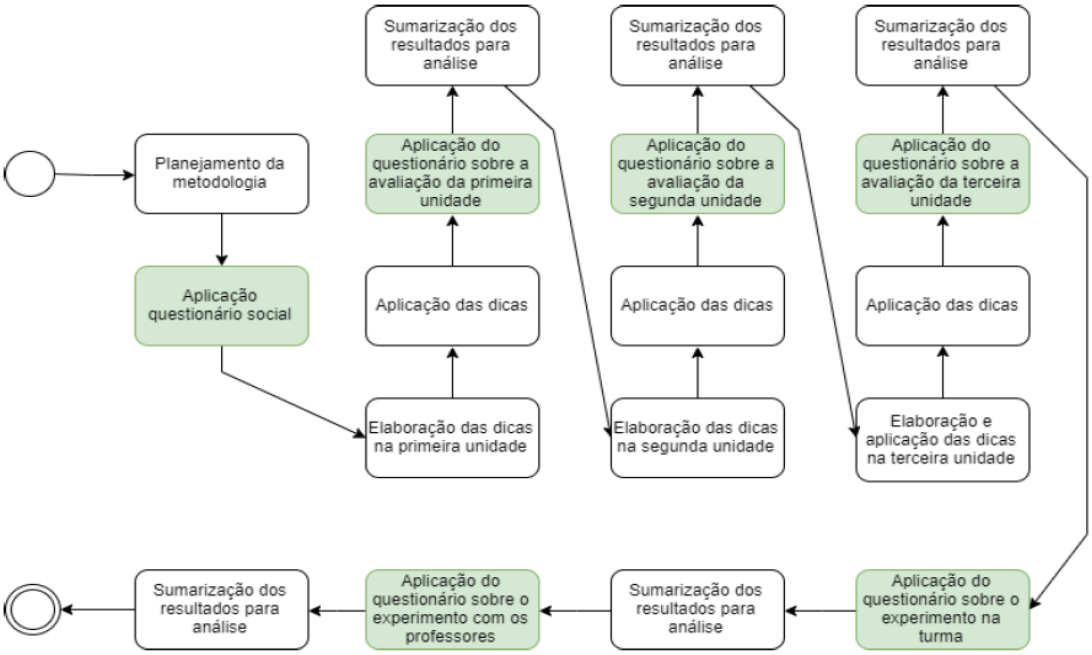
Dessa forma, muitos sistemas ou ambientes para o apoio ao ensino no geral e de programação de computadores foram e são criados e desenvolvidos. Pode-se citar diversos sistemas web para auxiliar o ensino/aprendizagem de programação de computadores, como, por exemplo, Code School, Khan Academy, Codecademy, URI Online Judge, entre outros ambientes.

2.3 INTERVENÇÃO METODOLÓGICA PARA AUXILIAR A APRENDIZAGEM DE PROGRAMAÇÃO

Freitas (2020), visando apoiar e expandir a aprendizagem de programação nas disciplinas de programação introdutória do curso de BTI da UFERSA-PDF, em seu trabalho de conclusão de curso, aplicou uma intervenção metodológica em turmas de tais disciplinas. O modelo da intervenção empreendido e aplicado é uma adaptação do modelo proposto por Holanda, Coutinho e Fontes (2018), que com as modificações tem como base principal conhecer os perfis e as características das turmas, definir exercícios, problemas de programação, denominados de cenário, que demanda aplicação do conhecimento dos conteúdos lecionados nas disciplinas, e dicas para ajudar os alunos no entendimento de tais problemas,

ficando a critério deles usarem ou não as dicas. Para aplicação dessa metodologia, Freitas (2020), a partir da Figura 1, apresenta os processos necessários.

Figura 1 - Fluxo de atividades da intervenção metodológica



Fonte: Freitas (2020)

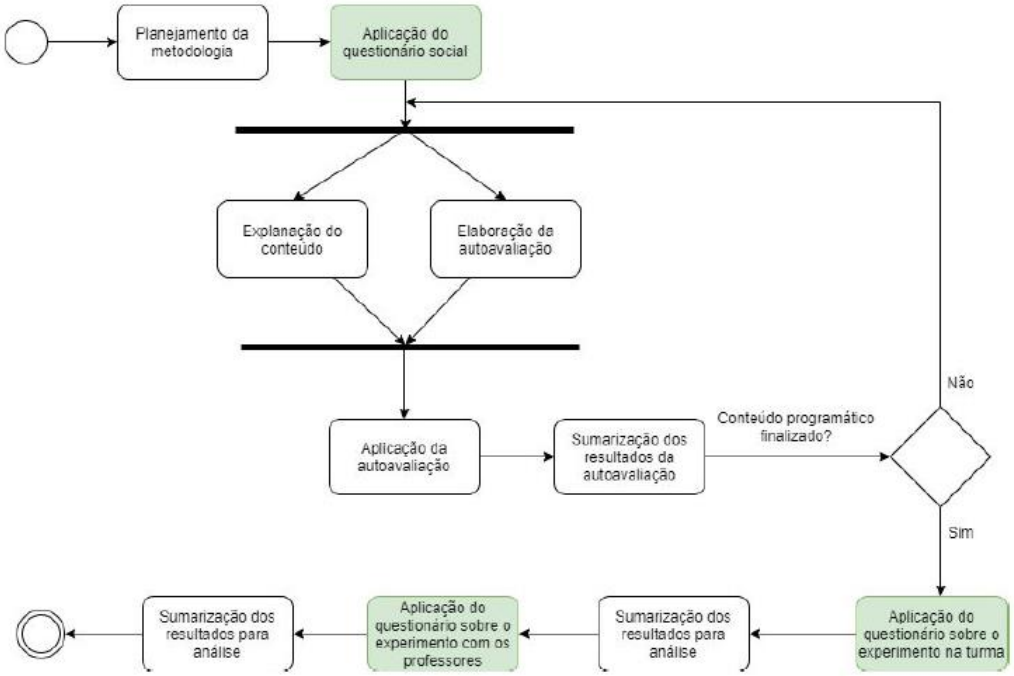
Segundo Freitas (2020), o planejamento da metodologia é a fase na qual se discute, junto com os docentes que lecionam as disciplinas, a metodologia a ser aplicada, definindo como será planejada em todo o processo de intervenção. O planejamento do professor para disciplina é adaptado para que as etapas do processo da Figura 1, principalmente, a aplicação dos cenários, sejam encaixados com o conteúdo programático da melhor forma.

As etapas indicadas com retângulos destacados em verde, envolvem a aplicação de questionários e objetiva conhecer os perfis e características das turmas, desde identificar quais subgrupos os alunos pertencem, bem como identificar os impactos das dicas nas notas finais e quais as dificuldades encontradas no decorrer da disciplina, tanto durante as realizações das avaliações quanto no geral (Freitas, 2020). As etapas de sumarização são essenciais para identificar as características das turmas, pois são nessas etapas que os resultados da aplicação dos questionários serão organizados e classificados.

Nas etapas “Elaboração das dicas por unidade” e “Aplicação das dicas por unidade”, respectivamente, se define três dicas para cada problema de programação da avaliação em curso e são disponibilizadas as dicas criadas, juntamente das avaliações.

Na realização deste trabalho houve a colaboração de Freitas (2020), onde em consultas feitas a ele, relatou-se algumas modificações no fluxo das atividades apresentadas na Figura 1. Apenas o fluxo de atividades foi modificado, mantendo a base principal, que é identificar os perfis e as características predominantes das turmas, definir os cenários (que representam problemas de programação) e dicas para auxiliar os alunos à encontrar as soluções para os problemas propostos, dentre outros elementos, conforme definido na Figura 2.

Figura 2 - Fluxo de atividades da intervenção metodológica modificado por Freitas



Fonte: Sugerido por Freitas em consultas

A etapa “Planejamento da metodologia” se mantém com o objetivo definido anteriormente, como também as etapas de sumarização e as etapas destacadas em verde, que têm, como anteriormente, o propósito de identificar os perfis e características especificas da turma. Já as etapas de “Elaboração das dicas por unidade” e “Aplicação das dicas por unidade” foram sintetizadas, respectivamente, nas etapas: “Elaboração de autoavaliação”, que se refere a elaboração das questões que irão compor a autoavaliação (os cenários, as listas de questões que passaram a ser chamados de autoavaliações) e suas respectivas dicas; e “Aplicação da autoavaliação” que se refere à aplicação das questões e dicas das autoavaliações.

O sistema de dicas projetado neste trabalho foca em automatizar esse novo fluxo de atividades, as etapas: “Elaboração de autoavaliação”, “Aplicação da autoavaliação” e “Sumarização dos resultados da autoavaliação”. As demais etapas relacionadas a aplicação de questionários podem ser abordadas numa próxima versão do projeto apresentado neste trabalho.

3 METODOLOGIA

Como o objetivo deste trabalho é desenvolver o protótipo de um sistema para informatizar a proposta de intervenção metodológica de Freitas (2020), tonando mais prática e eficiente, utilizou-se as etapas de projeto de sistemas web propostas por Garrett (2011). Nessa proposta o projeto (*design*) do sistema é guiado pelo usuário, ou seja, conforme as necessidade e interesses dos usuários e clientes (responsáveis pela solicitação do sistema), e dividido basicamente em cinco etapas descritas a seguir e que podem ser vistas na Figura 3.

- **Estratégia**: aborda as necessidades dos usuários e também as necessidades dos clientes. Corresponde a parte inferior da Figura 3, que está destacado em amarelo.
- **Escopo**: corresponde a elicitação dos requisitos funcionais do sistema e quais conteúdos o sistema conterá. Funciona como um delimitador do que será ou não acrescentado ao sistema. Associada a segunda etapa inferior, destacada em verde, na Figura 3.
- **Estrutura**: define que informações serão mostradas no sistema e a disposição dessas informações ao longo do sistema. Representada na parte intermediária, destacada em rosa, na Figura 3.
- **Esqueleto**: define os elementos de *design* de interface e como se dará a navegação entre os serviços oferecidos pelo sistema. Penúltima parte, de baixo para cima, destacada em azul mais escuro, da Figura 3.
- **Superfície**: define os detalhes dos aspectos visuais do ambiente. Corresponde a parte superior da Figura 3.

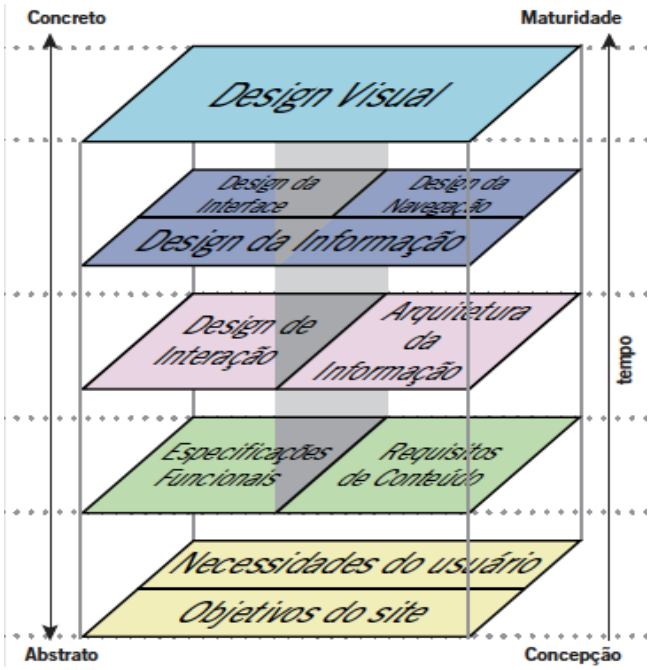
O processo proposto por Garrett (2011) é *bottom-up* (de baixo para cima). Na parte inferior são apresentados elementos mais abstratos, referentes, por exemplo, ao entendimento do sistema. À medida que se sobe sobre as etapas, o projeto do sistema vai se tornando mais concreto até a finalização dos detalhes do sistema na etapa de Superfície (*Design* Visual).

Como o processo utilizado para o projeto é guiado pelo usuário e/ou cliente, o proponente da abordagem de uso de dicas foi consultado sobre os principais aspectos do sistema. Também participou da consulta um docente que aplica a abordagem correntemente na disciplina.

Para facilitar a comunicação, foi criado um canal de comunicação instantânea entre esses envolvidos no projeto, que eram consultados diretamente por meio de mensagens à medida que o projeto foi sendo construído. Além disso, realizou-se encontros virtuais entre os

envolvidos que forneciam *feedback* sobre os artefatos criados e delimitavam direcionamentos, em caso da necessidade de mudanças.

Figura 3 - Processo de Projeto de Sistemas proposto por Garrett (2011)



Fonte: (HELLERHAUS, 2020)

A seguir, em cada seção será descrita como foram executadas e aplicadas as etapas do processo de Garret (2011) para criação do protótipo do sistema proposto. Na Seção 3.1 é descrita como foi executada a etapa Estratégia, na Seção 3.2, a etapa Escopo, na Seção 3.3 a etapa Estrutura, e as etapas Esqueleto e Superfície são descritas juntas.

3.1 ESTRATÉGIA

Nesta seção, apresenta-se a etapa em que são delimitadas as necessidades e interesses dos clientes e dos usuários do sistema. Pode-se pensar como possíveis clientes do sistema, coordenações de cursos e docentes que desejam entender melhor as turmas de programação introdutória. Nesse sentido, como já dito o trabalho contou com a colaboração de um docente que ministra a disciplina Algoritmos e do autor do trabalho mostrado em Freitas (2020), que propôs uma intervenção metodológica alvo do protótipo proposto neste trabalho, para entender quais são as necessidades de possíveis clientes do sistema. Nessa direção, alguns pontos que se imagina serem de interesse do ponto de vista do docente e coordenadores foram levantados.

- Deseja-se elevar a taxa de sucessos nas disciplinas voltadas à programação.
- Disponibilizar uma lista de problemas para turmas de alunos praticarem programação, aperfeiçoarem seus conhecimentos e superarem suas dificuldades, e sanarem dúvidas no momento que aparecem, isto é, no momento da resolução das questões por meio das dicas disponibilizadas.
- Espera-se acessar as resoluções dos problemas realizadas pelos discentes.
- Espera-se identificar quais dicas os discentes estão utilizando mais em determinados problemas referente a determinado assunto e assim determinar em que aspecto os discentes então com mais dificuldades; se é em relação a intepretação do enunciado da questão, ou em relação definir uma lógica de programação ou referente ao conteúdo teórico, a sintaxe.
- Espera-se identificar estatísticas sobre quais são os conteúdos que os discentes apresentam maiores dificuldades de forma que intervenções possam ser propostas para reduzir as dificuldades dos discentes em tópicos mais críticos. Por exemplo, o professor pode ministrar rapidamente conceitos que estão claros e focar a aula em conteúdo de maior dificuldade.
- O professor precisa dessas informações em tempo real, à medida que a turma está sendo conduzida, uma vez que as características das turmas podem ser diferentes e assim as dificuldades de uma turma em particular pode se concentrar em aspectos específicos da aula. Dessa forma, o acompanhamento deve ser feito por turma.

Do ponto de vista dos discentes, elencou-se alguns possíveis ponto de interesse:

- Elevar o desempenho na componente curricular de forma a obter êxito ao final da disciplina.
- Sanar alguns dúvida que são aparentes durante a resolução do problema.
- Receber *feedback* dos mentores sobre o seu desempenho, sobre suas dificuldades e sobre dúvidas constantemente.
- Preparar-se adequadamente para as próximas componentes do curso que precisam desse conhecimento. Por exemplo, a componente Algoritmos e Estruturas de Dados I.

3.2 ESCOPO

Nesta seção, apresenta-se os principais elementos apresentados pelo sistema, tanto do ponto de vista de requisitos do sistema e também de requisitos de conteúdo.

3.2.1 Requisitos Funcionais

De acordo com os interesses e necessidade elencadas com a colaboração do docente e do autor do trabalho Freitas (2020), foram especificados nesta versão de projeto os requisitos funcionais que devem ser atendidos prioritariamente por estarem relacionados as funções base do sistema proposto, assim como uma descrição de cada um deles, são apresentados abaixo:

- **RF001 - Cadastrar usuário**

  O sistema deverá permitir o cadastro de usuários (professor, aluno, monitor).

- **RF002 - Editar usuário**

  O sistema deverá permitir que os usuários editem seus dados cadastrados.

- **RF003 – Validar professor/monitor**

  O sistema deverá permitir ao administrador validar o cadastro de professores e monitores.

- **RF004 - Realizar login**

  O sistema deverá permitir que os usuários realizem login.

- **RF005 - Recuperar senha**

  O sistema deverá permitir que os usuários recuperem senha de login.

- **RF006 - Cadastrar turma**

  O sistema deverá permitir que professores cadastrem turmas.

- **RF007 - Editar turma**

  O sistema deverá permitir que professores editem as turmas criadas.

- **RF008 - Excluir turma**

O sistema deverá permitir que professores excluam as turmas que criaram.

- **RF009 – Buscar turma**

O sistema deverá permitir que professores/monitores busquem (pesquisem) por turmas especificas cadastradas.

- **RF010 - Visualizar turmas**

O sistema deverá permitir que professores/monitores visualizem as turmas cadastradas ao acessarem o menu das turmas.

- **RF011 - Acessar turma**

O sistema deverá permitir que os usuários (professor, monitor e aluno) acessem a turma que criou ou pertence.

- **RF012 – Compartilhar turma**

O sistema deverá permitir que professores compartilhem as turmas que criaram com monitores.

- **RF013 - Criar problema**

O sistema deverá permitir que professores/monitores criem problemas de programação.

- **RF014 - Editar problema**

O sistema deverá permitir que professores/monitores modifiquem os problemas de programação cadastrados por eles.

- **RF015 - Excluir problema**

O sistema deverá permitir que professores/monitores excluam os problemas de programação cadastrados por eles.

- **RF016 - Buscar problema**

O sistema deverá permitir que professores/monitores busquem (pesquisem) por problemas de programação específicos cadastrados.

- **RF017 - Visualizar problemas**

O sistema deverá permitir que professores/monitores visualizem os problemas cadastrados no sistema.

- **RF018 - Cadastrar dicas**

O sistema deverá permitir criar as dicas para cada problema de programação.

- **RF019 - Editar dicas**

O sistema deverá permitir modificar as dicas dos problemas de programação.

- **RF020 - Criar lista de problemas**

O sistema deverá permitir que professores/monitores criem listas de problemas de programação.

- **RF021 - Editar lista de problemas**

O sistema deverá permitir que professores/monitores editem as listas de problemas de programação cadastradas por eles.

- **RF022 - Excluir lista de problemas**

O sistema deverá permitir que professores/monitores excluam as listas de problemas de programação cadastradas por eles.

- **RF023 - Buscar lista de problemas**

O sistema deverá permitir que professores/monitores busquem por listas de problemas de programação especificas cadastradas.

- **RF024 - Acessar lista de problemas**

O sistema deverá permitir o acesso às listas de problemas de programação.

- **RF025 - Adicionar problemas**

O sistema deverá permitir que professores/monitores adicionem problemas nas listas de problemas.

- **RF026 - Retirar problemas**

O sistema deverá permitir que professores/monitores retirem problemas nas listas de problemas.

- **RF027 - Matricular aluno**

O sistema deverá permitir que professores/monitores matriculem alunos nas turmas que criaram ou pertencem.

- **RF028 - Retirar aluno**

O sistema deverá permitir que professores/monitores retirem alunos das turmas que criaram ou pertencem.

- **RF029 – Buscar aluno**

O sistema deverá permitir que professores/monitores busquem por alunos cadastrados.

- **RF030 - Visualizar alunos**

O sistema deverá permitir que professores/monitores visualizem os alunos cadastrados no sistema.

- **RF031 - Resolver problemas**

O sistema deverá permitir que alunos resolvam problemas de programação.

- **RF032 - Visualizar dicas**

O sistema deverá permitir que os alunos visualizem as dicas dos problemas de programação disponíveis.

- **RF033 - Sumarizar os dados coletados**

O sistema deverá sumarizar os resultados da resolução dos problemas de programação das turmas e todos os dados coletados em um banco de dados.

- **RF034 - Divulgar resultados**

O sistema deverá disponibilizar aos professores/monitores os resultados obtidos pelos discentes em cada resolução de problema de programação.

- **RF034 - Acessar resultados e estatísticas**

O sistema deverá permitir que professores/monitores acessem os resultados e estatísticas (dicas mais acessadas) nas resoluções dos problemas.

3.2.2 Requisitos Não Funcionais

No tocante, aos requisitos não funcionais do sistema propostos, estes são elencados abaixo:

- **RNF001 - Portabilidade (Adaptabilidade)**

O sistema deverá visualmente se adaptar ao tamanho das telas que estão sendo exibidas, como as telas de celulares e *tablets*.

- **RNF002 - Portabilidade (Capacidade para ser instalada)**

O sistema deverá funcionar nos principais *browsers* existentes.

- **RNF003 - Usabilidade (Atratividade)**

O sistema deverá ter opções para aumentar a fonte, mudar a cor de fundo e do texto.

- **RNF004 - Funcionalidade (Segurança de acesso)**

O sistema deverá realizar autenticação e criptografia nas informações de login.

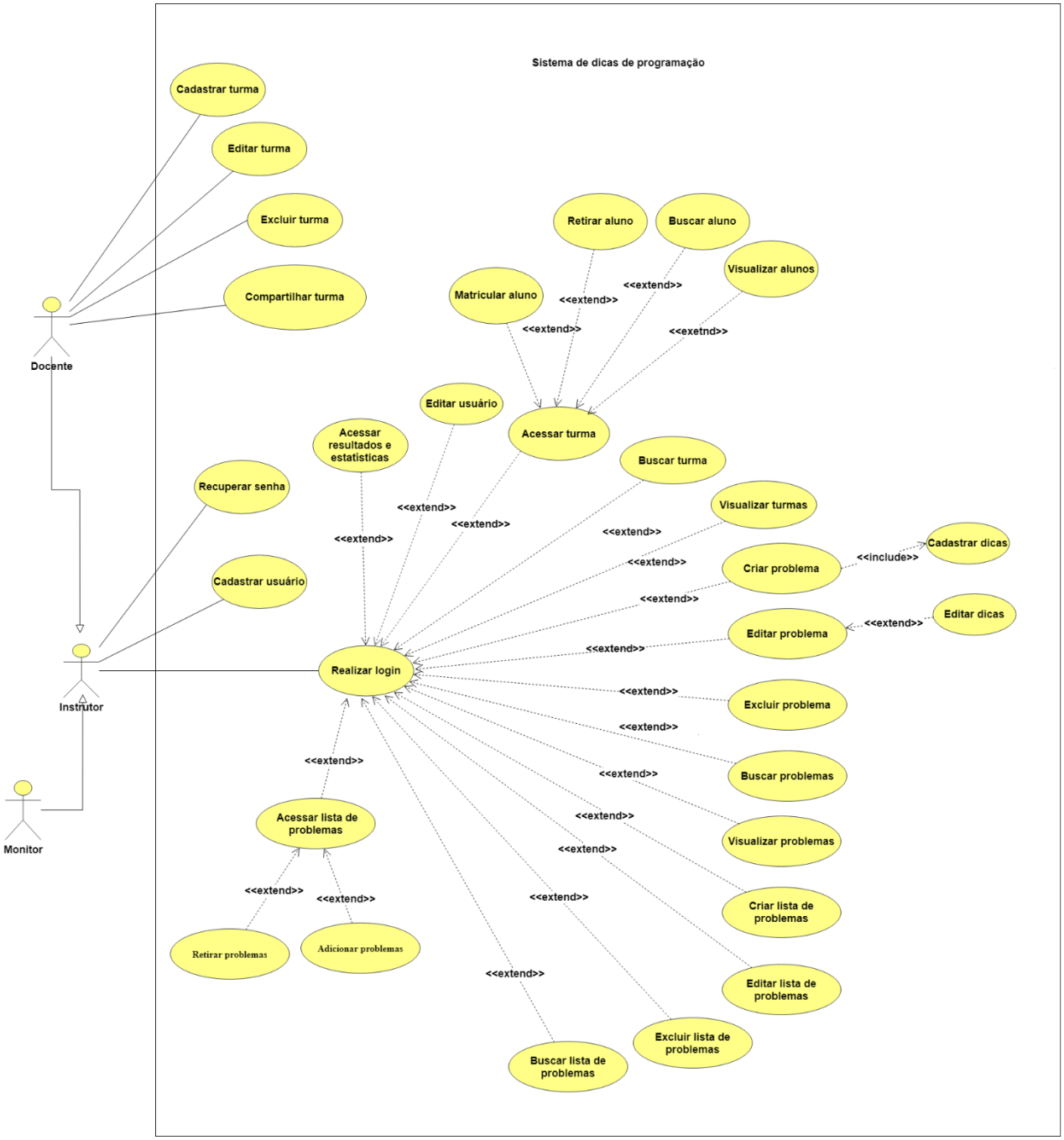
3.2.3 Diagrama de Casos de Uso

Para agrega a está etapa, utilizou o modelo de casos de uso da UML (*Unified Modeling Language*) através da criação do Diagrama de Casos de Uso para o sistema proposto, pois conforme Stadzisz (2002), trata de um instrumento eficiente para definir e/ou documentar os serviços, requisitos funcionais a serem realizados pelo sistema, e após criado o diagrama de casos de uso, guia o projeto por ele.

Segundo Stadzisz (2002), o diagrama de casos de uso do sistema, como outros da UML, utiliza como primitivas Atores, que são simbolizados por bonecos de palito e correspondem aos usuários ou serviços que utilizam o sistema; os casos de uso são representados por uma forma oval rotulada, e os Relacionamentos são indicados por linhas entre o ator e o caso de uso ou entre caso de uso e caso de uso.

Na idealização do sistema foram definidos quatro tipos de usuário, neste caso os atores, que são nomeados de administrador, professor, monitor e aluno, na qual foram instituídas funcionalidades especificas para cada tipo. O Professor e monitor foram tratados como uma especialização de “Instrutor”, por possuírem muitas funcionalidades iguais tendo apenas algumas mais especificas para o professor. O diagrama foi dividido em duas figuras para uma melhor organização. O diagrama de casos de uso do Docente/Monitor pode ser visto na Figura 4 e na Figura 5 pode ser visto o diagrama referente ao Discente e ao Administrador.

Figura 4 - Diagrama de casos de uso de Docentes e Monitores



Fonte: Autoria própria.

Figura 5 - Diagrama de casos de uso de Administradores e Discentes

Fonte: Autoria própria.

Nos quadros elencados abaixo é apresentado a especificação funcional, a descrição detalhada do funcionamento dos casos de uso do sistema de maior complexidade e que podem gerar dúvidas sobre seu funcionamento.

- **CSU – Criar problema**

Quadro 1 - Descrição do caso de uso Criar problema

Prioridade	Essencial
Ator Principal	Instrutor (docente e monitor)
Atores Secundários	-
Resumo	O sistema deve permitir que docentes/monitores criem problemas de programação no sistema.
Precondições	**CSU - Realizar login**
Fluxo Principal	
Ação do Ator	**Ação do Sistema**

O instrutor clica na opção “Criar problema”.	O sistema apresenta janela que solicitar que ele inserir o título, a descrição e o enunciado.
O instrutor preenche essas informações e clica no botão “Próximo”. **A1. E1. E2.**	O sistema solicita que ele realize o cadastro das dicas para o problema.
CSU – Cadastra Dicas. A1. E1. E2.	O sistema apresenta uma última janela para se inserir as informações finais, por exemplo, informar a que tópico de aula o problema se refere-se, vincular ou não o problema a uma lista e se o problema será público ou privado.
O instrutor clica em “Cadastrar”. **A1. E1. E2.**	O sistema salva o problema no sistema.
Fluxos Alternativos	
A1. Cancelar	
Ação do Ator	**Ação do Sistema**
O instrutor clica em “Cancelar”.	O sistema pergunta se tem certeza que quer realizar a ação.
O instrutor confirma “sim”. **A2**	O sistema descarta os dados já preenchidos e retorna para o estado que se encontrava.
A2. Não deseja realizar a ação	
Ação do Ator	**Ação do Sistema**
O instrutor confirma “não”, para não realizar a ação.	O sistema retorna a tela ou ação anteriormente.
Fluxos de Exceção	
E1. Campo obrigatório não preenchido	
Ação do Ator	**Ação do Sistema**
-	O sistema informa que algum campo obrigatório não foi preenchido.
-	Sistema solicita para o usuário preencher o campo faltando.
E2. Campo preenchido incorretamente	
Ação do Ator	**Ação do Sistema**
-	O sistema informa que algum campo foi preenchido incorretamente.
-	O sistema solicita para o instrutor preencher o campo corretamente.

Pós-condições	O problema de programação criado é salvo no sistema.

Fonte: Autoria própria.

- **CSU – Resolver problema de programação**

Quadro 2 - Descrição do caso de uso Resolver problema de programação

Prioridade	Essencial
Ator Principal	Discente
Atores Secundários	-
Resumo	O sistema deve permitir que discente resolvam problemas de programação das listas.
Precondições	**CSU - Realizar login. CSU – Acessar lista de problemas.**
Fluxo Principal	
Ação do Ator	**Ação do Sistema**
O discente no menu clica no botão resolver da lista de problemas que desejar.	O sistema apresenta a tela com as informações da lista de problemas e carrega junto o primeiro problema ou o problema selecionado da lista.
-	O sistema apresenta o título, a descrição e o enunciado da questão e disponibiliza o editor de código e um terminal para testes.
O discente implementa o código no editor de código e clica no botão próximo. **CSU-Visualizar dicas. A1. A2.**	O sistema salva a implementação feita pelo discente para aquele problema no sistema e apresenta o próximo problema, caso a lista tenha.
O discente após resolver todos os problemas da lista. Clica em “Finalizar lista”. **A1.**	O sistema salva todas os dados referentes a resolução dos problemas feitos pelo discente e retorna para tela de listagem de listas de problemas.
Fluxos Alternativos	
A1. Sair sem finalizar	
Ação do Ator	**Ação do Sistema**
Usuário sair da página de resolução.	O sistema salva as atividades já feitas, resoluções dos problemas já resolvidos e ainda deixa a lista disponível para caso o discente retorne a resolver.
A2. Finaliza lista antes de resolver todos os problemas	
Ação do Ator	**Ação do Sistema**

O discente clica em finalizar lista sem ter resolvido todas as questões.	O sistema salva as atividades já feitas, resoluções dos problemas já resolvidos e ainda deixa a lista disponível para caso o discente retorne a resolver.
Fluxos de Exceção	
-	
Pós-condições	Resolução dos problemas salvos no sistema.

Fonte: Autoria própria.

- **CSU – Compartilhar turma**

Quadro 3 - Descrição do caso de uso Compartilhar turma

Prioridade	Essencial
Ator Principal	Docente
Atores Secundários	-
Resumo	O sistema permitir que docentes compartilhem com monitores as turmas criadas por ele.
Precondições	**CSU - Realizar login. CSU – Acessar turma**
Fluxo Principal	
Ação do Ator	**Ação do Sistema**
O docente clica na opção compartilhar turma na turma que deseja e que está listada no menu acessar turma.	O sistema apresenta uma janela que solicita o e-mail dos monitores cadastrados no sistema que deseja compartilhar.
O docente insere o e-mail dos monitores que deseja compartilhar. **A1.**	O sistema realizar a busca com o e-mail e apresenta o resultado da busca. **E1.**
O docente clica em adicionar ao lado do nome do monitor encontrado e clica na opção "Compartilhar". **A1.**	O sistema disponibiliza e permite o acesso a turma para os monitores adicionados.
Fluxos Alternativos	
A1. Cancelar	
Ação do Ator	**Ação do Sistema**
O docente clica em "Cancelar".	O sistema pergunta se tem certeza que quer realizar a ação.
O docente confirma "sim". **A2**	O sistema descarta os dados já preenchidos e retorna para a página que se encontrava.
A2. Não deseja realizar a ação	

Ação do Ator	**Ação do Sistema**
O docente confirma “não”, para não realizar a ação.	O sistema retorna a tela ou ação anteriormente.
Fluxos de Exceção	
E1. Nenhum monitor encontrado	
Ação do Ator	**Ação do Sistema**
-	O sistema informa que nenhum monitor com o e-mail inserido foi encontrado e está cadastrado no sistema.
-	O sistema solicita que o docente insira o e-mail de um monitor cadastrado.
Pós-condições	O sistema salve os monitores na turma, e esses tenham acesso a turma.

Fonte: Autoria própria.

- **CSU – Matricular aluno**

Quadro 4 - Descrição do caso de uso Matricular aluno

Prioridade	Essencial
Ator Principal	Instrutor (docente e monitor)
Atores Secundários	-
Resumo	O sistema permite que os professores/monitores matriculem alunos nas suas turmas.
Precondições	**CSU - Realizar login. CSU – Acessar turma**
Fluxo Principal	
Ação do Ator	**Ação do Sistema**
O instrutor clica na opção matricular aluno na turma que deseja e que está listada no menu acessar turma.	O sistema apresenta uma janela que solicita o e-mail dos alunos cadastrados no sistema que deseja matricular.
O instrutor insere os e-mails dos alunos, um por um, que deseja matricular. **A1.**	O sistema realiza a busca com o e-mail e apresenta o resultado da busca. **E1.**
O instrutor clica em adicionar ao lado do nome do aluno encontrado. Após todos alunos adicionados, o instrutor deve clicar em “finalizar”. **A1.**	O sistema salva os alunos na turma e disponibiliza o acesso a esses.
Fluxos Alternativos	

A1. Cancelar	
Ação do Ator	**Ação do Sistema**
O instrutor clica em “Cancelar”.	O sistema pergunta se tem certeza que quer realizar a ação.
O instrutor confirma “sim”. **A2**	O sistema descarta os dados já preenchidos e retorna para página que se encontrava.
A2. Não deseja realizar a ação	
Ação do Ator	**Ação do Sistema**
O instrutor confirma “não”, para não realizar a ação.	O sistema retorna a tela ou ação anteriormente.
Fluxos de Exceção	
E1. Nenhum aluno encontrado	
Ação do Ator	**Ação do Sistema**
-	O sistema informa que nenhum aluno com o e-mail inserido foi encontrado e está cadastrado no sistema.
-	O sistema solicita que o instrutor insira o e-mail de um aluno cadastrado.
Pós-condições	O sistema salva os alunos nas turmas, e esses tenham acesso à turma.

Fonte: Autoria própria.

- **CSU – Acessar resultados e estatísticas**

Quadro 5 - Descrição do caso de uso Acessar resultados e estatísticas

Prioridade	Essencial
Ator Principal	Instrutor (docente e monitor)
Atores Secundários	-
Resumo	O sistema deverá permitir que professores/monitores acessem os resultados e estatísticas (dicas mais acessadas) nas resoluções dos problemas.
Precondições	**CSU - Realizar login.**
Fluxo Principal	
Ação do Ator	**Ação do Sistema**

O instrutor clica no menu "*Dashboard*".	O sistema apresenta uma página com as estatísticas em tempo real sobre as resoluções de problemas feitos pelos alunos, como, por exemplo, as dicas mais acessadas, quantos alunos estão resolvendo determinado problema, etc. **E1.**
A1.	-
Fluxos Alternativos	
A1. Sair da página.	
Ação do Ator	**Ação do Sistema**
O instrutor clica em outro menu ou opção.	O sistema sair da tela de *Dashboard* e carrega o menu ou opção clicada.
Fluxos de Exceção	
E1. Nenhum resultado encontrado	
Ação do Ator	**Ação do Sistema**
-	O sistema informa que não há nenhuma estática ou resultado para aquele problema.
Pós-condições	O sistema apresenta os resultados e estatísticas dos dados sumarizados das resoluções de problemas feito pelo alunos.

Fonte: Autoria própria.

3.2.4 Requisitos de Conteúdo

Alguns elementos são identificados como requisitos de conteúdo do sistema, são eles:

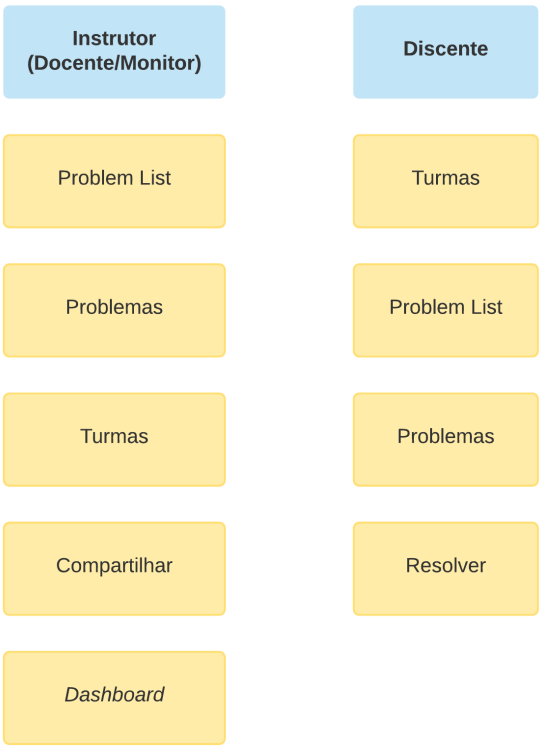
- Descrição do problema. Cada problema deve apresentar uma descrição do que é requisitado para a proposta de uma solução.
- Cada problema deve apresentar exemplos de entrada e saída para guiar o usuário na resolução do problema.
- O problema também deve apresentar uma descrição da entrada e da saída esperada.
- Para cada problema, se faz necessário elaborar três dicas para guiar o discente na resolução do problema.
- Cada problema também pode apresentar material de aula para o discente acessar de forma *off-line* ao acessar a turma.

3.3 ESTRUTURA

Nesta etapa foram definidas que informações e a sua disposição para os usuários. Os principais elementos associados à arquitetura da informação do sistema são mostrados na Figura 5. O docente possui várias opções à sua disposição; ele pode interagir com as listas de problemas, pode operar sobre problemas, assim como interagir com turmas, compartilhar informações e consultar o *dashboard* de turmas. Da mesma forma, o monitor possui um conjunto similar de operações. De outra forma, o discente possui acesso as turmas que estão matriculados e interagem com os problemas, os consultando nas listas e também os resolvendo ao longo da disciplina e dos momentos de aula.

O delineamento associado a cada uma dessas opções é elencado no Capitulo 5 (Seção 5.1), em que é detalhado o resultado das etapas de Esqueleto e Superfície (Protótipo final).

Figura 6 - Delineamento das opções



Fonte: Autoria própria.

3.4 ESQUELETO E SUPERFÍCIE

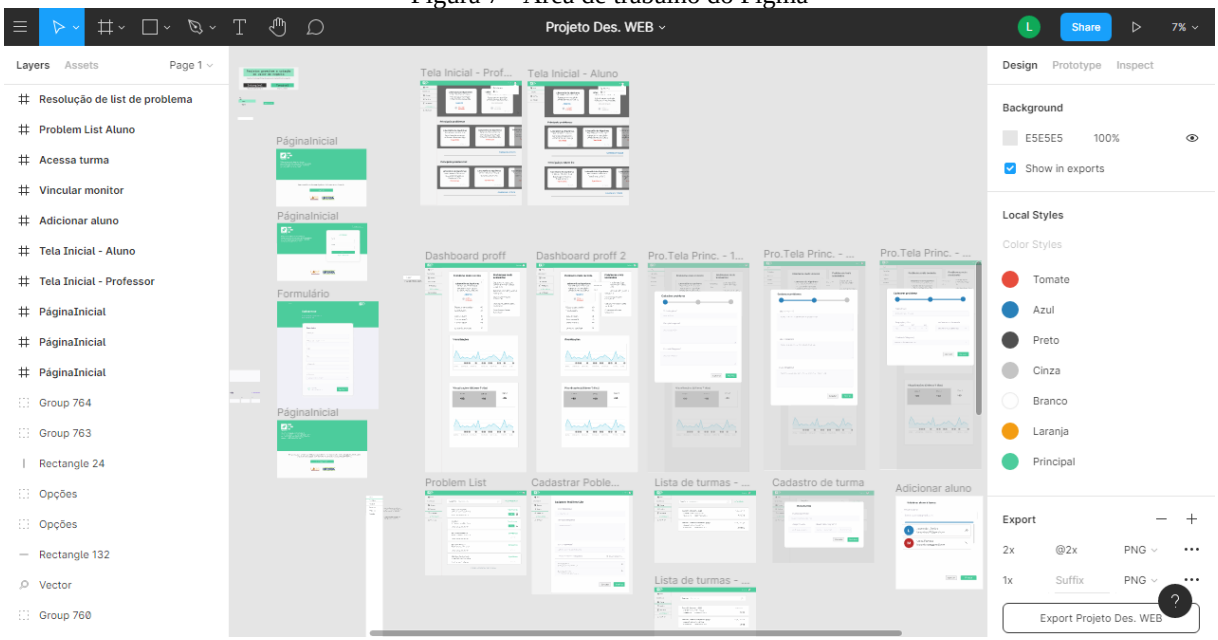
Nesta seção são descritas as etapas de esqueleto em que foram selecionados os elementos de interface corretos, definidos os principais elementos referentes a navegação que orientam o usuário e como são apresentadas as informações, e a de superfície na qual definiu-

se os detalhes do *design* visual do sistema (Cores, Tipografia, Imagens, Elementos de Identidade Visual, etc).

Como resultado das duas etapas foi construído o protótipo final do sistema que relacionou todos esses elementos de *design* de interface citados. Para isso, foi utilizada a ferramenta de *design* de interface Figma[1], ferramenta colaborativa e *online*, em que todo trabalho é feito utilizando o navegado, tornando-a compatível com todos os sistemas operacionais (SIRIUS INTERATIVA, 2019). Na Figura 6 é mostrada a área de trabalho do Figma.

O protótipo detalhado, descrevendo todo o seu funcionamento é discutido na Seção 5.1 do Capitulo 5 (Resultados).

Figura 7 - Área de trabalho do Figma



Fonte: Autoria própria.

[1] Disponível em: <https://www.figma.com>

4 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados ao proposto, buscando auxiliar os discentes a melhorarem o seu desempenho com atividades referentes à programação de computadores.

4.1 URI ONLINE JUDGE

Segundo Selivon (2015), o portal URI *Online Judge* é um projeto desenvolvido desde 2011 na URI - Universidade Regional Integrada - Campus de Erechim, com o objetivo de fornecer desafios no padrão do ICPC (*International Collegiate Programming Contest*) da ACM (*Association for Computing Machinery*), incluindo para o usuário correção em tempo real, utilização de juízes para testar as soluções implementadas, fórum e aceitação de soluções em diferentes linguagens de programação.

O ponto em destaque dessa ferramenta é a integração do portal com módulo denominado de '*Academic*', criado em 2013, que possibilita aos professores acompanhar melhor a evolução e rendimentos de seus alunos (SELIVON, 2015). Assim junto às demais características do URI, o professor tem controle das atividades propostas em horários de aulas e extraclasse, pode definir um período para resolução de uma lista de exercício, há também a possibilidade de o professor conferir as submissões realizadas pelo aluno e realizar o controle contra plágio das soluções implementadas pelos alunos para os problemas.

4.2 THE HUXLEY

Ferramenta web desenvolvida na Universidade Federal de Alagoas, também, focada em fornecer problemas de sua base de dados para alunos aprenderem, exercitarem, aprimorarem suas habilidades durante a resolução dos problemas (PAES, *et al.*, 2013). Suas principais características são: para o aluno, a possibilidade de programar direto na plataforma, correção automática das soluções por análise sintática do código e testes de aceitação, envio de *feedbacks* das correções aos alunos; e, para o professor, oferece possibilidade de criar diversas turmas, aplicar exercícios/provas e ter o controle do desempenho de cada uma com dados estáticos, incluindo a quantidade de problemas resolvidos, porcentagem de acertos/erros, erros específicos de cada aluno, visualizar quais alunos estão abaixo do esperado, entre outros dados.

A correção automática das soluções desta ferramenta junto com sua disponibilização ao professor de dados estatísticos são os diferenciais, pois permitem que os professores tenham uma visão mais analítica e fidedigna do desempenho dos alunos, e assim, possam atacar de forma eficaz problemas específicos de cada aluno, sem perder tempo com a atividade de correção.

4.3 PROALG

O sistema PROALG é também uma ferramenta web desenvolvida por Cléverton Heming (2018) em seu trabalho de conclusão de curso na Universidade do Vale do Taquari - UNIVATE, a qual possibilita o cadastro de professores, em que estes criam e gerenciam turmas, possibilita vincular nas turmas seus alunos que estão cadastrados na ferramenta, gerar para os alunos *templates* de exercícios de programação, incluindo dicas ou não, e assim apoiar o aprendizado de algoritmos e programação.

O sistema proposto por Heming (2018) é dividido basicamente em quatro módulos, um para cada tipo de usuário definido para o sistema, nomeados como administrador, coordenador, professor e aluno. Cada módulo possui funcionalidades pertinentes ao tipo de usuário. Os módulos do administrador e coordenador são destinados mais para organização e administração do sistema, por exemplo, realizar o cadastro e controle dos usuários, instituições, cursos, turmas, entre outras funções. Já as partes mais voltadas para auxiliar o aprendizado são o módulo do professor e o módulo do aluno, sendo os módulos de maior destaque.

Segundo Heming (2018), a ferramenta desenvolvida para os alunos exercitarem seus conhecimentos e resolverem atividades de programação implementa etapas e fundamentos da metodologia da problematização com Arco de Maguerez. Das etapas implementadas, a primeira denominada de “pontos-chaves”, a segunda de “hipótese de solução” e a terceira de “código fonte”.

Para o aluno, ao selecionar para resolver um exercício da turma que ele está vinculado é entregue uma interface com a descrição do problema e as etapas da metodologia implementada listadas para a resolução deste. Na primeira, “ponto-chaves”, o aluno é instigado a definir pontos-chaves do problema; na segunda, “hipótese de solução”, o aluno com base no problema deve descrever uma hipótese de solução pensada por ele para solução do problema, tendo como base os pontos-chaves e dicas; e a terceira e última etapa, “código-fonte” é o momento de realizar a implementação do código fonte. Para isso é fornecido um editor de

código. Ao final da última etapa do exercício, o aluno tem a possibilidade de testar o código fonte implementado por ele, receber avisos de erros na construção ou execução do programa, e saber a saída gerada pelo programa.

Os professores recebem as resoluções dos exercícios dos seus alunos e realizam as correções. Para isso, conta com um painel, que além da resposta do aluno, os possibilitam saber todos os movimentos do cursor no editor de código no momento em que o aluno está implementado, como também ações de copiar e colar; os possibilitam saber também quais dicas foram disponibilizadas e quantas vezes os alunos utilizaram, quantidade de tentativas, saber o tempo gasto para resolver cada etapa e saber quantas compilações foram realizadas (quantas deram erro e quantas deram certo).

Uma ferramenta com todas essas funcionalidade, tecnologias e opções para o professor, como proposto por Heming (2018), permite entender qual linha de raciocínio utilizado por cada aluno, analisa o rendimento do aluno, possibilita estudar e compreender as dificuldades dos alunos no aprendizado de introdução a programação, e assim, converter tudo isso em melhorias na forma de ensino de algoritmos e programação, ou até mesmo atacar as maiores dificuldades dos alunos e/ou as dificuldades específicas de cada aluno.

O trabalho proposto se diferencia dos demais por automatizar a abordagem proposta por Freitas (2020). Assim como Heming (2018), permite ao professor receber estatísticas sobre a resolução do problema. Além disso, permite ao professor um acompanhamento em tempo real do conjunto de resoluções produzidas pelo estudante e permite a ele decidir que pontos específicos abordar no momento de interação com os estudantes.

5 RESULTADOS

Este capítulo apresenta e discute os resultados da aplicação do processo de *design* guiado pelo usuário de Garrett (2011). Dessa forma, na Seção 5.1 é apresentado o protótipo final do sistema, mostrando as interfaces, telas que os usuários irão interagir e descrevendo os detalhes de navegação.

5.1 PROTÓTIPO

Nessa seção, é descrito o protótipo do sistema, detalhando os elementos, as opções e interações das interfaces.

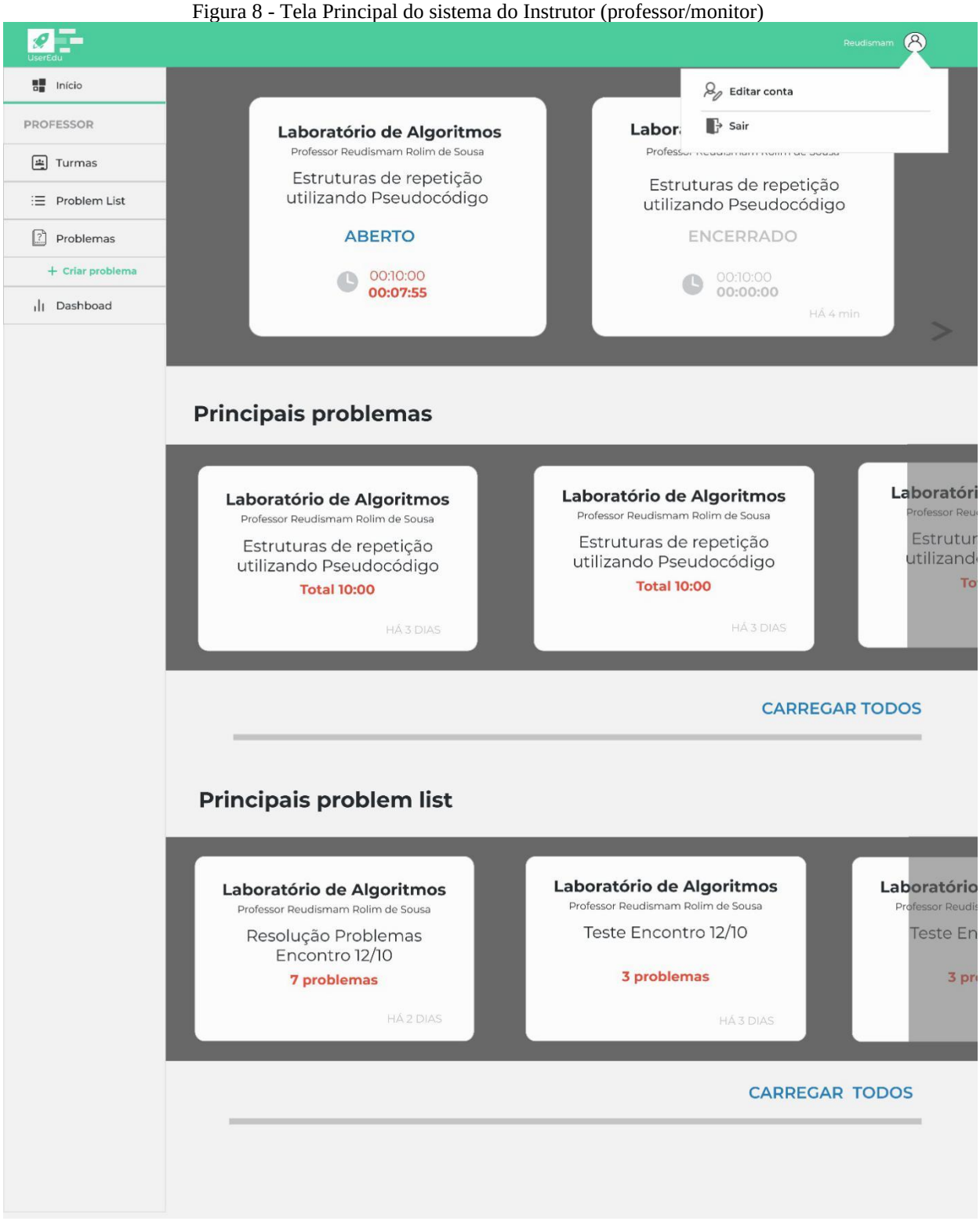
Nas Figura 8 e 9 são apresentadas as telas principais do instrutor e do discente, respectivamente. Tanto o usuário instrutor (professor/monitor) e discente terão visualizações similares de tela inicial. A diferença é que a tela de professor e monitor (Figura 8) apresenta no menu lateral o menu *dashboard* e o menu problemas com a opção "+ criar problema".

Além da visualização de telas iguais, o professor e monitor possui diversas funções iguais, com exceção de duas que são exclusivas para os professores, cadastrar novas turmas e vincular monitores a elas. Dessa forma, a tela de listagem de turmas do professor possui mais opções em relação a tela do monitor, como pode ser visto comparando a Figura 10 e a Figura 11, respectivamente.

O professor é responsável por cadastrar as turmas, para isso basta acessar o menu turma no menu lateral e pressionar o botão "Nova turma", disponível na tela de listagem, como mostrado na Figura 10. Um *pop-up* é apresentado para o cadastro de turmas (Figura 12); nele é informado a disciplina, o horário das aulas e o semestre letivo. O cadastro é concluído ao clicar no botão "Cadastrar".

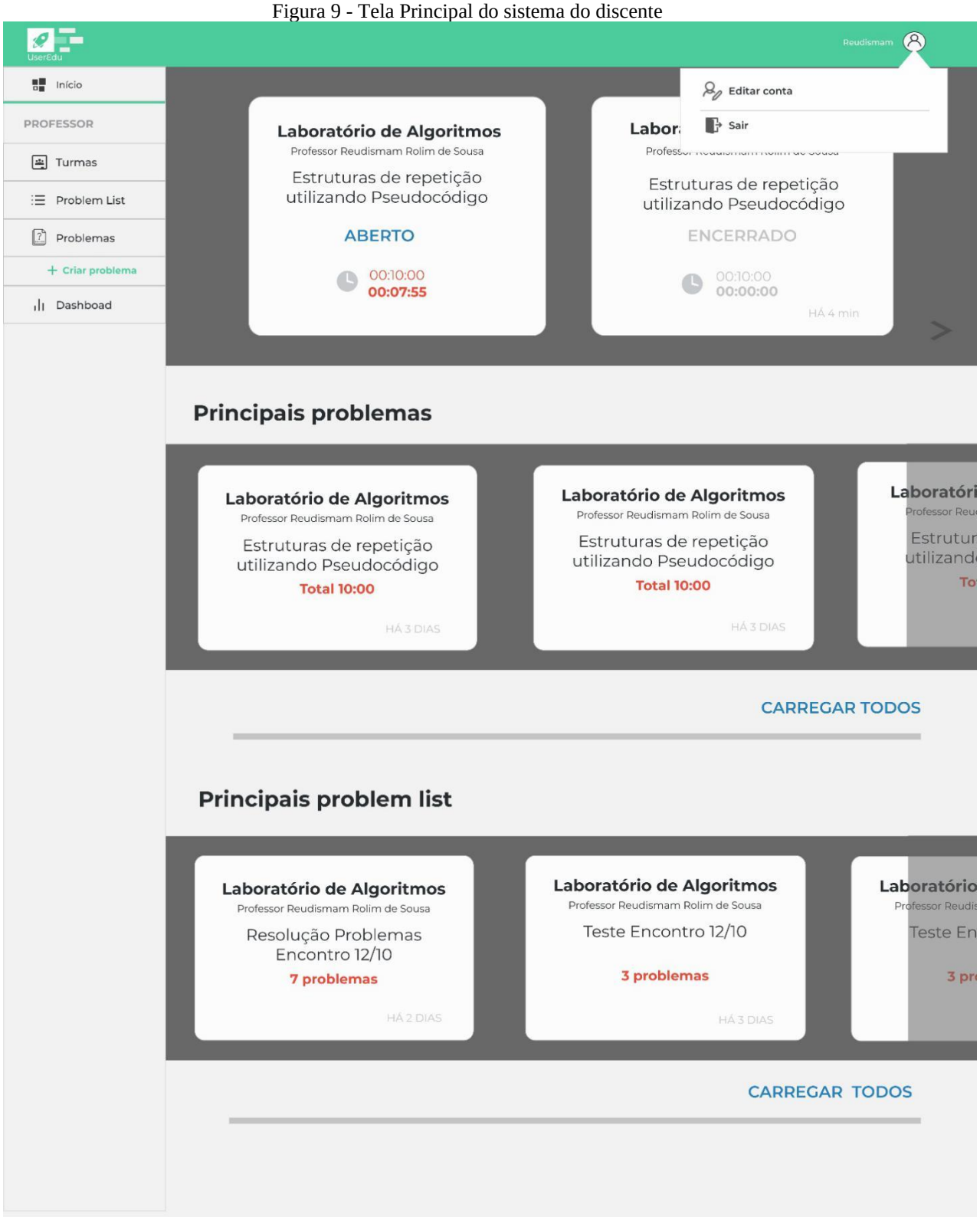
Após cadastrar a turma, o docente pode vincular a ela um monitor, clicando no botão compartilhar, ao lado direito da opção editar, presente na tela de listagem ao acessar o menu de turmas (Figura 8). Em seguida é apresentada uma janela solicitando o e-mail do monitor já cadastrado no sistema (Figura 13), o professor insere os dados e clica em buscar; o sistema busca e apresenta o monitor correspondente ao e-mail, para colocar o monitor na lista daqueles que terão acesso a turma, clica-se no botão adicionar ao lado do nome do monitor, por fim, clica-se no botão compartilhar, e todos adicionados na lista terão acesso à turma.

Figura 8 - Tela Principal do sistema do Instrutor (professor/monitor)



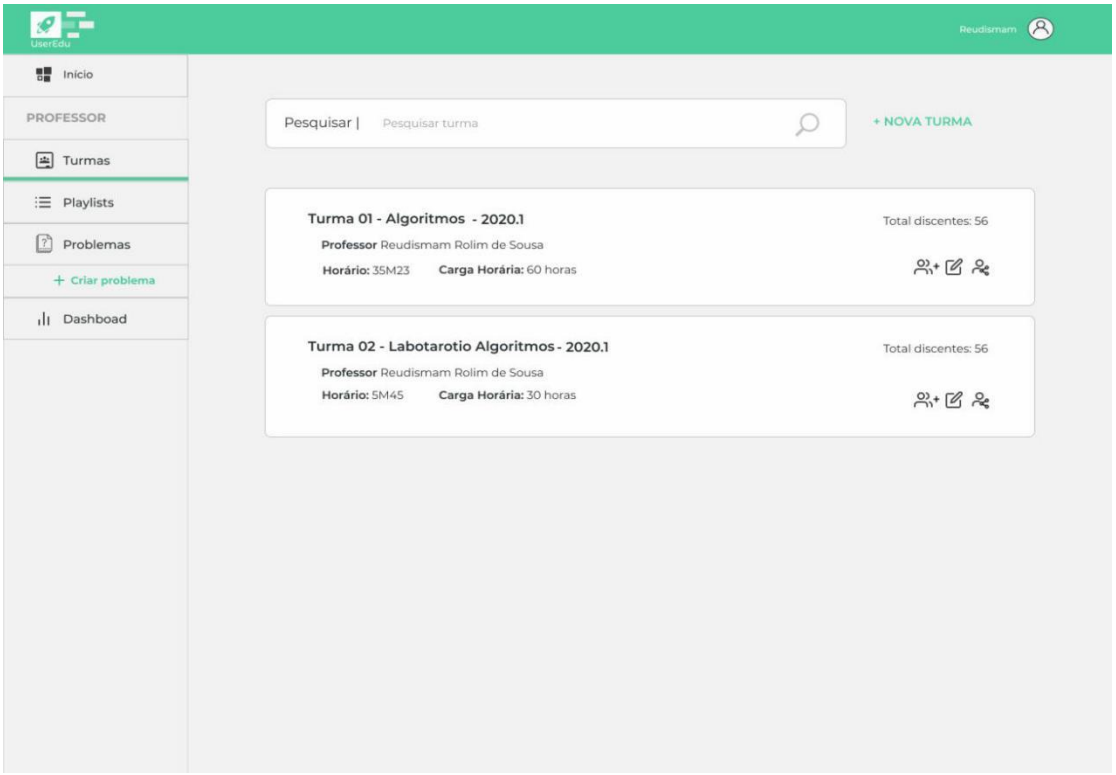
Fonte: Autoria própria.

Figura 9 - Tela Principal do sistema do discente



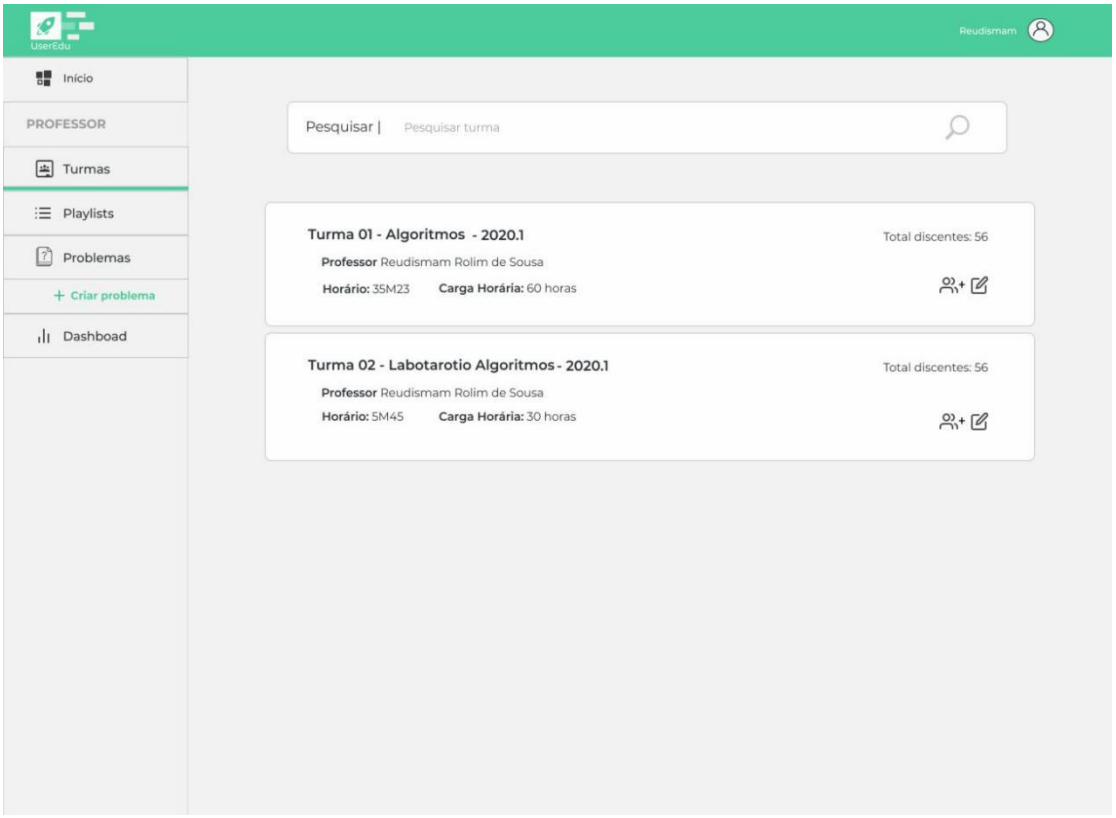
Fonte: Autoria própria.

Figura 10 - Listagem de turmas de professor



Fonte: Autoria própria.

Figura 11 - Listagem de turmas de monitor

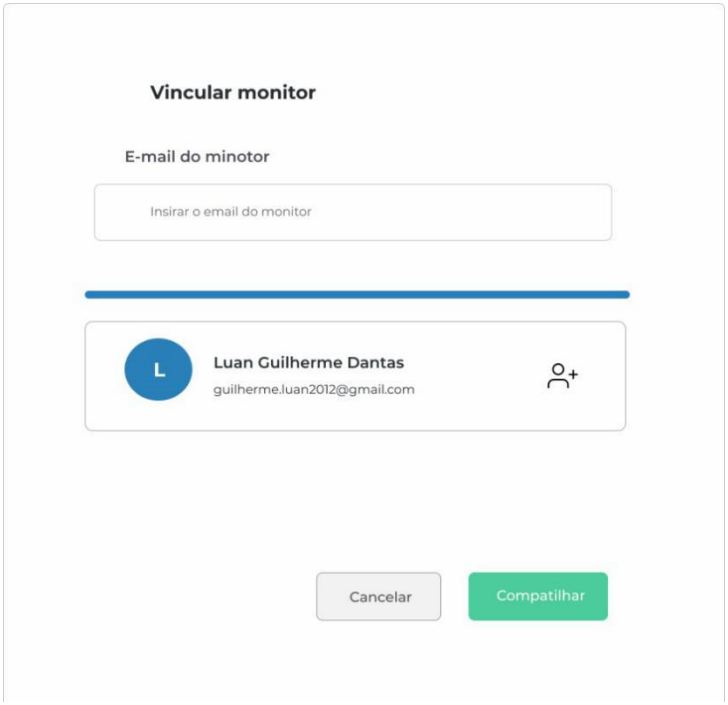


Fonte: Autoria própria.

Figura 12 - *Pop-up* de cadastro de turma

Fonte: Autoria própria.

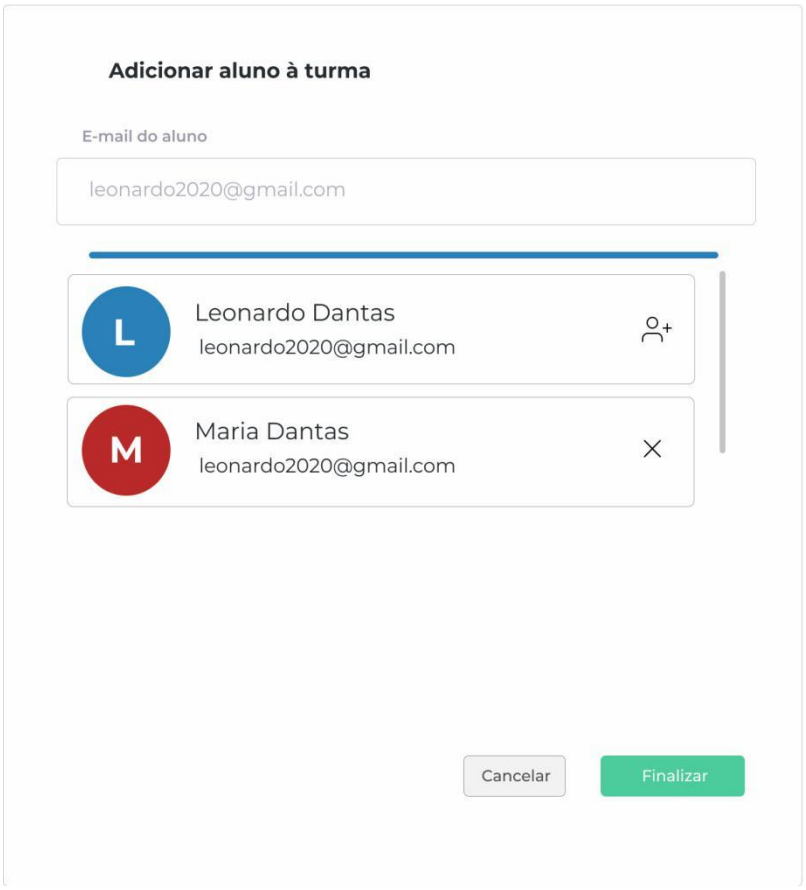
Figura 13 - *Pop-up* de adicionar monitor a turma



Fonte: Autoria própria.

Os alunos devem ser vinculados às turmas pelo docente ou monitor ao clicar no botão adicionar alunos ao lado esquerdo do botão editar disponível nas suas respectivas telas de listagem de turmas (Figura 10 e Figura 11). Todos os alunos que serão adicionados devem estar cadastrados no sistema. Uma janela será apresentada para se buscar o aluno pelo seu e-mail cadastrado, após encontrado, o aluno é inserido quando pressionado o botão adicionar ao lado de seu nome, adicionado todos para concluir basta clicar em finalizar (Figura 14).

Figura 14 - *Pop-up* de adicionar aluno a turma

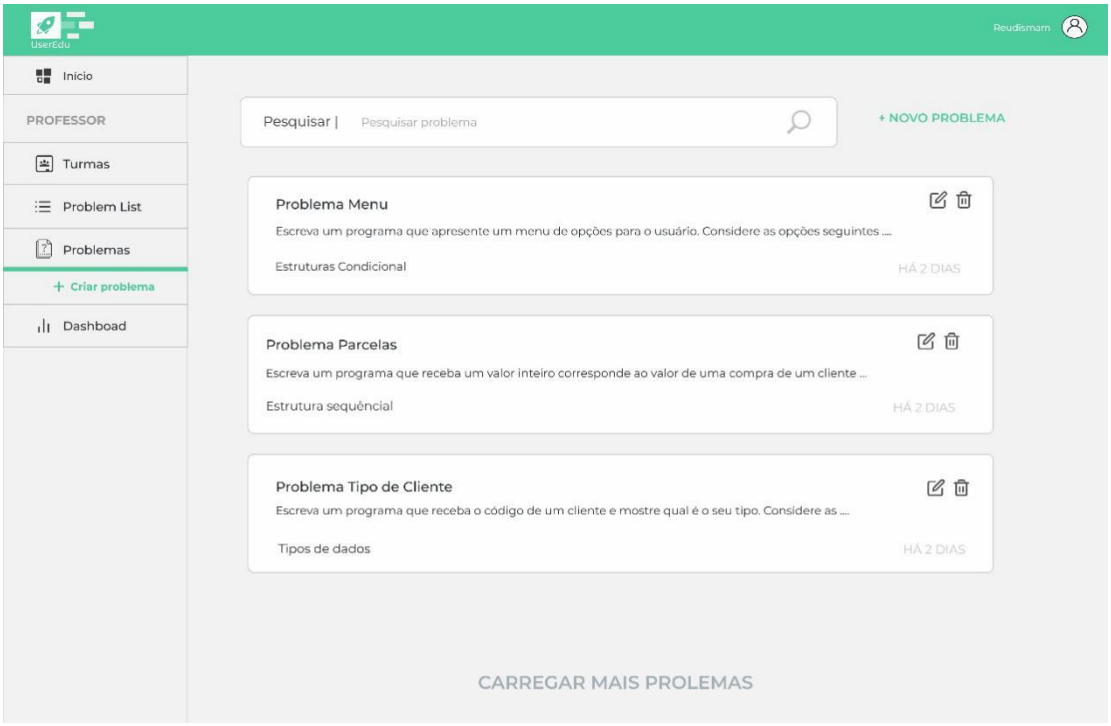


Fonte: Autoria própria.

Os problemas que serão resolvidos pelos alunos são cadastrados pelo professor ou monitor e estão vinculados a eles, tornando possível o seu uso em diferentes turmas. Outra característica é poderem ser disponibilizados como público, para que outros professores possam utilizar o problema. Para realizar o cadastro, pode-se clicar diretamente na opção criar problema do menu problemas no menu lateral ou acessar o menu problemas e pressionar o botão novo problema presente na tela de listagem (Figura 15), em seguida, passará por três etapas.

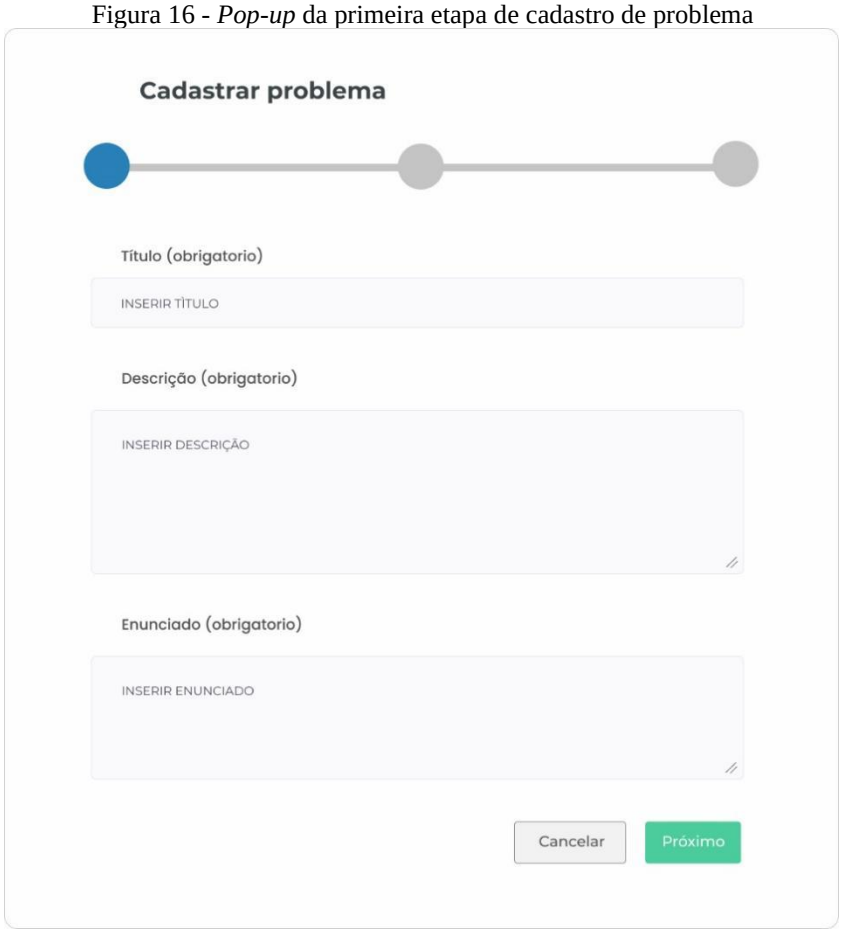
Na primeira etapa, é solicitado o título, a descrição do problema e o enunciado, como pode ser visto na Figura 16. É importante definir bem a sua descrição para que o discente seja estimulado a refletir sobre o contexto. Concluída a inserção dos dados, ele passará para a próxima etapa, clicando no botão próximo.

Figura 15 - Listagem dos problemas cadastrados



Fonte: Autoria própria.

Figura 16 - *Pop-up* da primeira etapa de cadastro de problema

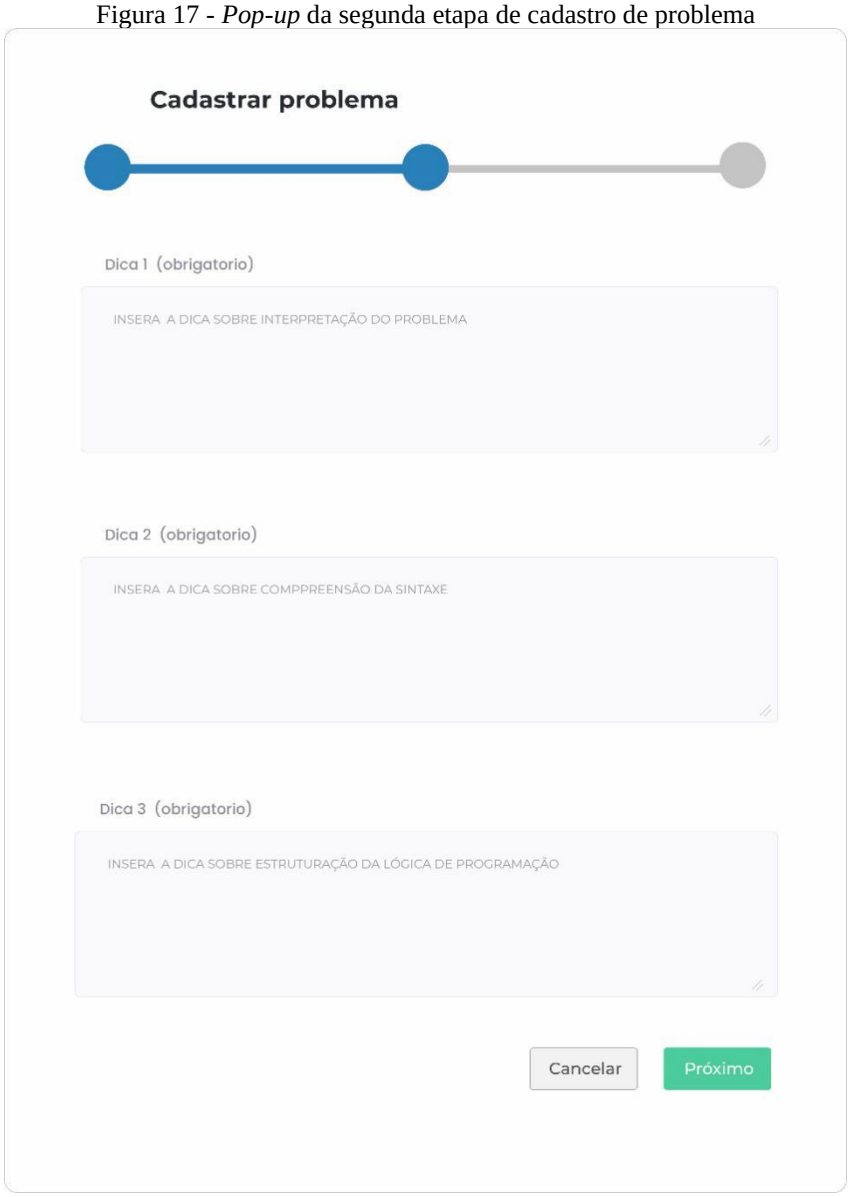


Fonte: Autoria própria.

Na segunda etapa, o professor/monitor, no *pop-up* mostrado na Figura 17, deve inserir as dicas que serão disponibilizadas aos alunos durante a realização da questão e auxiliar no entendimento e no desenvolvimento de uma solução. As dicas devem seguir a ordem e classificação: (i) interpretação do problema; (ii) compreensão da sintaxe; e (iii) estruturação da lógica de programação. Essa classificação foi proposta por Moreira et al. (2018) e citada por Freitas (2020) em seu trabalho de conclusão de curso, uma proposta de intervenção metodológica para auxiliar a aprendizagem de programação.

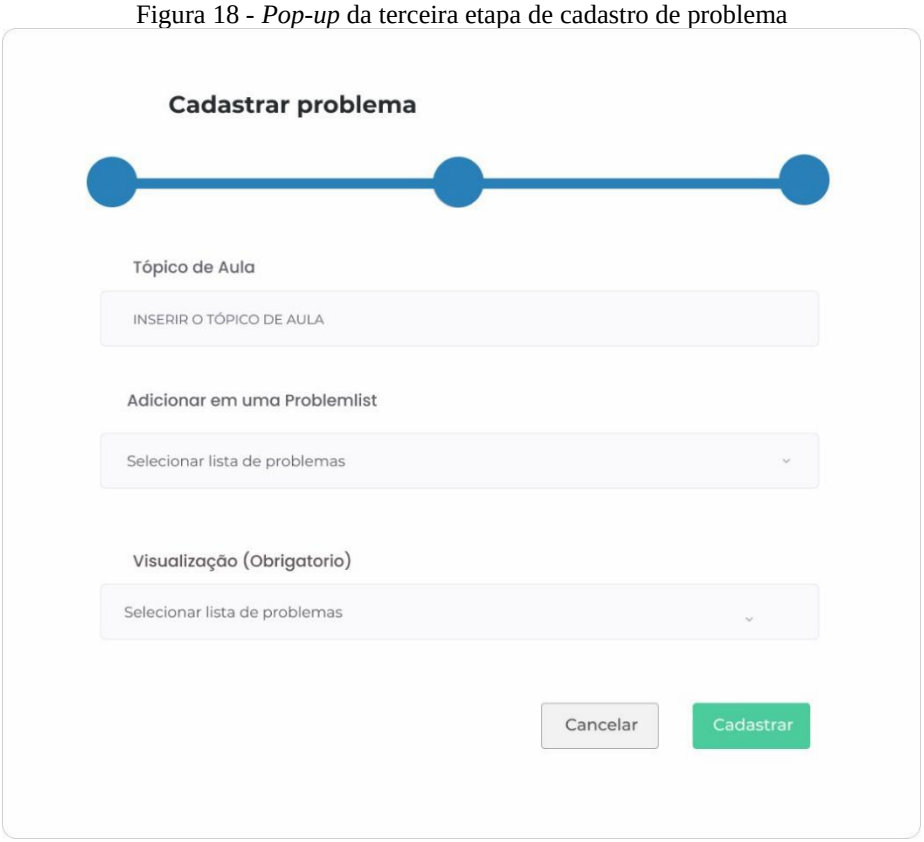
Na última etapa do cadastro, como mostrado na Figura 18, define-se a que tópico de aula o problema se refere, como será sua visualização (público ou privado) e o adicionar ou não a alguma lista de problemas. Finalizado todo o preenchimento, o sistema salvará a questão, após o professor ou monitor clicar em cadastrar.

Figura 17 - *Pop-up* da segunda etapa de cadastro de problema



Fonte: Autoria própria.

Figura 18 - *Pop-up* da terceira etapa de cadastro de problema



Fonte: Autoria própria.

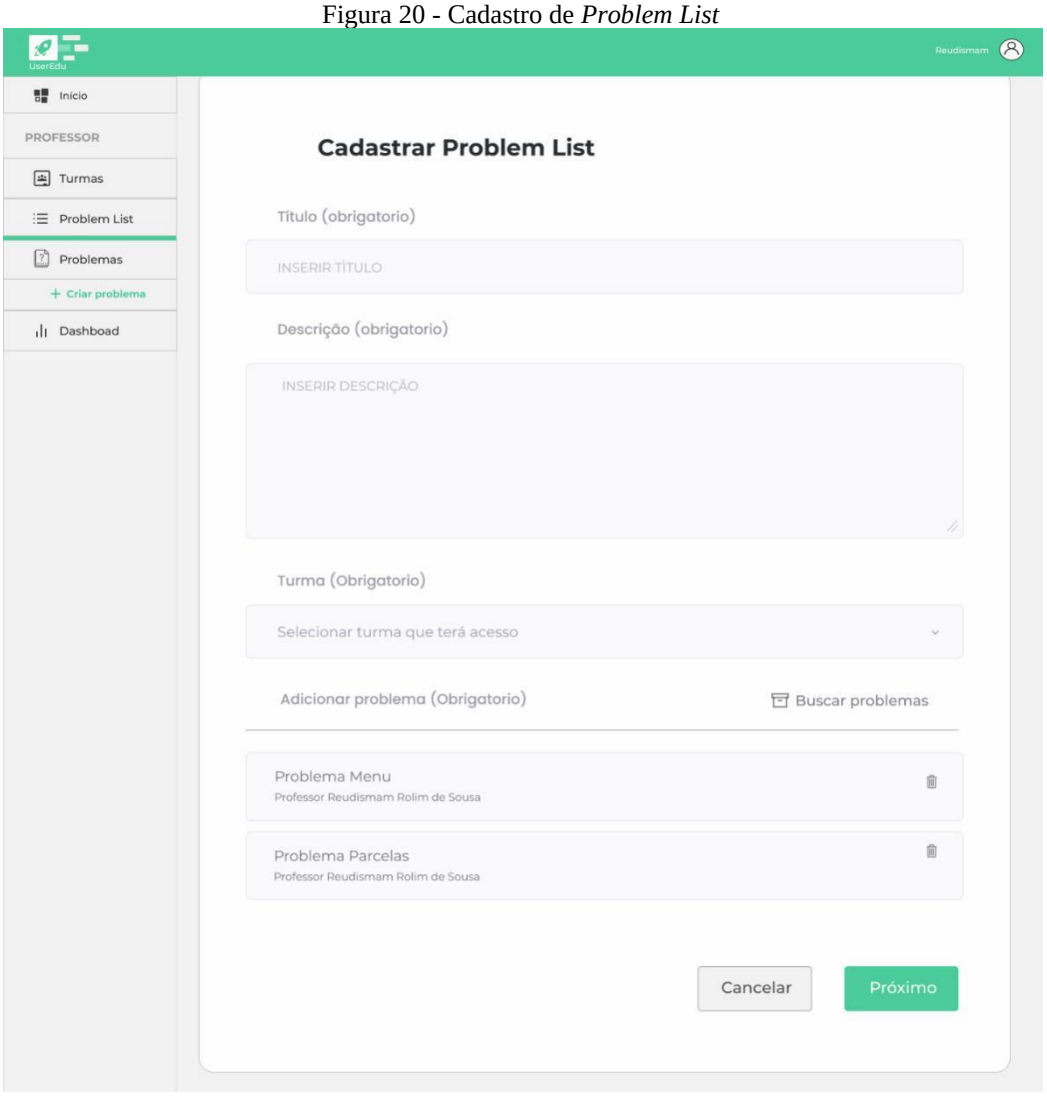
Após todos os cadastros anteriores serem realizados, o professor ou monitor pode realizar o cadastro de listas de problemas, acessando o menu "*Problem list"* no menu lateral, e na tela de listagem das listas de problemas (Figura 19) clicar na opção "nova *problem list"*. Na tela de cadastro das listas, como mostrado na Figura 20, preenche-se o título e a descrição, seleciona-se a turma a qual terá acesso a lista e adiciona-se os problemas que irão compor a lista; por fim, clica-se em cadastrar.

Criada a lista de problemas, o próximo passo é disponibilizar a lista para os alunos resolverem. Neste processo, é criado um registro da lista de problemas para cada aluno matriculado na turma. A disponibilização da lista é feita acessando a tela de listagem (Figura 19), clicando no botão publicar, que fica ao lado do botão alterar. Ao concluir o processo, o botão publicar e editar será removido e inserida a informação de a quanto tempo a lista foi publicada.

Figura 19 - Listas de problemas (modo professor)

Fonte: Autoria própria.

Figura 20 - Cadastro de *Problem List*



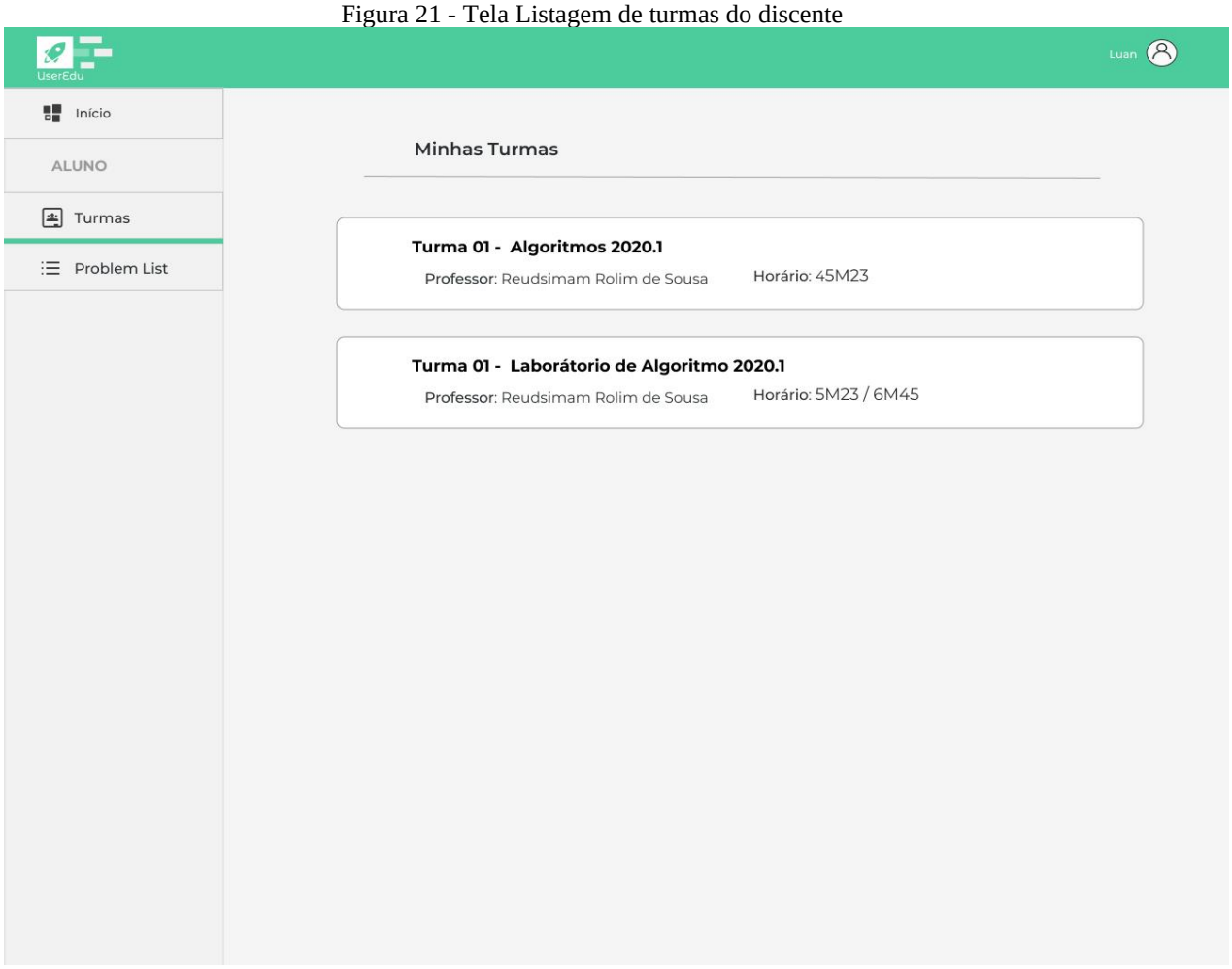
Fonte: Autoria própria.

O aluno ao acessar o sistema, visualiza na tela inicial do sistema em destaque, as principais listas de problemas de programação e as listas que estão sendo trabalhadas, como mostrado na tela da Figura 9. As suas opções, disponíveis no sistema, são apenas duas, o menu "Turma" e o menu "*Problem List"*.

Quando o menu turma é acessado pelo aluno, será carregado a tela com a lista e informações das turmas a que ele está vinculado, como pode-se ver na Figura 21.

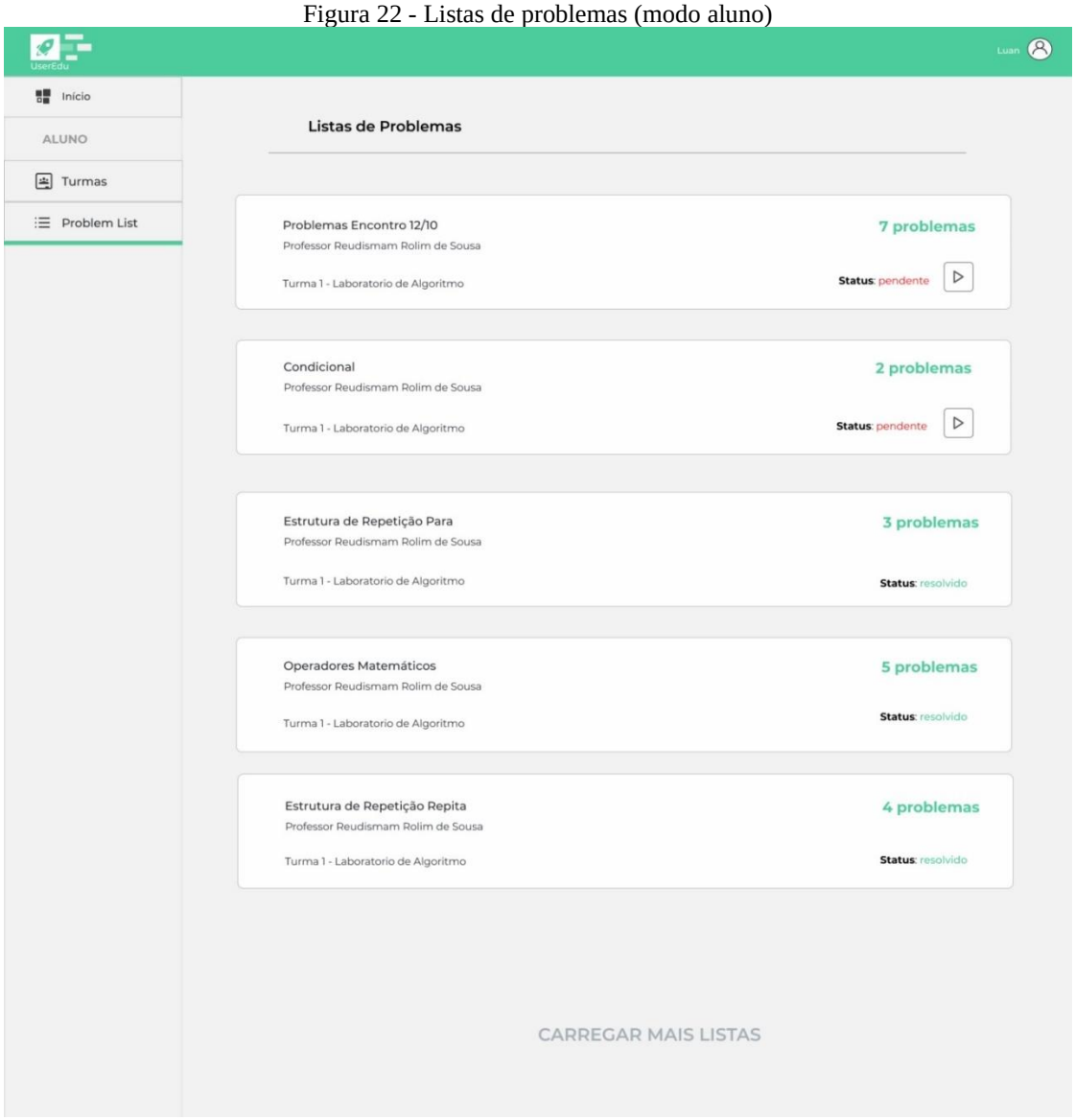
Ao clicar no menu "*Problem List"*, será disponibilizada todas as listas de problemas, a serem respondidas e as já respondidas, que foram disponibilizadas pelos professores nas turmas que o discente está matriculado (Figura 22).

Figura 21 - Tela Listagem de turmas do discente



Fonte: Autoria própria.

Figura 22 - Listas de problemas (modo aluno)

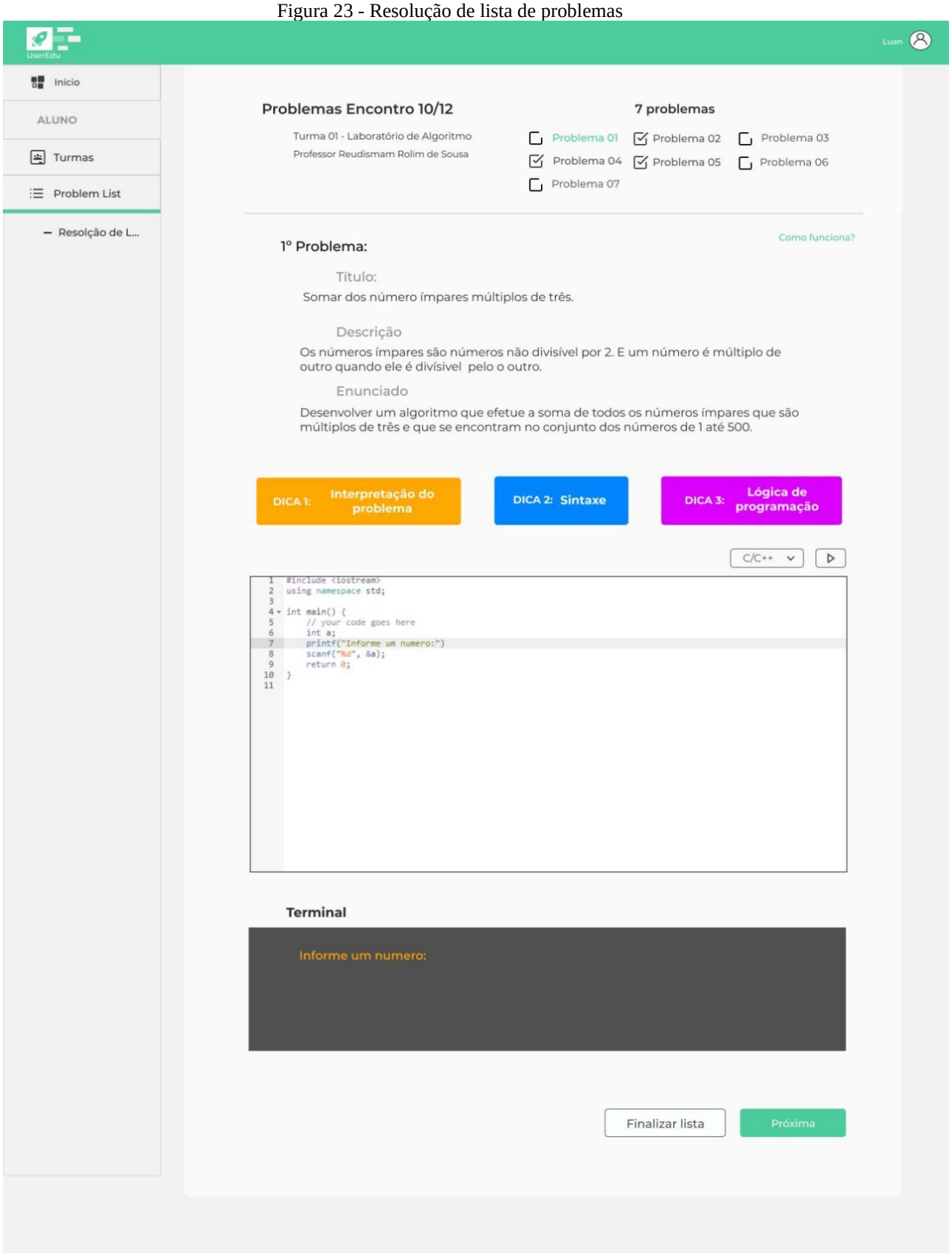


Fonte: Autoria própria.

O aluno pode iniciar a resolução da lista ao clicar no botão “iniciar” ao lado do *status*; feito isso, o aluno será direcionado para uma tela em que serão apresentadas as informações da lista e o primeiro problema, incluindo um editor de código integrado de um compilador com terminal para o aluno testar sua implementação, como mostrado na Figura 23.

Outra característica da tela de resolução da lista e principal função do sistema, é fornecer ajuda para o aluno durante a atividade. Estão disponíveis três botões das dicas disponibilizadas pelos docentes, quando o discente clica em cada uma delas será mostrada a dica solicitada. A ajuda pode ser solicitada apenas depois de um determinado tempo pré-definido de que iniciar o problema, para fazer com que o discente leia atentamente a descrição e o enunciado do problema, e assim interprete e compreenda o mesmo por si só, não indo direto visualizar a ajuda das dicas. A definição desse tempo fica a critério do docente.

Figura 23 - Resolução de lista de problemas



Fonte: Autoria própria

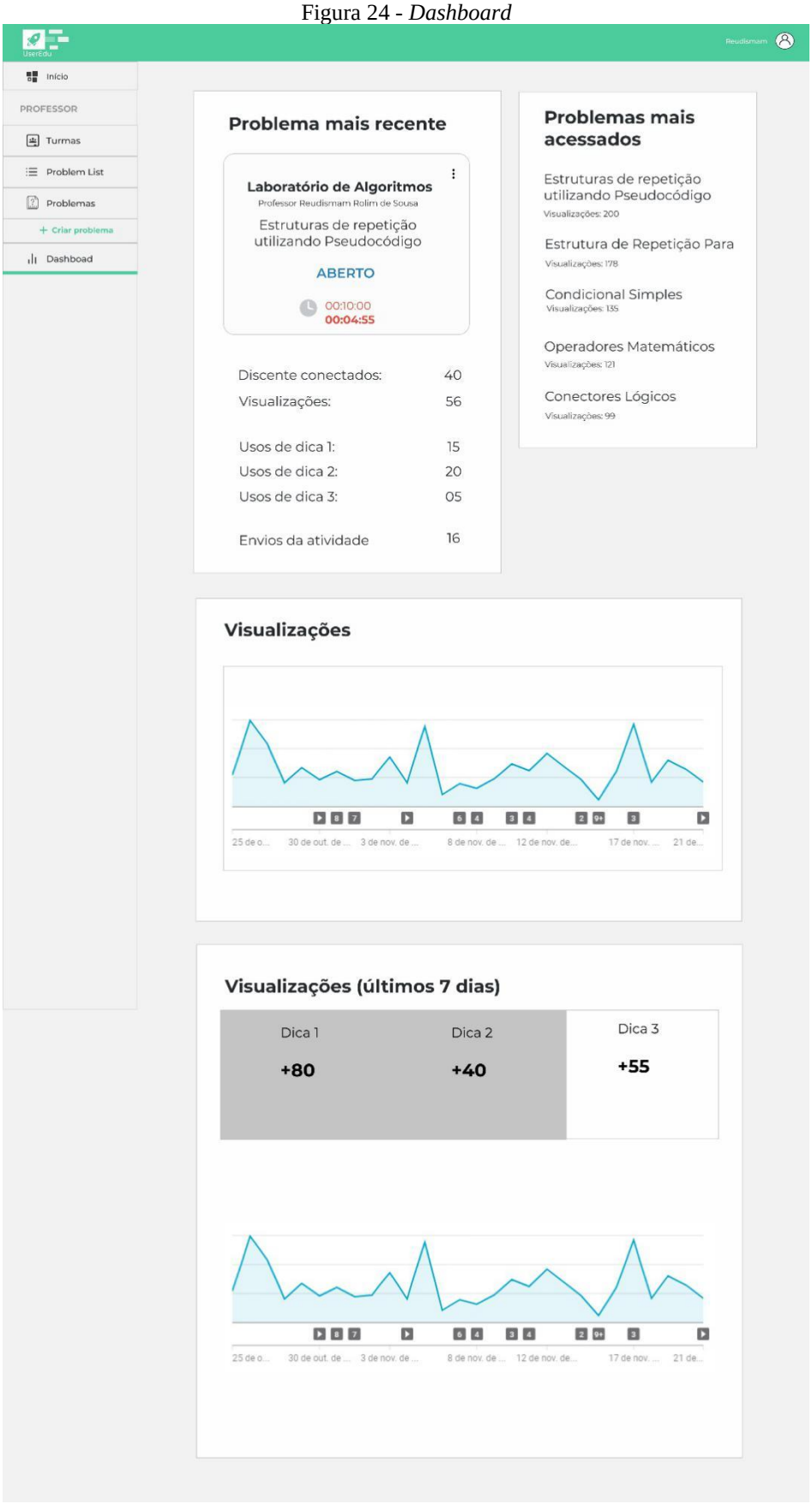
Finalizada uma questão, a próxima será carregada na tela, caso tenha mais de uma, ao clicar no botão “Próxima”. Na tela de resolução da lista também se encontra o botão “Finalizar lista”, que deve ser acionado quando o aluno terminar de implementar as soluções para cada problema; feita essa ação um registro da lista de cada aluno com suas implementações será salvo no sistema.

Ao clicar em *dashboard*, o monitor ou professor será redirecionado para a página da Figura 24. Nela, ele terá à sua disposição informações sobre o número de discentes conectados, visualizações, sobre o consumo e resolução dos problemas ao longo da aula, o uso de dicas (por exemplo, quais dicas estão sendo mais utilizadas, número de acesso a determinada dica, entre outras), dentre outras informações. O sistema apresentará informações sobre a lista de problemas que está sendo consumida no momento (ou sobre o problema, caso este seja único na lista de problemas). Todas essas informações apresentadas são correspondentes a cada turma, para que o ensino/aprendizagem seja focado em aspectos específicos desta e não em dados estatísticos de turmas passadas. As dificuldades dos discentes podem variar ao longo de turmas diferentes, principalmente, em turma de períodos referentes à discentes de primeira ou segunda chamada (por exemplo, 2020.1 e 2020.2).

As informações, na tela de *Dashboard* atuam de forma estratégica para guiar o ensino/aprendizagem do discente, de forma que o docente possa perceber e focar nos aspectos que os discentes apresentam maior dificuldade ou que trarão um maior ganho para o entendimento de como resolver problemas de programação de uma forma geral.

O professor ou monitor poderá usar essas informações de diferentes formas, por exemplo, ao verifica que a dica de sintaxe está sendo bastante utilizada, ele pode iniciar um momento de discussão sobre o assunto associado ao problema durante a aula. Assim, sanando as dúvidas e as dificuldades dos discente no momento que surgem. Essas informações também podem ser utilizadas *offline* pelo docente para que ele possa realizar o planejamento dos conteúdos de aula de forma mais efetiva, direcionado aos pontos mais críticos identificados nas turmas ao consumirem os problemas.

Figura 24 - *Dashboard*



Fonte: Autoria própria.

6 CONCLUSÃO

Neste trabalho foi apresentado o protótipo de um sistema de dicas para auxiliar os discentes em componentes curriculares de programação introdutória. O sistema prototipado busca automatizar o processo de gerenciamento de dicas proposto em Freitas (2020). Essa abordagem busca fornecer dicas de programação a discentes que estão resolvendo problemas de programação. As dicas são divididas de acordo com o tipo (três dicas) e o estudante pode consumi-las ao longo da resolução do problema. Para o projeto do sistema foram utilizadas as etapas do processo de *design* guiada pelo usuário propostas por Garrett (2011). Neste processo de *design*, o projeto do sistema é dividido basicamente em cinco etapas, desde o entendimento das necessidades dos usuários e clientes até o projeto dos detalhes da interface do sistema. Seguindo esse processo, os requisitos do sistema foram elencados e como resultado um conjunto de interfaces foram propostas, o protótipo final do sistema quando implementado busca automatizar a abordagem descrita em Freitas (2020).

Como trabalho futuros, pretende-se realizar a implementação do sistema para que os discentes, docentes e monitores possam interagir com ele. Após implementado, pretende-se realizar uma avaliação do sistema proposto com discentes de disciplinas voltadas à programação introdutória, tais como Algoritmos, e verificar a sua contribuição para a intervenção metodológica de Freitas (2020).

Espera-se que o projeto e protótipo criado possa guiar a implementação desse sistema. Com a implementação desse projeto, e a validação em turmas reais por vários semestres, acredita-se que o sistema possa elevar os resultados obtidos por Freitas (2020) com aplicação da metodologia, e consequentemente auxiliar os discentes a minimizarem os seus problemas com programação e também que os docentes possam ter uma visão geral do desempenho de suas turmas em aspectos da resolução de código e consumo de dicas para que possa intervir, buscando atender aos objetivos dos componentes curriculares. De outra forma, espera-se que os resultados dos estudos possam contribuir para reduzir os insucessos em disciplinas referentes à programação de computadores e reduzir a taxa de evasão.

REFERÊNCIAS

ALMEIDA, Maria Elizabeth Bianconcini de. **Educação a distância na internet**: abordagens e contribuições dos ambientes digitais de aprendizagem. Educação e Pesquisa. São Paulo, v. 29, n. 2, Jul-Dez, 2003.

DICIO. **Dicionário Online de Português**, 2020. Disponivel em: <https://www.dicio.com.b>. Acesso em: 22 out. 2020.

UFERSA, UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO. 2014. **Projeto Pedagógico do Curso de Bacharelado em Tecnologia da Informação**, 2014.

CORMEN, Thomas H. et al. **Algoritmos:** teoria e prática. tradução da segunda edição [americana] Vandenberg D. de Souza. Rio de Janeiro: Elsevier, 2002. 4ª Reimpressão.
DICIO. Dicionário Online de Português, 2020. Disponivel em: <https://www.dicio.com.b>. Acesso em: 22 out. 2020.

FREITAS, Brígido Conrado de Brito. **Uma Intervenção Metodológica para Auxiliar a Aprendizagem de Programação Introdutória da Ufersa, Campus Pau dos Ferros**. Orientador: Profa. Dra. Laysa Mabel de Oliveira Fontes. 2020. Trabalho de Conclusão de Curso (Bacharelado em Tecnologia da Informação) - Universidade Federal Rural do Semi-árido, Campus Pau dos Ferros, Pau dos Ferros, 2020.

GARRETT, J. J. **The Elements of User Experience**: User-Centered Design for the Web and Beyond, Second Edition. Barkeley, CA 94710: Second Edition, 2011.

HELLERHAUS. **The Elements of User Experience de Jesse James Garrett**. Disponível em <hellerhaus.com.br>. Acesso em 01 de dezembro de 2020.

HEMING, Cléverton. **Ferramenta de Apoio ao Ensino e Aprendizagem de Algoritmos e Programação**. Orientador: Prof. Me. Evandro Franzen. 2018. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade do Vale do Taquari - UNIVATES, Lajeado, 2018.

MOREIRA, G. L. et al. **Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros:** um estudo exploratório. In: Anais do III Encontro de Computação do Oeste Potiguar (ECOP). Pau dos Ferros: ECOP'20. 2018. p. 90-96.

PROEC UFABC. Ambientes Virtuais de Aprendizagem. **Site da ProEC UFABC**, 2014. Disponivel em: <http://proec.ufabc.edu.br/uab/index.php/roteiros/roteiro5/19-fteadinicio/fteadaulas/127-aula51>. Acesso em: 11 Novembro 2020.

PEREIRA JÚNIOR, J. C. R.; RAPKIEWICZ, C. E.; DELGADO, C.; XEXEO, J. A. M. Ensino de algoritmos e programação: uma experiência no nível médio. In: **Anais do XIII Workshop de Ensino em Informática (WEI)**, São Leopoldo, 2005.

QUEIROZ, J. V.; RODRIGUES, L. M.; COUTINHO, J. Um relato dos fatores motivacionais na aprendizagem de programação na perspectiva de alunos iniciantes em programação da

universidade federal rural do Semi-Árido campus Pau dos Ferros/RN. **In Proceedings of the III Encontro de Computação do Oeste Potiguar. ECOP '18**, Pau dos Ferros, 29 maio 2018. 90-96.

ROLIM, Reudismam. Motivando os Discentes e Solucionando seus Desafios de Aprendizagem, um Estudo do Projeto de Ensino Pré-Algoritmos. **V Encontro de Computação do Oeste Potiguar**. ECOP'20. 2020.

RODRIGUES JUNIOR, M. C. Como Ensinar Programação? Informática – B**oletim Informativo** Ano I nº 01, ULBRA, Canoas, RS, 2002.

SELIVON, M.; BEZERRA, J.; TONIN, N. **URI Online Julge Academic:** Integração e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem. Workshop Sobre Educação em Computação (WEI). Portalegre: Sociedade Brasileira de Computação. 2015. p. 188-195. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2015.10235.

SIRIUS INTERATIVA. Figma: uma nova ferramenta para design de interface que está ganhando o mercado. **Sirius Interativa**, 2019. Disponivel em: <https://medium.com/@Sirius_/figma-uma-nova-ferramenta-para-design-de-interface-que-est%C3%A1-ganhando-o-mercado-sirius-interativa-2e78e0905b44>. Acesso em: 22 nov. 2020.

SOARES, F. A. L.; CARVALHO, R. B. Proposta de um Portal Educacional para estudantes de programação de computadores. In **Revista Abakós**. v. 5, n. 2, p. 36-58, 2017.

SOUSA, R. R.; LEITE, F. T.; GUIMARÃES, A. O.; and OLIVEIRA, A. R. Pré-algoritmos - Ações de Apoio à Melhoria do Ensino de Graduação. **Brazilian Journal of Development**, 6 (3): 12625–12635. 2020.

STADZISZ, Paulo Cézar. **Projeto de Software usando a UML**. Paraná: Centro Federal de Educação Tecnológica do Paraná, 2002. 69 p.