**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**
**DEPARTAMENTO DE INFORMÁTICA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

**LUCINÉIA BARBOSA DA COSTA CHAGAS**

# UM AMBIENTE PARA APRENDIZAGEM DE PROGRAMAÇÃO

**VITÓRIA**
**2014**

LUCINÉIA BARBOSA DA COSTA CHAGAS

# UM AMBIENTE PARA APRENDIZAGEM DE PROGRAMAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Informática, na área de concentração em Informática na Educação.
Orientador: Prof. D.Sc. Orivaldo de Lira Tavares
Co-orientador: Prof. D.Sc. Crediné Silva de Menezes.

VITÓRIA
2014

# LUCINÉIA BARBOSA DA COSTA CHAGAS

# UM AMBIENTE PARA APRENDIZAGEM DE PROGRAMAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Informática, na área de concentração de Informática na Educação.

Aprovada em 04 de dezembro de 2014.

## COMISSÃO EXAMINADORA

Prof. DSc. Orivaldo de Lira Tavares (orientador)
Universidade Federal do Espírito Santo

Prof. DSc. Crediné Silva de Menezes (co-orientador)
Universidade Federal do Espírito Santo

Prof[a]. DSc. Rosane Aragón (examinadora externa)
Universidade Federal do Rio Grande do Sul

Prof[a]. DSc. Márcia Gonçalves de Oliveira (examinadora externa)
Universidade Federal do Espírito Santo

*"Para realizar grandes conquistas,*
*devemos não apenas agir,*
*mas também sonhar;*
*não apenas planejar,*
*mas também acreditar"*
***Anatole France***

# AGRADECIMENTOS

*“Quero, um dia, poder dizer às pessoas que nada foi em vão…*
*[...] que a vida é bela sim, e que eu sempre dei o melhor de mim… e que valeu a pena.”*

***Mário Quintana***
*“Certezas”*

Agradeço a Deus pelas conquistas alcançadas. A Ele toda honra e glória, para todo o sempre. Agradeço aos meus pais Jovenita dos Santos Costa e Lindolfo Barbosa da Costa, pelo esforço e apoio dedicado desde minha infância. Vocês sem dúvidas são o meu alicerce, minha base, estrutura e o meu maior exemplo de vida.

Sou profundamente grata a eles pelo amor e cuidado em todos os momentos da minha vida. Agradeço em especial ao meu esposo Márcio Gomes Chagas e meu filho Gabriel da Costa Chagas, pelo amor, carinho e compreensão. Vocês são meu porto seguro em todos os momentos. Amo, vocês!

Agradeço ao Professor Orivaldo que me deu a oportunidade de dar continuidade aos estudos, dando-me a honra de por ele ser orientada. Admiro-o muito pela sua sabedoria, excelência e simplicidade.

Agradeço imensamente a CAPES pelo financiamento de minhas pesquisas no mestrado durante esses dois anos. Agradeço a todos os que foram meus professores durante este curso, pois neste trabalho há uma contribuição expressiva de cada um de vocês.

Agradeço a duas pessoas mais que especiais pra mim: Jackson Schwanz Groner e Mário Correa Junior, a conquista deste trabalho também é de vocês. Não posso deixar de agradecer a dois grandes amigos que tive a oportunidade de conhecer e que foram fundamentais na minha vida acadêmica, que são os professores Vanderson Ildefonso Silva e a professora Márcia Gonçalves de Oliveira. Obrigado pelo incentivo e por me mostrar que sou capaz de ir muito além de que posso imaginar. Vocês moram no meu coração!

*Luciméia B. Costa Chagas.*

Vitória, 04 de dezembro de 2014.

*“O temor ao senhor é o princípio de toda sabedoria; Todos o que cumprem os seus preceitos revelam bom senso. Ele será louvado para sempre.”*

***Salmos: 111:10***
*“Bíblia sagrada”*

# RESUMO

A aprendizagem de programação de computadores tem sido objeto de estudo de vários grupos de pesquisa em todo o mundo. Os fatores que contribuem para que a aprendizagem de programação continue a ser um problema complexo estão relacionados às múltiplas habilidades e às múltiplas etapas do processo de programação. No ponto de vista de muitos estudantes, os cursos introdutórios de programação são considerados difíceis, no que resulta em baixo desempenho e muitas reprovações.

Nesta pesquisa, interessamo-nos pela criação de um ambiente de apoio à aprendizagem de programação, em suas várias etapas. Desse modo definimos um ambiente computacional composto de recursos computacionais para suporte a implementação de várias arquiteturas úteis para a aprendizagem de programação.

Apresentamos os elementos que compõem essas arquiteturas (objetivos, estratégias e recursos digitais) e as validações delas realizadas por aprendizes de programação.

# ABSTRACT

Several research groups around the world have studied the process of learning how to program a computer. The learning of programming continues to be a complex problem because of the multiple skills and multiple stages required to the programming process.

In the view of many students, introductory programming courses are considered difficult, this results in poor performance and many failures. In this research, we are interested in the creation of a supportive environment for learning programming in its several stages.

Thus we define a virtual environment with computing resources to support the implementation of pedagogical architectures useful for learning programming. We present the elements of these architectures (objectives, strategies and digital resources) and their validations based on the apprentices' experiences.

# LISTAS DE FIGURAS

# LISTA DE TABELAS

# SUMÁRIO

# 1 INTRODUÇÃO

A aprendizagem de programação de computadores é estudada por vários pesquisadores em todo o mundo. Todavia, mesmo com vários estudos feitos, a aprendizagem de programação continua a ser um grande desafio (TAVARES, 2013).

Os fatores que contribuem para a dificuldade de aprendizagem estão baseados nas múltiplas habilidades e múltiplos processos necessários na programação (GOVENDER, 2009). No ponto de vista de muitos estudantes, os cursos introdutórios de programação são considerados difíceis, no que resulta em baixo desempenho e muitas reprovações (MOONS, 2013).

Tavares *et alii* (2012) relatam que após tentativas frustradas de construir programas para resolver problemas, os alunos reiniciam suas ações em busca de uma solução correta. Desta forma, tentam descobrir e corrigir os erros cometidos inicialmente em um processo de tentativa e erro.

Ao ganhar experiência com o problema, os alunos tomam consciência sobre o mesmo e sobre a construção de soluções corretas. Nesse processo eles corroboram as observações de Piaget sobre a aprendizagem (PIAGET,1978). Ela acontece em um processo de tomada de consciência que inicia com etapas de tentativas e erros (fazer), para depois levar à reflexão sobre as tentativas, erros e acertos e, ao final, chegar à compreensão sobre o problema/tema a ser aprendido.

Em termos pedagógicos, o aprendizado de programação é benéfico para o desenvolvimento de habilidades cognitivas. Didaticamente a aprendizagem de programação permite aos alunos o desenvolvimento de conhecimentos em diversas áreas relacionadas a cursos voltados para área de informática (FESSAKIS *et alii*, 2013).

O professor como ator do processo de ensino-aprendizagem precisa estar ciente de seu papel, e é necessário que ele possua a percepção do que o aluno não consegue assimilar, e que para ensinar é necessário refletir sobre sua prática e refazer suas

metodologias para que possa atingir o seu objetivo maior, que é o de fazer com que o aluno aprenda (VALENTIM, 2009).

Na busca de propor melhorias no processo de ensino-aprendizagem de programação, é apresentado neste trabalho, um ambiente para aprendizagem de programação no intuito de auxiliar os professores no processo de ensino-aprendizagem. A contribuição deste trabalho está em propor um ambiente de aprendizagem de programação que dê suporte as arquiteturas pedagógicas: *CSPL*, Programação em Pares e Recomendação de exercícios, com o objetivo de auxiliar professores e alunos no processo de aprendizagem.

## 1.1 Referencial teórico

A disciplina relacionada à programação de computadores é uma das principais razões pelas quais há evasão e reprovação nas primeiras fases dos cursos de informática, da maioria das escolas de ensino técnico ou superior (ASSIS, 2008; CABRAL, 2007).

Para que um aluno de programação se torne um bom programador, este precisa possuir: competência (um conjunto de saberes), e habilidades (um saber-fazer relacionado à prática) para resolução de problemas, pouco exercitadas no ensino básico.

Perrenoud (1999, p. 07), define competência como “uma capacidade de agir eficazmente em um determinado tipo de situação, apoiada em conhecimentos, mas sem limitar-se a eles”. As competências não se formam com a assimilação de conhecimentos, mas sim com a construção de um conjunto de disposições e esquemas que permitem mobilizar os conhecimentos na situação, no momento certo e com discernimento (PERRENOUD, 1999).

Dentre as principais habilidades e competências citadas pelos autores estão: resolução de problemas (KAZIMOGLU, 2012; ATER-KRANOV, 2010), habilidades cognitivas (CASEY, G. CEGIELSKI & DIANE, J. HAL, 2006); raciocínio lógico (QUALLS & SHERRELL, 2011; WING, 2006); resolver problemas de forma

colaborativa (CÁMARA, 2013); desenvolver várias soluções para o mesmo problema (TAVARES, 2013); trabalho em equipe e espírito crítico (WANG e LIN, 2007); codificar algoritmos em linguagens de programação (LAU, 2011; KHVILON & PATRU, 2002; COHEN e HABERMAN, 2007), dentre outras.

Para amenizar essas e outras dificuldades dos alunos, propomos o uso de um ambiente de aprendizagem com uso de Arquiteturas Pedagógicas especialmente desenvolvidas para a aprendizagem de programação, sendo aplicáveis ao desenvolvimento das habilidades necessárias para cada uma das etapas do processo de programar, conforme proposto por Tavares *et alii* (2013).

Carvalho *et alli* (2005) afirmam que as arquiteturas pedagógicas são, antes de tudo, estruturas de aprendizagem realizadas a partir da confluência de diferentes componentes: abordagem pedagógica, *software*, internet, inteligência artificial, educação a distância, concepção de tempo e espaço.

Tem como caráter pensar na aprendizagem como um trabalho artesanal, construído na vivência de experiências e na demanda de ação, interação e meta-reflexão do sujeito sobre os fatos, os objetos e o meio ambiente sócio-ecológico (Kerckhove, 2003).

O objetivo das arquiteturas pedagógicas está em filtrar informações que mostrem ao aluno seus pontos de dificuldades e neles intervenha com avaliações diagnósticas e formativas no tratamento dessas deficiências, para proporcionar melhorias no processo de aprendizagem.

Esta dissertação propõe um ambiente de aprendizagem que dê suporte a várias arquiteturas pedagógicas dotadas de recursos computacionais especialmente desenvolvidos para apoiar a formação de programadores habilidosos e competentes.

## 1.2 Problematização

A disciplina de programação de computadores é uma das disciplinas com maior índice de reprovação nas Instituições de Ensino Superior (*IES*). Por este motivo, o

seu ensino é considerado um dos sete grandes desafios do ensino de computação do século (CAMPOS, 2009).

O nível de insucesso dos estudantes nas disciplinas de programação é elevado e alvo de várias pesquisas. Dentre as dificuldades apresentadas pelos alunos no processo de programar, estão: o baixo nível de abstração, a falta de competências na resolução de problemas e a inadequação dos métodos pedagógicos aos estilos de aprendizagem (JENKINS 2002). Souza (2012) enfatiza dificuldades na compreensão dos conceitos de programação e visões erradas sobre a atividade de programar.

Acrescenta-se a essa lista: erros de sintaxe e semântica; dificuldades na compreensão do enunciado dos problemas e na concepção de algoritmos; incapacidade de detectar erros de lógica de programação, conforme GOMES e MENDES (2000); dificuldades relacionadas à capacidade de abstração; dificuldades impostas pela sintaxe e estruturas abstratas da linguagem de programação, conforme RIBEIRO (2012).

Tavares *et.alli* (2012) relatam que os alunos ingressantes trazem consigo bases diferenciadas de conhecimentos, que evidentemente influenciam nos rendimentos deles ao longo do curso. Para se ajustar ao nível de exigência dos cursos superiores de computação, o aluno precisaria passar por um treino intensivo em resolução de problemas, de modo a desenvolver competências de diversas áreas para obtenção de um retorno mínimo na aprendizagem (DIJKSTRA,1989)e (PERKINS *et alli*. 1988).

Todos esses e outros fatores levam o aluno que têm pouco ou nenhum conhecimento prévio a abandonar o curso, pois ele sente a necessidade de um acompanhamento personalizado que o professor nem sempre disponibiliza e que os métodos de ensino tradicionais nem sempre dispõem (TAVARES, 2013).

Em geral, as dificuldades no aprendizado de programação não são tão fáceis de serem solucionadas, porém podem ser amenizadas com melhorias pedagógicas e maneiras interativas e intuitivas de ensinar e avaliar. Uma dessas formas pode envolver a utilização de arquiteturas pedagógicas que auxilie no processo de aprendizagem.

Os problemas mencionados, além de dificultar o ensino e a aprendizagem de programação de computadores, reduzem a carga de exercícios, quando o ofício de programar exige muita prática. Hoje, poucos exercícios de programação são realizados sob a supervisão de um professor, o que consideramos preocupante devido a importância desse acompanhamento.

### 1.2.1 Questões norteadoras

A pesquisa científica visa encontrar respostas para as indagações propostas. Logo, as ações da pesquisa concentram-se em descobrir as respostas para as ditas indagações (SILVA E MENEZES, 2001). A seguir, estão listadas as principais questões que foram investigadas por esta pesquisa:

i. Quais as dificuldades encontradas pelos alunos no processo de programar?
ii. Quais as habilidades que um aluno precisa para se tornar um bom programador?
iii. Quais as abordagens usadas para ensinar programação de computadores?
iv. Como auxiliar professores e alunos no processo de ensino aprendizagem de programação de computadores?
v. Como uma arquitetura pedagógica, pode auxiliar os professores no ensino-aprendizagem de programação?
vi. Que recursos computacionais podem instrumentalizar arquiteturas pedagógicas apropriadas para ajudarem na aprendizagem de programação.

## 1.3 Motivação

As teorias de ensino-aprendizagem buscam compreender como se dá a aprendizagem. O interacionismo de *Vigotsky* e de seus seguidores, ao lado de outras teorias como o construtivismo de Piaget, alteraram profundamente o modo de ver todo o processo do ensino e aprendizagem (POZO, 2002).

Aprender nesse contexto “requer assimilar informação, ter conhecimentos, realizar operações, exercitar procedimentos e estratégias para tirar o melhor partido do que se conhece. Conhecer mais, resolver problemas, tomar decisões” (GARCÍA; VEIGA, p.92, 2007).

A maior motivação deste trabalho se dá na possibilidade de promover melhorias significativas no processo de ensino e de aprendizagem de programação, com o uso de tecnologias que potencializem a aprendizagem.

Em resumo, buscamos construir um ambiente com uso de arquiteturas pedagógicas que facilitem o processo de promover, acompanhar e regular a aprendizagem de programação dos alunos, de modo a nivelá-los e dar-lhes condições favoráveis de aprendizagem.

## 1.4 Objetivos

É de vital importância e premência que as instituições de educação se atualizem, adaptem e interajam com as evoluções tecnológicas e estruturais. Quando a programação é olhada pela ótica cognitiva, aqueles que a ensinam sabem que a aprendizagem de programação não é um dom, mas sim um processo complexo que qualquer um pode realizar com sucesso, desde que sejam usados métodos e tecnologias educacionais adequadas.

Este trabalho tem como objetivo propor, construir e validar um ambiente para aprendizagem de programação, com suporte de arquiteturas pedagógicas que melhorem o processo de aprendizagem de programação.

## 1.5 Metodologia

Diante dos problemas apresentados no ensino-aprendizagem de programação de computadores na Seção 1.1, torna-se inadequado manter um modelo pedagógico que preserve as desigualdades de níveis de aprendizagem e não atente para o desenvolvimento de habilidades.

A metodologia usada é alicerçada nos estudos de PIAGET (1978) que evidencia a importância do “fazer” como precursor do “compreender” e da interação como base da aprendizagem.

Outra base teórica adotada são os estudos socioculturais de VIGOTSKY (2001) que vê o processo de aprendizagem como o surgimento de novas formas e novos pensamentos, que são acompanhados pela emergência de novas funções mentais, novas atividades e novos mecanismos de conduta. Essa concepção sugere que a aprendizagem não pode somente seguir o desenvolvimento, mas também superá-lo, projetando-o para frente e suscitando novas formações.

Considerando tais recomendações, esta pesquisa foi desenvolvida em três etapas: *Problema de pesquisa, especificação de solução e avaliação da proposta de solução*. Essas etapas foram orquestradas de modo que os resultados produzidos em uma fase fossem insumos para a etapa seguinte, conforme ilustrado na Figura 01.

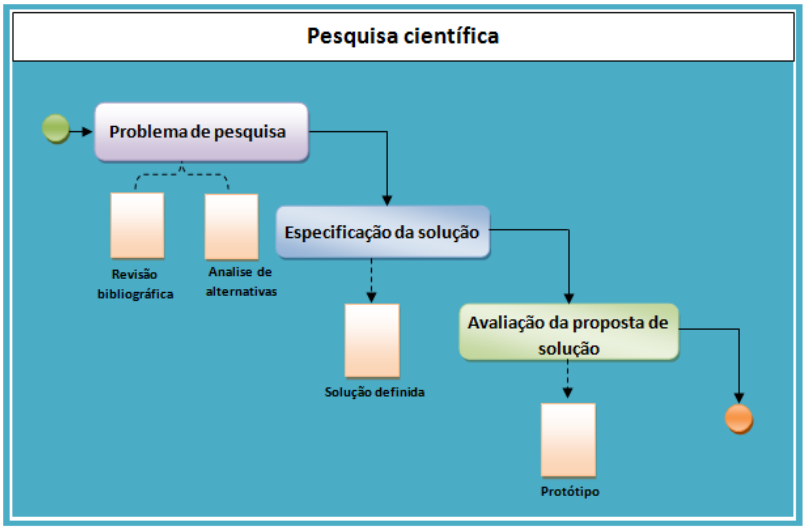


Figura 01. Etapas da Pesquisa Científica

Na etapa *Problema de pesquisa,* definimos o problema a ser pesquisado. Como resultados desta etapa temos uma *revisão bibliográfica e uma análise de alternativas*. Na *revisão bibliográfica*, buscamos abordagens similares e fizemos uma comparação delas, no intuito de produzir um referencial que admitisse à comparação e a análise das melhores *alternativas de solução*. Já a análise de alternativas, constitui-se justamente em analisar as soluções propostas pelos autores.

Na etapa *Especificação da solução,* é proposto como resultado esperado a especificação de uma solução que consiste de um ambiente para aprendizagem de programação que dê suporte às arquiteturas pedagógicas, apoiadas por recursos digitais especialmente desenvolvidos, para apoiarem a aprendizagem ao longo do processo de aprendizagem de programação de computadores.

Para isso, usamos como insumo os resultados produzidos na fase *Problema de pesquisa*, que nos deram suporte na seleção dos métodos e técnicas adequados aos objetivos desta pesquisa. Com a proposta de solução definida na etapa anterior, a etapa *Avaliação da proposta de solução* se constituiu em avaliar a implementação da solução na forma de um protótipo.

## 1.6 Produção Científica

Além desta dissertação, os seguintes artigos foram frutos desta pesquisa:

i. TAVARES, O. L.; VASSOLER, G.; COSTA, L. B .: **Arquitetura pedagógica para ambientação de educadores no uso das TICs na educação presencial.** In: Tise 2012, 2012, Santiago/Chile. Memorias del XVII Congreso Internacional de Informática Educativa, 2012. p. 1-8.

ii. TAVARES, O. L.; MENEZES, C. S.; ARAGON, R.; COSTA, L. B. .**Uma arquitetura pedagógica auxiliada por tecnologias para ensino e aprendizagem de programação**. In: XXXIII Congresso da Sociedade Brasileira de Computação - Cidades inteligentes: desafios para computação, 2013, Maceió/ Alagoas. WEI - XXI workshop sobre educação em computação, 2013.

iii. LIBERATO, A. B. ; SILVA, V. J. I. ; COSTA, L. B**.** ; TAVARES, O. L. ; MENEZES, C. S. . **Ordenador Semântico de Mensagens do Correio Eletrônico**. In: XXXIII Congresso da Sociedade Brasileira de Computação - Cidades inteligentes: desafios para computação, 2013, Maceió/ Alagoas. Desafie - II Workshop de desafios da computação aplicada à Educação, 2013.

iv. TOREZANI, C. ; COSTA, L. B**.** ; TAVARES, O. L. . ***NewProg* - um ambiente online para crianças aprenderem programação de computadores**. In: CBIE- Congresso Brasileiro de Informática na Educação, 2013, São Paulo. WIE, 2013.

v. COSTA, L. B. ; TAVARES, O. L. ; MENEZES, C. S. ; ARAGON, R. . ***Pedagogical architectures and web resources in the teaching-learning of programming***. In: TISE 2013 - XVIII Conferencia Internacional sobre Informática na Educação, 2013, Porto Alegre - RS. Nuevas Ideas en Informática Educativa, 2013. v. 9. p. 430-434.

## 1.7 Estrutura da dissertação

Esta dissertação está estruturada na seguinte forma:

Neste Capítulo, de caráter introdutório, é apresentado uma visão geral da dissertação, com destaque no contexto, nas motivações, no problema e na metodologia de trabalho observada pela pesquisa.

No Capítulo 2, é mostrada uma revisão sistemática sobre a aprendizagem de programação de computadores, na qual são apontados os principais problemas encontrados pelos pesquisadores e propostas de melhorias na aprendizagem de programação.

No Capítulo 3, são estabelecidos as bases para a discussão sobre as arquiteturas pedagógicas, propondo subsídios à compreensão do tema sob a ótica histórica e teórica. Apresenta-se uma análise das abordagens similares à nossa pesquisa, a partir de um conjunto de características, de modo a propiciar o entendimento do estado da arte nesta área de pesquisa, a partir de um conjunto de informações quantitativas.

No Capítulo 4, cerne da pesquisa, é descrito a solução proposta, um ambiente de aprendizagem de programação. Esse ambiente de aprendizagem de programação segue um processo no qual as atividades são coordenadas para a resolução do problema de aprendizagem de programação de computadores.

No Capítulo 5, são relatados os resultados obtidos a partir da avaliação da solução proposta, com o uso de um protótipo computacional especialmente desenvolvido.

No Capítulo 6, apresenta-se as considerações finais e os principais direcionamentos para pesquisas futuras.

# 2 REVISÃO SISTEMÁTICA SOBRE A APRENDIZAGEM DE PROGRAMAÇÃO

Este capítulo apresenta os resultados de uma revisão sistemática de estudos sobre aprendizagem de programação. Esta revisão contou com a análise de sete artigos sobre o processo de aprendizagem de programação de computadores publicados em periódicos internacionais dos últimos seis anos.

Essa revisão teve como foco a identificação de fatores que afetam no desempenho dos alunos e nas habilidades que eles precisam para se tornarem bons programadores, além de como acompanhar o processo de aprendizagem para potencializá-la.

Com o intuito de identificar os fatores que afetam o desempenho dos alunos e apontar as habilidades que eles precisam para se tornarem bons programadores, este capítulo está organizado da seguinte forma: na seção 2.1 é apresentado uma revisão sobre abordagens da aprendizagem de programação registrada em diversos trabalhos examinados; a seção 2.2 aborda a metodologia usada na revisão sistemática; na seção 2.3, são mostrados os resultados obtidos; na seção 2.4 é apresentado uma síntese da revisão sistemática.

## 2.1 Revisão sobre abordagens de aprendizagem de programação

A aprendizagem de programação tem sido objeto de estudo por vários pesquisadores em todo o mundo. Para Kordaki (2010) a programação é mais uma habilidade mental do que um corpo de conhecimento e é uma tarefa complexa, o que inclui a compreensão, codificação, testes e depuração de programas.

Govender (2009) relata que os fatores que contribuem para a dificuldade de aprendizagem na disciplina de programação, estão baseados nas múltiplas habilidades, nos múltiplos processos, e aponta como uma novidade educacional. No ponto de vista de muitos estudantes, os cursos introdutórios de programação são considerados difíceis, no que resulta em baixo desempenho e muitas reprovações (Moons, 2013).

Na busca de compreender e minimizar as várias dificuldades apresentadas pelos alunos no processo de aprendizagem de programação são desenvolvidas diversas experiências com o uso de tecnologias (Hwang, 2008) e de abordagens pedagógicas (Cámara, et.ali, 2013, Chen, et. ali. 2012; Astrachan & Reed,1995; Kölling & Barnes, 2004, Kalles, 2008).

Todavia, mesmo com vários estudos feitos, a aprendizagem de programação continua ser um grande desafio. Na busca de respostas para os questionamentos em relação à aprendizagem de programação, é apresentada uma Revisão Sistemática (*RS*) focada na aprendizagem de programação de computadores.

## 2.2 Metodologia

As revisões sistemáticas são particularmente úteis para integrar as informações de um conjunto de estudos realizados separadamente sobre determinada intervenção, que podem apresentar resultados conflitantes e/ou coincidentes, bem como identificar temas que necessitam de evidência, no auxilio a orientação para investigações futuras. Sampaio (2007) define revisão sistemática como uma forma de pesquisa que utiliza como fonte de dados, a literatura sobre determinado tema.

Para Kitchenham (2007), as *RSs* são realizadas para identificar, avaliar e interpretar estudos que estejam disponíveis e que sejam relevantes a uma determinada questão de pesquisa. Ela pode ser realizada com os objetivos de

identificar lacunas existentes em uma área de pesquisa ou fornecer um conjunto relevante de trabalhos relacionados para embasar novas pesquisas.

Para realizar esta pesquisa, foi feito um levantamento de artigos, no Portal de Periódicos da Capes, entre os dias 03 de julho ao dia 10 de setembro de 2013, com o uso da chave de busca*: learning programming*. Os artigos apresentados foram escolhidos de acordo com os critérios de inclusão e exclusão descritos na Seção 2.2.1.

A pesquisa realizada contou com 60 artigos pré–selecionados, nos quais apenas sete foram incluídos para a extração de dados. A busca pelos artigos foi feita em duas etapas: *Pré-seleção e Análise*. A *Pré-seleção de artigos* consistiu no levantamento de artigos sobre aprendizagem de programação e a etapa *análise* consistiu na aplicação de critérios de inclusão e exclusão de artigos.

### 2.2.1Questões de pesquisa

Nesta seção, é usada a seguinte questão central de pesquisa: “Como a aprendizagem de programação é vista por pesquisadores internacionais?”. Para responder a essa questão, foram definidas quatro questões específicas, conforme mostra a Tabela 01.

Tabela 01: Questões de Pesquisa

| **Questões de pesquisa** | |
|---|---|
| **Q1:** | Quais as dificuldades encontradas pelos alunos no processo de programar? |
| **Q2:** | Quais as abordagens usadas para ensinar programação de computadores? |

| | |
|---|---|
| **Q3:** | Que artigos abordam o problema com o uso de técnicas de recomendação de problemas, recomendação de soluções; *feedback* automático ou regulação; mapas conceituais ou qualquer recurso digital (algoritmo ou técnica) para apoiar o ensino-aprendizagem de programação? |
| **Q4:** | Quais os artigos que fazem abordagens únicas (diferentes das demais)? |

As questões de pesquisas apontadas na Tabela 01, foram escolhidas no intuito de buscar na literatura atual, os principais pontos que influenciam a aprendizagem de programação, o que se tem feito para propor melhorias além de responder a questão central desta pesquisa.

A Questão 1 mostrada na Tabela 01, *quais as dificuldades encontradas pelos alunos no processo de programar?* Tem como objetivo encontrar as dificuldades apresentadas pelos estudantes durante a aprendizagem de programação.

A Questão 2 *quais as abordagens usadas para ensinar programação de computadores?* Tem como objetivo buscar as abordagens usadas em cada artigo para trabalhar a aprendizagem de programação.

Na Questão 3 *que artigos abordam o problema com o uso de técnicas de recomendação de problemas, recomendação de soluções; feedback automático ou regulação; mapas conceituais ou qualquer recurso digital (algoritmo ou técnica) para apoiar o ensino-aprendizagem de programação?* Tem como objetivo verificar quais as técnicas usadas pelos pesquisadores para propor melhorias no aprendizado.

Na quarta e última questão *Quais os artigos que fazem abordagens únicas (diferentes das demais)?* Buscou-se pesquisar quais abordagens únicas foram usadas na aprendizagem de programação. Os resultados das questões norteadoras são apresentados na sessão 2.3.

### 2.2.2 Levantamento inicial dos artigos

O levantamento de artigos foi realizado por meio do Portal de Periódicos da Capes, com a chave de *busca learning programming.* Num primeiro momento, vários artigos foram mostrados, porém nem todos foram escolhidos, por não estarem dentro dos critérios pré-estabelecidos. Os critérios pré-estabelecidos para inclusão e exclusão de artigos são apresentados na Tabela 02.

Tabela 02: Critérios de inclusão e exclusão de artigos

| Inclusão de artigos | Exclusão de artigos |
|---|---|
| **Artigos completos com abordagem focada no processo de aprendizagem de programação para iniciantes.** | Artigos com abordagem de programação como ferramenta. |
| **Artigos que sejam publicados nos últimos seis anos** | Artigos que não sejam claramente sobre o processo de ensino-aprendizagem de programação para iniciantes. |
| | Artigos que não sejam publicados nos últimos seis anos. |

Os critérios de inclusão e exclusão de artigos foram escolhidos com o objetivo de responder a questão central da revisão sistemática. Neste propósito a inclusão de artigos se deu a todo artigo que fosse completo com abordagem focada na aprendizagem de programação para iniciantes e que fossem publicados nos últimos 6 anos.

Na exclusão de artigos, os critérios *Artigos com abordagem de programação como ferramenta, Artigos que não sejam claramente sobre o processo de*

*ensino-aprendizagem de programação para iniciantes* e *Artigos que não sejam publicados nos últimos seis anos*. Foram usados com intuito de excluir os artigos que não estavam enquadrados nos critérios de inclusão.

A seleção dos artigos foi executada em duas etapas: *Pré-seleção e Análise*. A *Pré-seleção* consistiu-se em levantar artigos sobre aprendizagem de programação em periódicos que fossem publicados nos últimos 6 anos. Enquanto que na etapa *análise* tinha como objetivo fazer a aplicação dos critérios de inclusão e exclusão de artigos.

A aplicação desses critérios determinou a permanência ou a exclusão de cada artigo na revisão. Após a aplicação dos critérios, os artigos selecionados tiveram seus dados extraídos para fornecimento de respostas para a questão de pesquisa e sistematizados para representação dos resultados.

## 2.3 Resultados

A pesquisa resultou em 82 artigos, dos quais 60 foram pré-selecionados. Após a etapa de pré-seleção, os artigos passaram por uma análise feita por meio da leitura de cada um deles, com base nos critérios de inclusão e exclusão, apresentados na seção 2.2.2. A Tabela 03 apresenta os resultados gerais dos processos de pré-seleção e inclusão dos artigos.

Tabela 03. Resultado geral da pesquisa

| | **Artigos pré-selecionados** | **Artigos incluídos** | **Total de artigos completos** |
|---|---|---|---|
| **Artigos** | 60 | 7 | 82 |
| **Total** | 60 | 7 | 82 |

Dos 60 artigos pré-selecionados apenas sete foram incluídos na revisão sistemática, são eles:

- ✓ Govender, I. (2009). ***The learning context: Influence on learning to program****.Computers & Education*, *53*(4), 1218-1230.
- ✓ Hwang, W. Y., Wang, C. Y., Hwang, G. J., Huang, Y. M., & Huang, S. (2008). ***A web-based programming learning environment to support cognitive development. Interacting with Computers***, *20*(6), 524-534.
- ✓ Hwang, Wu-Yuin, et al. ***"A pilot study of cooperative programming learning behavior and its relationship with students' learning performance."*** *Computers & Education* 58.4 (2012): 1267-1281.
- ✓ Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). ***Learning Programming at the Computational Thinking Level via Digital Game-Play.Procedia*** *Computer Science*, *9*, 522-531.
- ✓ Law, K. M., Lee, V., & Yu, Y. T. (2010). ***Learning motivation in e-learning facilitated computer programming courses***. *Computers & Education*, *55*(1), 218-228.
- ✓ Moons, J.; Backer, C. D.: ***The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.*** In: Computers & education 60 (2013) 368-384.
- ✓ Serrano-Cámara, L. M. ; Paredes-Velasco, M. ; Alcover, C. M. ; Velazquez,I., J. Ángel. ***An evaluation of students' motivation in computer-supported collaborative learning of programming concepts*** In: Computers in Human Behavior, 2013.

### 2.3.1. Distribuição temporal dos artigos incluídos na RS

A Tabela 4 apresenta informações referentes à Distribuição Temporal e a quantidade dos artigos incluídos nesta revisão sistemática.

Tabela 04 – Distribuição Temporal

| Ano | Artigo |
|---|---|
| **2008** | 1 |
| **2009** | 1 |
| **2010** | 1 |
| **2012** | 2 |
| **2013** | 2 |

Os dados apresentados na Tabela 04 estão apresentados de acordo com o ano e a quantidade de artigos encontrados sobre a aprendizagem de programação. Dos sete artigos selecionados, dois foram publicados no ano de 2012 e dois em 2013, o que equivale a 58% dos artigos selecionados.

A distribuição temporal dos artigos mostra que nos últimos anos, tem crescido esforços para propor melhorias na aprendizagem de programação. Todavia essa aprendizagem ainda tem sido um grande desafio para os educadores, tanto nacionais como internacionais.

### 2.3.2 Nível de escolaridade encontrado nas pesquisas

A classificação dos artigos se deu de acordo com o nível de escolaridade para os quais as pesquisas se destinam. Quanto aos níveis de escolaridade apresentados, foi verificado que 100% dos trabalhos selecionados abordaram

estudos no contexto da educação superior. Mostra-se claramente a carência de pesquisas que abordem o processo de ensino-aprendizagem de programação para alunos iniciantes de cursos em níveis fundamental, médio e técnico. O que torna um assunto propício para o desenvolvimento de novas pesquisas.

### 2.3.3 Principais problemas na aprendizagem de programação

Nesta revisão, buscou-se encontrar os principais problemas abordados na aprendizagem de programação, os resultados são apresentados na Tabela 05.

Tabela 05. Principais problemas/assuntos tratados nos artigos

| Problema/assunto | Artigos |
|---|---|
| **Questões relacionadas à aptidão** | *The learning context: Influence on learning to program.* |
| **Fatores cognitivos** | - *The learning context: Influence on learning to program;*<br>- *Learning Programming at the Computational Thinking Level via Digital Game-Play;*<br>- *A web-based programming learning environment to support cognitive development;*<br>- *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.* |
| **Estilos de aprendizagem** | - *The learning context: Influence on learning to program.*<br>- *A pilot study of cooperative programming learning behavior and its relationship with students' learning* |

| | |
|---|---|
| | *performance;*<br>- *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.* |
| **Motivação** | - *The learning context: Influence on learning to program;*<br>- *Learning motivation in e-learning facilitated computer programming courses;*<br>- An evaluation of students' motivation in computer-supported collaborative learning of programming concepts; |
| **Resolução de problemas** | *A web-based programming learning environment to support cognitive development.* |
| **Compreensão do problema** | *A web-based programming learning environment to support cognitive development.* |
| **Codificação da solução do problema** | *A web-based programming learning environment to support cognitive development* |
| **Planejamento para projetar programas** | - *A web-based programming learning environment to support cognitive development.*<br>- *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.* |

A Tabela 05 mostra os principais problemas encontrados na aprendizagem de programação. A tabela foi divida em duas colunas, nas quais a coluna *Problema/assunto* aborta os principais problemas e assuntos mencionados pelos autores no processo de aprendizagem e na coluna *Artigos* é mostrado os artigos abordam tais problemas.

### 2.3.4 Abordagens usadas

Nobre *et alli* (2013) enfatizam que ao considerar a intencionalidade de toda ação educativa exercida por professores, em situações planejadas de ensino aprendizagem, uma questão importante permeia as preocupações básicas dos educadores: as várias formas de se conceber o fenômeno educativo. A Tabela 06 mostra as abordagens pedagógicas usadas nos artigos.

Tabela 06. Abordagens usadas

| Abordagens | Artigos |
|---|---|
| **Construtivista** | *The learning context: Influence on learning to program.* |
| **Construtivista** | *Learning motivation in e-learning facilitated computer programming courses.* |
| **Colaborativa** | *An evaluation of students' motivation in computer-supported collaborative learning of programming concepts.* |
| **Cognitiva** | *Learning Programming at the Computational Thinking Level via Digital Game-Play.* |
| **Colaborativa** | *A pilot study of cooperative programming learning behavior and its relationship with students' learning performance.* |
| **Cognitiva** | *A web-based programming learning environment to support cognitive development.* |
| **Construtivismo e Construcionismo** | *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.* |

A abordagem *Construtivista,* usada nos artigos *The learning context: Influence on learning to program* , *Learning motivation in e-learning facilitated computer programming courses* e *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism* tem como enfoque a construção de novo conhecimento e maneiras de pensar mediante a exploração e a manipulação ativa de objetos e ideias, tanto abstratas como concretas e explicam a aprendizagem através das trocas que o indivíduo realiza com o meio.

A abordagem *Colaborativa,* usada nos artigos *An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. e A pilot study of cooperative programming learning behavior and its relationship with students' learning performance,* enfatiza um processo de aprendizagem em que os estudantes trabalham colaborativamente atuando como parceiros entre si e com o professor, com o objetivo de adquirir conhecimentos sobre um dado objeto. Torres (2004, p.50), afirma que a abordagem colaborativa possui as seguintes características:

- ✓ participação ativa do aluno no processo de aprendizagem;
- ✓ mediação da aprendizagem feita por professores e tutores;
- ✓ construção coletiva do conhecimento, que emerge da troca entre pares, das atividades práticas dos alunos, de suas reflexões, de seus debates e questionamentos;
- ✓ interatividade entre os diversos atores que atuam no processo;
- ✓ estimulação dos processos de expressão e comunicação;
- ✓ flexibilização dos papéis no processo das comunicações e das relações a fim de permitir a construção coletiva do saber;

- ✓ sistematização do planejamento, do desenvolvimento e da avaliação das atividades;
- ✓ aceitação das diversidades e diferenças entre alunos;
- ✓ desenvolvimento da autonomia do aluno no processo ensino-aprendizagem;
- ✓ valorização da liberdade com responsabilidade;
- ✓ comprometimento com a autoria;
- ✓ valorização do processo e não do produto.

A abordagem *Cognitiva,* usada nos artigos: *Learning Programming at the Computational Thinking Level via Digital Game-Play e A web-based programming learning environment to support cognitive development* tem como enfoque o processo central do ser humano. O que implica, dentre outros aspectos, estudar cientificamente a aprendizagem como sendo mais que um produto do ambiente, das pessoas os de fatores que são externos ao aluno.

Nessa abordagem existe a ênfase em processos cognitivos e na investigação científica, separada dos problemas sociais contemporâneos. As emoções são consideradas em suas articulações com o conhecimento. Este tipo de abordagem é predominantemente interacionista (NOBRE, 2013).

O *Construcionismo*, usado no artigo *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.* É uma abordagem pedagógica desenvolvida pelo pesquisador em Educação e Tecnologias da Informática, Seymour Papert, que propõe uma transformação na concepção do processo de ensino-aprendizagem através do uso do computador como uma ferramenta que propicia ao aluno condições concretas de explorar o seu potencial intelectual,

desenvolvendo ideias nas mais diferentes áreas do conhecimento. Papert reforça a ideia de que o professor deve assumir um papel de mediador e promotor do processo de aprendizagem, estimulando a reflexão, a depuração e a construção do conhecimento de forma a criar um ambiente onde o aluno é o sujeito da aprendizagem.

### 2.3.5 Ferramentas desenvolvidas para apoiarem a aprendizagem de programação

Nesta seção são apresentadas as ferramentas desenvolvidas para apoiarem a aprendizagem de programação citadas nos artigos selecionados, como propostas de melhorias ao processo de aprendizagem. Na Tabela 07, são apresentados os artigos e as respectivas ferramentas.

Tabela 07. Ferramentas desenvolvidas

| Artigo | Ferramenta |
|---|---|
| ***A pilot study of cooperative programming learning behavior and its relationship with students' learning performance.*** | *WPASC* - Um sistema assistido de programação baseada em *web* para a cooperação. |
| ***A web-based programming learning environment to support cognitive development.*** | Um ambiente de aprendizagem de programação baseado em *web* para apoiar o desenvolvimento cognitivo |
| ***The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.*** | Um ambiente de aprendizagem interativo para programação. |
| ***Learning Programming at the computational Thinking Level via Digital*** | Jogo digital que permite o desenvolvimento de habilidades relacionadas ao *CT* |

| | |
|---|---|
| ***Game-Play*** | *(Computational Thinking)* usando um número limitado de construções de programação. |

- Os artigos *The learning context: Influence on learning to program, An evaluation of students' motivation in computer-supported collaborative learning of programming concepts, Learning motivation in e-learning facilitated computer programming courses,* não apresentaram nenhuma ferramenta. Abaixo listamos o que cada um desses artigos propõem para melhorar o processo de aprendizagem de programação.

- O Artigo *The learning context: Influence on learning to program*- argumenta sobre a influência que as experiências do contexto de aprendizagem tem sobre os alunos, e quais as implicações que esta proporciona para o ensino e a aprendizagem de programação.

  O contexto de aprendizagem em que o autor do artigo se refere se trata de palestras, processo de estudo, conhecimento anterior ou experiências de ensino que o aluno trás, antes de ingressar em uma disciplina de programação.

- O artigo *An evaluation of students' motivation in computer-supported collaborative learning of programming concepts*, apresenta uma avaliação da motivação dos alunos, com o objetivo de apresentar os níveis de motivação dos estudantes com relação a quatro abordagens pedagógicas: aula tradicional, aprendizagem colaborativa, aprendizagem colaborativa guiada por *CIF*(uma estrutura de ensino para aprendizagem colaborativa) e aprendizagem colaborativa guiados por *CIF* e apoiado por *MoCAS*(uma ferramenta de aprendizagem colaborativa).

- O artigo *Learning motivation in e-learning facilitated computer programming courses* representa uma iniciativa importante para entender os principais fatores que afetam a aprendizagem do aluno em cursos de programação de computadores.

Este trabalho fornece evidências de que um ambiente de aprendizagem bem facilitado, ou seja, um ambiente que seja de fácil entendimento para o aluno pode, efetivamente, melhorar a motivação e eficácia do mesmo.

## 2.4 Síntese da revisão sistemática

Nesta seção, é apresentada a Tabela 08, com uma síntese de dados coletados nesta Revisão Sistemática.

Tabela 08. Síntese da Revisão sistemática

| Referência | Público alvo | Habilidades de programação | Recursos digitais usados | Abordagem usada | Problemas tratados | Aspectos importantes |
|---|---|---|---|---|---|---|
| Govender, I. (2009). The learning context: Influence on learning to program.*Computers & Education*, *53*(4), 1218-1230. | Estudantes, professores e pesquisadores da área de Informática na Educação. | Habilidade em resolver problemas; | Diário Digital, este recurso digital foi usado como suporte computacional para que os alunos pudessem registrar suas experiências de aprendizagem, ao longo do curso. As reflexões publicadas nele, juntamente com as transcrições das entrevistas foram utilizados na análise final dos dados. | Construtivismo | Motivação. | O artigo apresenta diferentes contextos de aprendizagem, considerados como influências na aprendizagem de programação. |
| Law, K. M., Lee, V., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218-228. | Estudantes, professores e pesquisadores da área de Informática na Educação. | Habilidade em resolver problemas; | *PASS (Programming Assignment aSsessment System*). O *PASS* visa proporcionar uma infra-estrutura de modo a facilitar a aprendizagem. Através de fatores notavelmente motivador, como: "a atitude individual e expectativa", "orientação clara" e "recompensa e reconhecimento". | Construtivismo | Estilos de aprendizagem | O artigo é um estudo preliminar que investiga os fatores motivadores que afetam a aprendizagem dos estudantes universitários que participam de cursos de programação de computadores . |

| | | | | | | |
|---|---|---|---|---|---|---|
| Serrano-Cámara, L. M. ; Paredes-Velasco, M. ; Alcover, C. M. ; Velazquez,I., J. Ángel. An evaluation of students' motivation in computer-supported collaborative learning of programming concepts In: Computers in Human Behavior, 2013. | Estudantes, professores e pesquisadores da área de Informática na Educação. | Codificar a solução do problema. | *MoCaS (Mobile Collaborative Argument Support*) ferramenta de aprendizagem colaborativa, usada como suporte computacional para uma estrutura de ensino para aprendizagem colaborativa. | Colaborativa | Motivação | O artigo apresenta uma avaliação da motivação de 139 alunos, com o objetivo de avaliar os níveis de motivação em relação a quatro metodologias com e sem suporte tecnológico. |
| Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning Programming at the Computational Thinking Level via Digital Game-Play.*Procedia Computer Science*, *9*, 522-531. | Estudantes, professores e pesquisadores da área de Informática na Educação. | Habilidades cognitivas;<br><br>Desenvolver soluções;<br><br>Projetar sistemas que coincidem com o pensamento lógico;<br><br>Resolver problemas;<br><br>Construção de algoritmos. | Jogo digital que permite o desenvolvimento de habilidades relacionadas ao *CT (Computational Thinking)* usando um número limitado de construções de programação. | Cognitiva | Fatores cognitivos | O artigo descreve um modelo de jogo para o desenvolvimento de habilidades cognitivas. |
| Hwang, W. Y., Wang, C. Y., | Estudantes, | | *WPASC* - Um sistema assistido de | | | Esta pesquisa propôs um |

| | | | | | | |
|---|---|---|---|---|---|---|
| Hwang, G. J., Huang, Y. M., & Huang, S. (2008). A web-based programming learning environment to support cognitive development. *Interacting with Computers*, *20*(6), 524-534. | professores e pesquisadores da área de Informática na Educação. | Habilidade de desenvolver programas de forma colaborativa | programação baseada em web para a aprendizagem cooperativa. | Colaborativa | Estilos de aprendizagem | assistente de aprendizagem cooperativa para aprendizagem de programação baseado na web, para a cooperação ( *WPASC* ). Ambiente usado para projetar atividade para facilitar a aprendizagem cooperativa de programação. |
| Moons, J.; Backer, C. D.: The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. In: Computers & education 60 (2013) 368-384 | Estudantes, professores e pesquisadores da área de Informática na Educação. | Habilidades em aprender e trabalhar com conceitos novos;<br><br>Compreender e solucionar problemas;<br><br>Habilidades cognitivas complexas;<br><br>Raciocínio lógico. | um ambiente de aprendizagem interativo para programação. | Construtivismo e Construcionismo | Fatores cognitivos;<br><br>Estilos de aprendizagem;<br><br>Planejamento para projetar programas. | Este artigo apresenta a arquitetura e avaliação de um ambiente para aprendizagem de programação. Este ambiente usa técnicas de visualização de programas para ajudar os professores e os estudantes de um curso introdutório de programação . O ambiente é inspirado por paradigmas de aprendizagem construtivista e cognitivista. |

| | | | | | | |
|---|---|---|---|---|---|---|
| Hwang, Wu-Yuin, et al. "A pilot study of cooperative programming learning behavior and its relationship with students' learning performance." *Computers & Education* 58.4 (2012): 1267-1281. | Estudantes, professores e pesquisadores da área de Informática na Educação. | Habilidade de desenvolver programas de forma colaborativa | *WPASC* - Um assistente de programação baseada em web para a cooperação | Colaborativa | Estilos de aprendizagem | O artigo, é um estudo piloto sobre o comportamento cooperativo dos alunos na aprendizagem de programação. |

A Tabela 08 é uma síntese da revisão sistemática. Uma síntese nada mais é que um resumo do texto original, onde são considerados apenas os pontos principais abordados pelo autor.

A síntese apresentada na Tabela 08 é formada por sete colunas e oito linhas. Nas colunas são inseridas informações como: Referência, Público alvo, Habilidades de programação, Recursos digitais usados, Abordagem usada, Problemas tratados e Aspectos importantes.

A coluna *Referência* é composta pela referência de cada artigo, esta referência é descrita com: os nomes dos autores, o título do artigo e o ano em que foram escritos. Na coluna *Público alvo*, é mostrado para que público aquele determinado artigo foi escrito, este publico pode ser: alunos, professores ou pesquisadores.

A coluna *Habilidades de programação* apresenta informações referentes às habilidades de programação tratadas em cada artigo. A próxima coluna *Recursos digitais usados*, aborda os recursos digitais usados em cada artigo, os recursos digitais são ferramentas tecnológicas usadas nos artigos para atingir um determinado resultado.

A coluna *Abordagem usada*, descreve a abordagem pedagógica usada nos artigos escolhidos na revisão sistemática. As abordagens usadas foram:

- *colaborativa* – artigos: *An evaluation of students' motivation in computer-supported collaborative learning of programming concepts, A pilot study of cooperative programming learning behavior and its relationship with students' learning performance, A web-based programming learning environment to support cognitive development;*

- *cognitiva* – artigos: *Learning Programming at the Computational Thinking Level via Digital Game-Play;*

- *construtivismo* – artigos: *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism, The learning context: Influence on learning to program, Learning motivation in e-learning facilitated computer programming courses;*

- *construcionismo* – artigo: *The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism*.

A coluna *Problemas tratados*, mostra quais os problemas que são tratados em cada artigo, observou-se que a maioria dos artigos trataram como problema os estilos de aprendizagem. O termo *Estilos de aprendizagem* trata-se de maneiras usadas para aprender o que lhe é proposto, tais estilos são únicos e pessoais, pois cada pessoa apresenta facilidade com determinados estilos aprendizagem e dificuldades em outros.

E por fim temos a coluna *Aspectos importantes*, esta coluna apresenta de forma resumida os aspectos julgados mais importantes de cada artigo. Cada linha da Tabela 08 mostra informações relacionadas a um determinado artigo.

# 3 ARQUITETURAS PEDAGOGICAS

Tavares *et.alli* (2012) definem arquiteturas pedagógicas como abordagens pedagógicas e os recursos tecnológicos necessários para sua implementação. Neste sentido, o objetivo deste capítulo é definir o que são “Arquiteturas pedagógicas”, tema capital para o desenvolvimento da presente dissertação.

A Seção 3.1 se inicia com uma breve fundamentação teórica sobre as Arquiteturas pedagógicas. Na Seção 3.2 é mostrado alguns trabalhos que serviram de inspiração para a escolha do tema desta dissertação. Na Seção 3.3 são apresentadas as considerações finais.

## 3.1 Fundamentação Teórica

A cada ano, novas tecnologias vem proporcionar uma distribuição rápida e eficiente de informação. Para Stahl (1995) o uso das novas tecnologias amplia consideravelmente o nível de informação e contribui para o aumento do conhecimento.

A tecnologia permite o trabalho e aprendizagem a distância. Viabiliza o dialogo, a discussão, a pesquisa, perguntas, respostas, transmissão de informações, por meio de recursos que permitam aos interlocutores viverem nos lugares mais longínquos, com a possibilidade de se comunicarem e enriquecerem com contatos mútuos (MASETTO, 2000).

Reis & Godói (2010) enfatizam que vivemos num mundo em constantes transformações, o qual traz novos desafios ao ambiente escolar. O ritmo em que acontecem essas transformações tem gerado novas necessidades relacionadas à prática docente e, consequentemente, novas formas de ensinar e aprender são questões que merecem reflexão.

Para Brito (2011) a verdadeira função do aparato educacional não é de ensinar, mas sim, a de criar condições de aprendizagem. Desta forma o ambiente educativo contribui sobremaneira para o desenvolvimento de novos processos cognitivos.

Como definição de arquiteturas pedagógicas, o conceito adotado neste trabalho está baseado em Carvalho, Nevado e Menezes (2007), que definem Arquiteturas Pedagógicas como “estruturas de aprendizagem realizadas a partir da confluência de diferentes componentes: abordagem pedagógica, software educacional, internet, inteligência artificial, Educação a Distância, concepção de tempo e espaço”.

Behar, *et.alli* (2007) apresentam os elementos que constituem uma arquitetura pedagógica como: aspectos organizacionais, metodológicos, tecnológicos e conteúdo. A construção das arquiteturas pedagógicas metaforicamente pode ser relacionada a uma *atividade artesanal*, uma vez que é tecida em uma rede de relações entre as experiências vivenciadas e a reflexão sobre diferentes fatos e objetos relacionados com o meio de atuação em estudo.

Na perspectiva das arquiteturas pedagógicas, alteram-se as concepções de tempo e espaço para a aprendizagem, já que o conhecimento tem como ponto de partida arquiteturas plásticas que se moldam aos ritmos impostos pelo sujeito que aprende (Nevado, *et.alli*, 2009).

## 3.2 Trabalhos correlatos

Nesta seção, é feita uma abordagem de arquiteturas pedagógicas que serviu de motivação para a realização deste trabalho.

### 3.2.1 *Pedagogical architectures to support teaching and learning of computer programming*

Tavares *et alli* (2012) apresentam uma arquitetura pedagógica baseada em uma abordagem construtivista que enfatiza o aprender a aprender, a autonomia dos estudantes, o trabalho cooperativo, a socialização dos resultados, a busca por gerar várias soluções possíveis.

Essa arquitetura pedagógica tem como princípios:

- ✓ *autoria de soluções,* com o incentivo aos alunos para que desenvolvam soluções, reflitam sobre os resultados e sobre o processo de desenvolvimento das soluções e criem um método personalizado de construção de programas;

- ✓ *cooperação,* os alunos trabalham de forma cooperativa para melhorar a aprendizagem e se motivam à participar mais ativamente neste processo;

- ✓ *pensamento crítico*, este princípio pode ser desenvolvido no aluno por meio de comparações das soluções construídas por ele com as soluções criadas por outros alunos.

### 3.2.2 Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático

Carvalho, *et alli* (2005) apresentam quatro tipos de arquiteturas pedagógicas que configuram-se como re-leituras de abordagens pedagógicas, ao realizarem a intersecção entre projeto educativo e o suporte telemático.

Essas arquiteturas buscam traduzir em situações de aprendizagem, propostas pedagógicas concebidas para a mediação da aprendizagem, caracterizadas

por deslocamento das concepções hierárquicas e disciplinares de ensino, na direção de uma concepção do conhecimento interdisciplinar e do modelo de formação de professores como elementos de uma rede de relações (CARVALHO *et. alli*, 2005). As arquiteturas pedagógicas apresentadas por Carvalho *et alli* (2005) foram:

*Arquitetura de Projetos de Aprendizagem* - nesta arquitetura a sistematização da aprendizagem compreende o lançamento de questões de investigação e formulações a partir de *Certezas Provisórias* e *Dúvidas Temporárias* a respeito das questões, formuladas pelo grupo de desenvolvimento do projeto. Em seguida é feito um inventário dos conhecimentos. O processo de investigação consiste no esclarecimento das dúvidas e na validação das certezas.

Nesta arquitetura o suporte telemático pode ser feito através da criação de um ambiente *web* de apoio ao desenvolvimento das diferentes etapas do projeto de aprendizagem. Nesse ambiente pode haver recursos para apoiar as interações do grupo de aprendizes, publicação dos itens de conhecimento levantados e socialização dos resultados, em um processo que se retroalimenta infinitamente.

A publicação dos resultados na *web* tem várias contribuições para o próprio processo. Cada estudante apresenta-se como produtor de conhecimento, abrem-se possibilidades dele se integrar a uma rede de autores, ao invés de um mero consumidor.

*Arquitetura de Estudo de Caso ou Resolução de Problema* - essa proposta apresenta a ciência como relativa e mutável. Nessa arquitetura a verdade da ciência não está capitalizada apenas nas verdades adquiridas, na verificação das teorias conhecidas, mas no seu caráter aberto de aventura.

O caso ou problema tem sentido para alguém que necessita saber algo. Essa necessidade é criada pelo próprio sujeito através de algo que lhe instiga como:

perguntas, filmes, problemas etc. o caso deve responder a exigência de uma tarefa complexa, no qual são satisfeitas as seguintes questões:

- ✓ O que o estudante deve saber para resolver esse problema?
- ✓ O estudante pode realizar por si ou realiza somente com o auxílio de especialistas?
- ✓ O que o estudante deve considerar?

A cada pergunta há a necessidade de indexar situações que possam auxiliar a iniciar e a prosseguir com um problema de modo relativamente autônomo. O suporte telemático usado nessa arquitetura é um ambiente de autoria de casos, um ambiente de resolução de casos e um ambiente para apoio à recuperação de casos. O ambiente de autoria deve permitir uma descrição dos casos no uso de diferentes mídias, textos, vídeos e debates.

O ambiente de resolução de casos deve apoiar o debate, a análise e o registro de progressos. O ambiente de recuperação de casos é um sistema de armazenamento, categorização e recuperação de documentos multimídia.

*Arquitetura de Aprendizagem Incidente* - essa arquitetura é uma forma de aprendizagem derivada, particularmente em relação a alguma informação impreterível ou cansativa. Tal arquitetura se presta em tarefas sobre algum conteúdo onde é necessário dispor de informação a fim de cumprir um objetivo.

A chave do aprendizado incidente é descobrir atividades que sejam intrinsecamente divertidas de se realizar com o computador e com a rede/internet. Como exemplo, podemos citar uma atividade em que os estudantes explorem um programa para que descubram algo, experimentem e avancem até onde se sintam satisfeitos. Após a descoberta os alunos criam um relatório descrevendo tal experiência.

O suporte telemático apresentado é constituído de ambientes de autoria. A ideia dessa arquitetura é que o aluno realize várias atividades, de preferência

com certo grau de independência, com a visão de atingir uma atividade mais geral.

*Arquitetura de Ação Simulada* - a ideia dessa arquitetura defende que o melhor meio de aprender a realizar uma atividade é fazê-la ou *aprender a fazer fazendo*. Neste caso a arquitetura exige a criação de simulações que efetivamente reproduzem situações da vida real. O suporte telemático usado são ambientes de simulação que permitem suporte à autoria de cenários, descrição de regras e a execução monitorada de ações.

## 3.3 Considerações finais

Nos últimos anos, as instituições de ensino têm enfrentado um significativo processo de transformação na educação. As transformações que envolvem o processo de educação decorrem da introdução das Tecnologias da Informação e Comunicação (*TICs*) na educação, pois essas *TICs* salientam a necessidade de mudar de forma significativa o modelo pedagógico anteriormente vigente.

As arquiteturas pedagógicas mostram-se relevantes como possibilidades estruturantes de aprendizagem, pois trazem avanços na construção do conhecimento mediante os registros de mudanças de pensamentos, no que favorece a aprendizagem, por meio do acompanhamento da trajetória de cada estudante.

# 4 UM AMBIENTE PARA APRENDIZAGEM DE PROGRAMAÇÃO

O objetivo deste capítulo é apresentar um ambiente para aprendizagem de programação com o uso de arquiteturas pedagógicas, no intuito de propor melhorias no processo de aprendizagem.

O professor por intermédio das arquiteturas pedagógicas identifica problemas encontrados na correção de programas escritos pelos aprendizes, como: baixo nível de abstração (JENKINS 2002; RIBEIRO, 2012) e dificuldades na compreensão dos conceitos de programação (SOUZA, 2012).

Erros de sintaxe e semântica (GOMES e MENDES, 2000), erros de conhecimentos básicos de matemática, como: cálculo básico, teoria de números, conceitos geométricos e trigonométricos simples, relação na descrição textual de um problema com a fórmula matemática que o resolva (GOMES, 2006), além de problemas relacionados a deficiências do raciocínio lógico (GOMES, 2008) e erros com variáveis, tipos de dados e memória dinâmica (DUNICAN, 2002).

## 4.1 Motivações

Real (2011) afirma que as arquiteturas pedagógicas em sua proposta rompem com a pedagogia tradicional. Isso porque o uso de recursos digitais fomenta uma aprendizagem interativa, na qual o discente torna-se o sujeito de sua aprendizagem.

O professor tem um papel fundamental nesse processo, pois deve propiciar espaços e orientações por meio de arquiteturas pedagógicas que apoiem a construção da autonomia e do conhecimento dos aprendizes.

Nesta pesquisa, interessamo-nos por um ambiente de aprendizagem de programação com uso de arquiteturas pedagógicas alicerçadas na aprendizagem significativa.

Aprendizagem significativa é o mecanismo humano usado para adquirir e armazenar a vasta quantidade de ideias e informações representadas em qualquer campo de conhecimento (AUSUBEL, 1963).

Neste sentido, considera-se que a ressignificação de conteúdos por meio de arquiteturas pedagógicas propõe melhorias no processo de aprendizagem, uma vez que trazem avanços na construção do conhecimento mediante registros de mudanças de pensamentos que favorecem a trajetória de cada estudante.

## 4.2 Arquitetura Pedagógica *CSPL* (*Computer Supported Programming Learning – Computadores como suporte a aprendizagem de programação*)

A arquitetura Pedagógica *CSPL*, combina os conceitos de aprendizagem baseada em problemas e aprendizagem cooperativa para facilitar o processo de ensino-aprendizagem. Nessa arquitetura pedagógica, usou-se como referência a teoria de Piaget (1972).

Segundo a teoria de Piaget, o indivíduo passa por um processo de desenvolvimento cognitivo que se inicia com a manipulação de objetos concretos, passando pelo uso de operações mentais, até desenvolver a habilidade de formalização, o que permite abstrair detalhes sem importância, generalizar características de objetos, dentre outros processos cognitivos. A abordagem pedagógica escolhida tem como objetivo, explorar a aprendizagem de forma individual e colaborativa, tendo como base:

- ✓ resolução de problemas de forma individual;

- ✓ resolução de problemas cooperativamente;
- ✓ regulação da aprendizagem.
- ✓ espírito crítico

Estratégia pedagógica usada nesta arquitetura é apresentada na Figura 02:

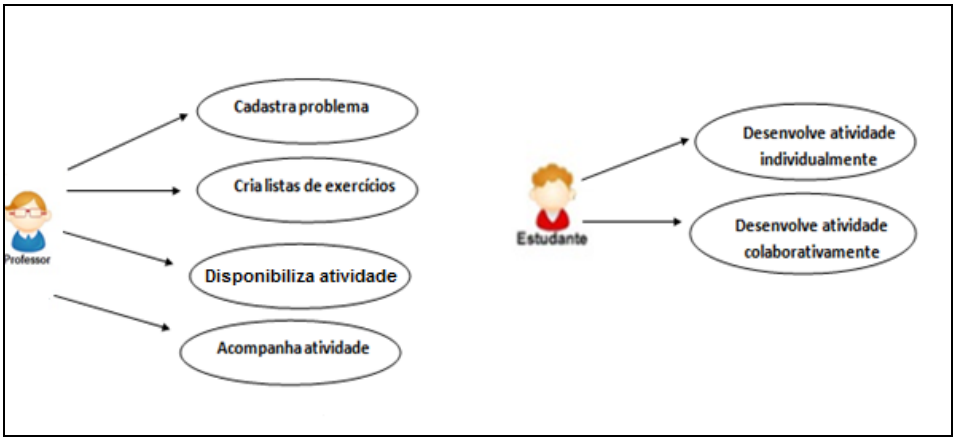


Figura 02 – Cenário da Arquitetura pedagógica *CSPL*

O método escolhido para a construção de programas foi o *MCP* (Método de construção de programas). É com base neste método que a arquitetura pedagógica *CSPL* foi construída. O *MCP* é um método criado por Tavares *et alli* (2013) no qual é estruturado em seis etapas, em que os estudantes passam por cada uma delas para a construção de algoritmos. A Tabela 09 apresenta as etapas do *MCP*.

Tabela 09 – Etapas do Método *MCP- fonte Tavares, ett alli 2013*

| Etapas | Objetivo |
|---|---|
| Compreensão do problema | Compreender as principais dificuldades apresentadas pelo aluno em relação à compreensão do problema proposto |
| Planejamento do teste | Definir mapeamentos entre dados de entrada relevantes e resultados esperados |
| Especificação da solução | Especificar uma solução a partir da composição dos métodos e/ou funções de mapeamento usados no plano de teste. |
| Codificação do programa | Codificar a solução em uma linguagem de programação |
| Teste da solução | Testar a correção da solução codificada |
| Avaliação do processo e seus resultados | Analisar cada etapa da construção da solução, consolidando os pontos fortes e corrigindo os pontos fracos. |

*Etapa 1 - Compreensão do problema:* nessa etapa o estudante relata a sua compreensão do problema proposto, levando em consideração a relação entre os dados de entrada e saída, o domínio de entrada e a resposta esperada.

*Etapa 2 - Planejamento do teste:* nessa etapa o estudante define os dados de entrada, representativos dos subdomínios relevantes do problema, a serem usados para testar a correção da solução, bem como os resultados esperados para cada um desses dados de entrada;

*Etapa 3 - Especificação da solução:* nessa etapa o estudante identifica os métodos e/ou funções que compõem a solução do problema e constrói uma ou mais opções de solução. Aqui ele explicita as funções usadas para mapear os dados de entrada nos resultados esperados, usados implicitamente na etapa 2, e compõe essas funções para construir uma solução;

*Etapa 4 - Codificação do programa:* nessa etapa, é codificada a solução em uma linguagem de programação. Para esta pesquisa, optamos por codificar as soluções em *Haskell*, que é uma linguagem de programação funcional e de fácil compreensão para os graduandos de computação, por usar a linguagem matemática conhecida por eles (Souza, 2009).

*Etapa 5 - Teste da solução:* nessa etapa, é feito o teste da correção da solução codificada. O teste da solução consiste em usar os dados de entrada do plano de teste para avaliar o funcionamento da solução codificada, comparar os resultados esperados do plano de teste com os resultados obtidos na execução do código da solução, com o uso dos dados de entrada definidos no plano de teste;

*Etapa 6 - Avaliação do processo e seus resultados:* nessa etapa, o estudante analisa cada etapa da construção da solução, consolida os pontos fortes e corrige os pontos fracos. Nessa análise é sugerida uma quantificação de vários

critérios, tais como, legibilidade, manutenibilidade, reusabilidade, eficiência e recursos necessários.

### 4.2.1 Participantes da Arquitetura Pedagógica *CSPL*

Os participantes desta arquitetura pedagógica são professores e alunos de uma disciplina inicial de programação de computadores. No intuito de favorecer a aprendizagem, os alunos desenvolvem soluções para os problemas propostos e fazem a edição e a entrega dessas soluções com recursos digitais que compõem um ambiente virtual de aprendizagem.

### 4.2.2 Etapas da Arquitetura Pedagógica *CSPL*

As etapas da Arquitetura Pedagógica *CSPL* são constituídas de interações distribuídas entre os professores e alunos, como apresenta a lista a seguir:

*Cadastro de problemas* – esta etapa é atribuída ao professor, que por sua vez faz o cadastro de todos os problemas que deseja que os estudantes resolvam. No cadastro dos problemas, o professor indica o título e o enunciado do problema, enviando essas informações para uma base de dados. Os problemas dessa base de dados serão usados na criação de listas de exercícios.

Assim como na etapa anterior, na etapa *criar listas de exercícios* o docente cria listas de exercícios e insere os problemas já cadastrados. Essas listas de exercícios criadas serão usadas para que os estudantes visualizem e escolham as listas que pretendem desenvolver.

Além da inserção dos problemas nas listas, o professor pode adicionar um prazo para que as atividades sejam desenvolvidas e informar se as atividades

serão desenvolvidas de forma individual ou em grupo. No caso da lista de exercícios ser preparada para o desenvolvimento em grupo, o professor pode informar a quantidade de grupos e de componentes dos grupos.

Na *disponibilização de atividades* são apresentadas aos alunos as listas de exercícios criadas pelo professor, para que os alunos comecem a desenvolver as soluções de cada problema que compõe a lista.

Na quarta etapa, na *escolha de atividades propostas* os alunos têm a opção de escolher dentre as listas de exercícios propostas pelo professor, a lista que começará a desenvolver. A escolha é feita quando o aprendiz acessa a lista de exercícios desejada e escolhe o problema a ser desenvolvido.

Na quinta etapa, *desenvolvimento de atividades propostas* os alunos desenvolvem as soluções para cada problema que compõe a lista de exercício. Em cada desenvolvimento de solução dos problemas, o aluno passa pelas etapas do método *MCP* (compreensão do problema, planejamento do teste, especificação da solução, codificação da solução, teste de solução e avaliação dos processos e seus resultados).

Ao final, é realizado o envio das etapas para que o professor possa fazer a avaliação da atividade. Na sexta e última etapa, *Acompanhamento de atividades*, o professor faz o acompanhamento das atividades que estão sendo desenvolvidas pelos alunos.

No acompanhamento da aprendizagem o docente visualiza as etapas do método de construção de programas (compreensão do problema, especificação da solução, plano de teste e o código da solução) que foram desenvolvidas pelos alunos. Este acompanhamento é importante, pois o professor consegue acompanhar a evolução da aprendizagem de cada discente ao analisar cada etapa desenvolvida.

### 4.2.3 Recursos tecnológicos da Arquitetura Pedagógica CSPL

Os recursos tecnológicos aplicados nesta arquitetura pedagógica foram:

- ✓ cadastrador de problemas;
- ✓ distribuidor de tarefas, individuais ou em grupo;
- ✓ criador de listas de exercícios;
- ✓ apresentador de listas de exercícios;
- ✓ apresentador de problemas;
- ✓ apresentador de etapas do *MCP* (compreensão do problema, especificação da solução, codificação da solução e especificação da solução);
- ✓ recebedor de etapas;
- ✓ compilador de soluções;
- ✓ comparador do plano de teste com a solução executada;
- ✓ acompanhador de aprendizagem;
- ✓ distribuidor de tarefas, individuais ou em grupo;

## 4.3 Arquitetura Pedagógica Programação em Pares

A arquitetura Pedagógica Programação em Pares é uma arquitetura no qual os alunos têm a possibilidade de desenvolver tarefas em pares (duplas). Com essa arquitetura pedagógica, é exercitado o espírito crítico dos aprendizes e são exploradas outras possíveis soluções para cada problema, uma vez que cada aprendiz é convidado a revisar a solução do seu par e construir, colaborando com o seu par, uma terceira solução, diferente das duas anteriormente desenvolvidas individualmente.

A abordagem pedagógica proposta para essa arquitetura tem como objetivo explorar a aprendizagem de programação em pares e desenvolver o espírito

crítico do aprendiz. A estratégia pedagógica usada nesta arquitetura consiste dos seguintes passos:

- ✓ o professor propõe a atividade;
- ✓ cada aluno desenvolve e envia individualmente suas soluções (uma ou mais);
- ✓ o professor forma os pares por similaridade (ou diferenças) entre as soluções desenvolvidas de forma individual, a partir de um recurso tecnológico de análise de similaridades entre programas;
- ✓ cada participante revisa a solução de seu par;
- ✓ cada par desenvolve, valida (com o uso de recurso digital de análise de similaridades entre programas) e envia sua(s) solução(ões).

A Figura 03 apresenta os casos de uso dos recursos digitais de suporte à Arquitetura pedagógica Programação em pares.

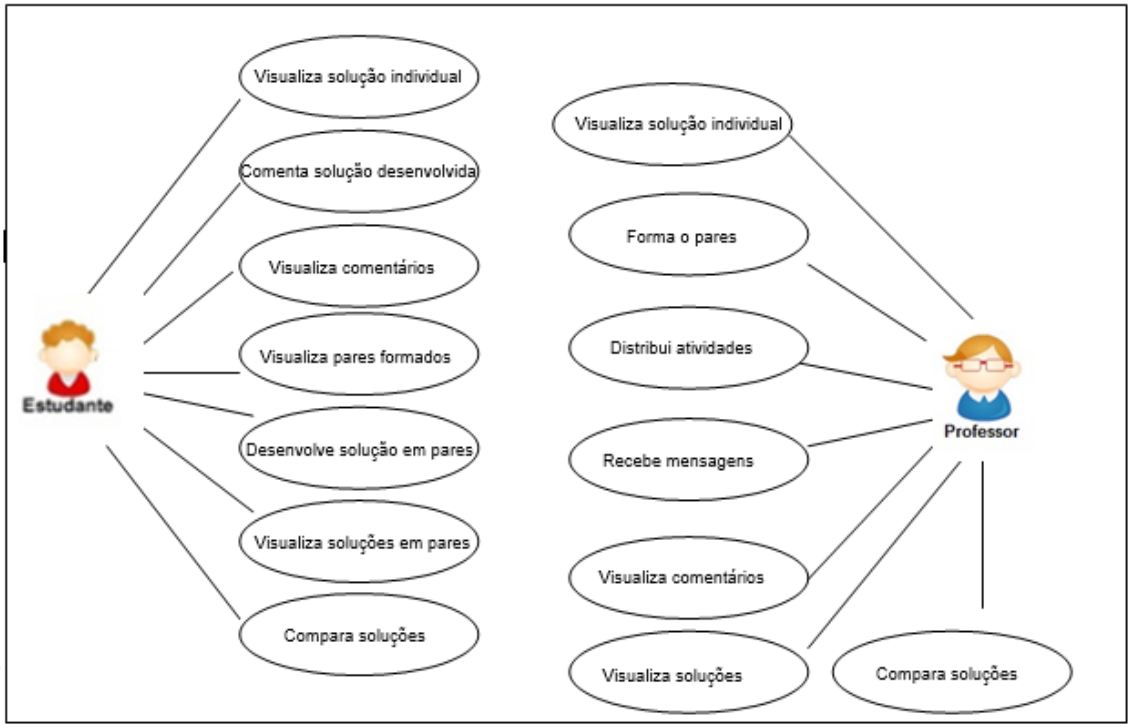


Figura 03 – Casos de uso da Arquitetura pedagógica Programação em pares

### 4.3.1 Participantes da Programação em Pares

Os participantes desta arquitetura pedagógica são professores e alunos de uma disciplina inicial de programação de computadores. No intuito de favorecer a aprendizagem, os alunos desenvolvem soluções para os problemas propostos e fazem seu envio através de um ambiente virtual de aprendizagem. O professor, por sua vez, faz a formação dos pares com base em um percentual de semelhança ou diferença entre as soluções desenvolvidas.

### 4.3.2 Etapas da Programação em Pares

As etapas para a Programação em pares são constituídas das atividades realizadas pelos participantes da arquitetura pedagógica, conforme mostra a Figura 04.

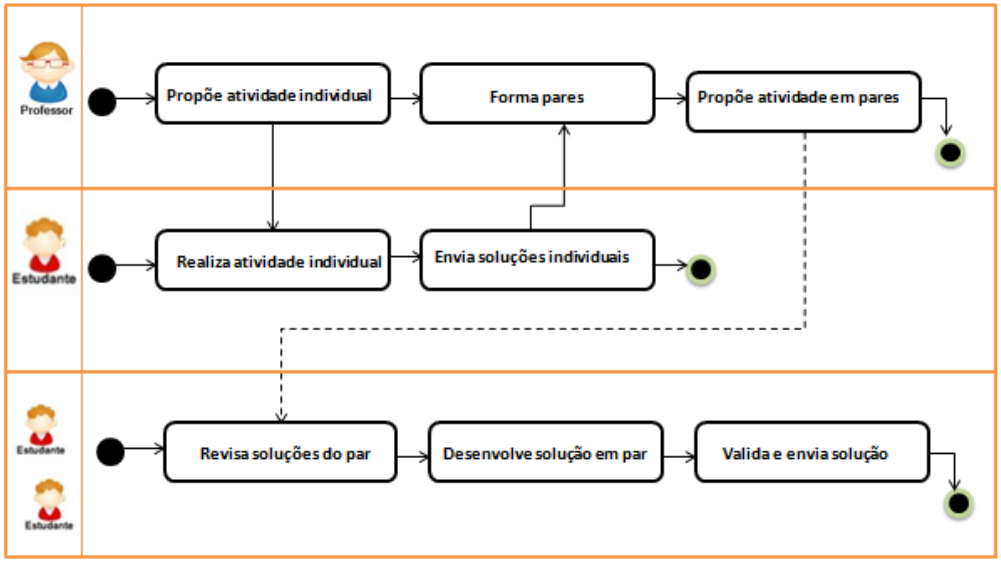


Figura 04: Fluxo das etapas da formação de pares

Na primeira etapa *propõe atividade individual*, o professor disponibiliza atividades para serem desenvolvidas pelos alunos de forma individual. Essas

atividades são relacionadas à resolução das listas de exercícios com problemas pré-cadastrados na arquitetura pedagógica *CSPL.*

Na segunda etapa - *realiza atividade individual*", o estudante desenvolve os problemas propostos nas listas de exercícios de forma individual, passando por todas as etapas do método *MCP.*

Na terceira etapa - *envia soluções individuais*, os alunos constroem e enviam as soluções, para que o professor possa fazer a análise das soluções e formar os pares. O envio das soluções é feito via recursos de um ambiente virtual de aprendizagem.

Na quarta etapa - *forma pares*, o professor faz a análise das soluções no intuito de formar os pares pela semelhança ou diferença entre elas.

Na quinta etapa - *propõe atividades em pares*, é feito o envio para cada estudante de duas outras atividades a serem realizadas: revisar a solução do parceiro de par e construir colaborativamente uma nova solução com seu par.

Na etapa *revisa soluções do par*, os alunos comentam a solução individual desenvolvida por cada integrante do par, apontando semelhanças e diferenças entre as soluções e buscando propor melhorias para o desenvolvimento de uma nova solução. Os comentários postados podem ser visualizados tanto pelo professor, como também pelos demais colegas do curso.

Na etapa *desenvolver solução em par*, o par desenvolve uma nova solução para o mesmo problema proposto. Esta solução é feita de forma colaborativa e deve ser diferente das duas soluções desenvolvidas individualmente pelos componentes do par.

Na etapa a seguir, *valida e envia solução,* cada par faz o teste da solução colaborativa e, caso esteja correta, a envia ao professor.

As novas soluções desenvolvidas pelos pares devem possuir no mínimo 20% de diferença, se comparadas àquelas desenvolvidas anteriormente de forma individual.

### 4.3.3 Recursos tecnológicos da Programação em Pares

Os recursos tecnológicos usados na arquitetura pedagógica Programação em Pares foram:

- ✓ apresentador de problema;
- ✓ recebedor de solução;
- ✓ medidor da diferença/ semelhança entre as soluções;
- ✓ comentador da solução do colega;
- ✓ visualizador de comentários;
- ✓ publicador da solução colaborativa;
- ✓ visualizador de soluções em pares;

## 4.4 Arquitetura Pedagógica Recomendação de Exercícios

A Arquitetura Pedagógica Recomendação de Exercícios, tem como objetivo facilitar a aprendizagem do estudante de programação, por meio de recomendações de exercícios adequados ao perfil de aprendizagem do estudante. Além disso, o histórico de aprendizagem do aluno fica armazenado, para permitir o acompanhamento de sua aprendizagem pelo professor.

A recomendação de exercícios é feita de forma automática, em que o próprio ambiente imita o professor recomendando exercícios para os alunos com base no seu perfil de aprendizagem. A Figura 05 apresenta os casos de uso da Arquitetura pedagógica Recomendação de Exercícios.

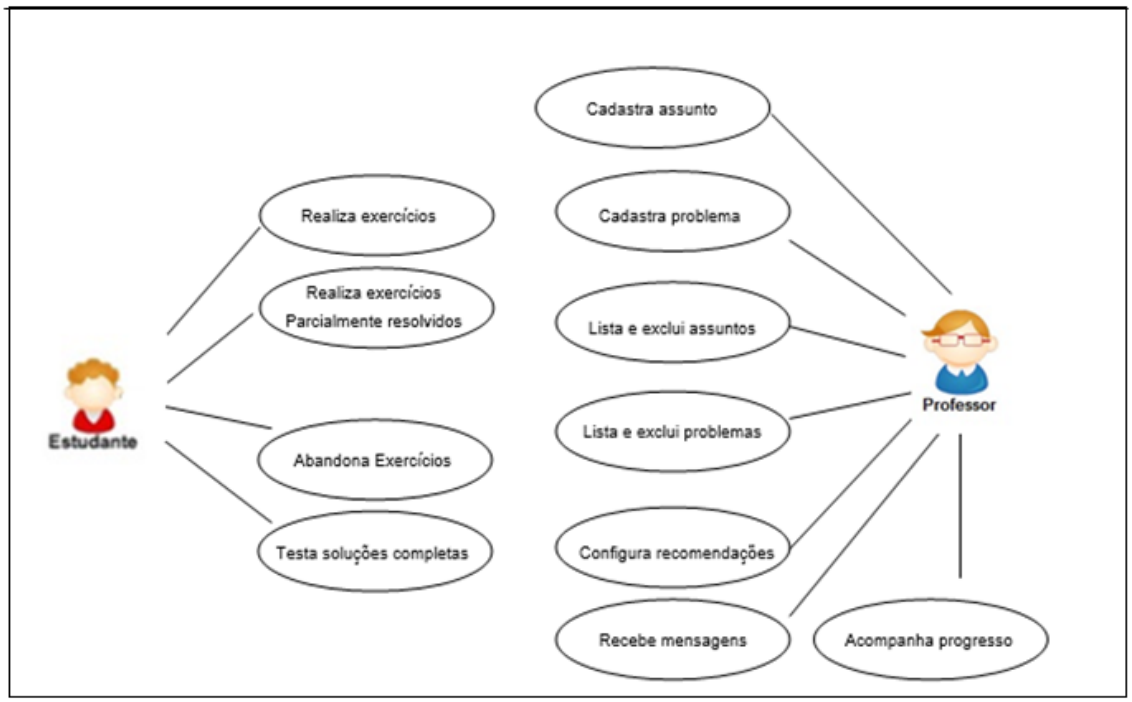


Figura 05 – Diagramas de casos de uso da Arquitetura pedagógica Recomendação de exercícios

A abordagem pedagógica proposta para a Arquitetura Pedagógica Recomendação de Exercícios, teve como base:

- ✓ resolução de problemas de modo individual;
- ✓ recomendação de problemas, parcialmente ou totalmente resolvidos, equivalentes aqueles anteriormente recomendados e não resolvidos;
- ✓ autorregulação – o próprio estudante vê imediatamente o resultado da avaliação da sua solução;
- ✓ espírito crítico – o estudante pode ter acesso a várias possíveis soluções de modo a facilitar sua tomada de consciência sobre o processo de resolução de cada classe de problemas.

O objetivo desta abordagem é acompanhar a aprendizagem de programação de modo individual, recomendando exercícios que auxiliem o processo de aprendizagem de cada estudante. A estratégia pedagógica desta arquitetura funciona da seguinte forma:

- ✓ O aluno escolhe dentre os assuntos disponibilizados pelo professor, o problema a ser resolvido;
- ✓ O aluno desenvolve uma solução para o problema proposto;
- ✓ Caso o aluno não consiga desenvolver uma solução para o problema proposto, será recomendado outro exercício equivalente;
- ✓ Se o aluno não conseguir solucionar o exercício equivalente, outro exercício equivalente e parcialmente resolvido será recomendado e o estudante terá a possibilidade de completar e testar a solução.

### 4.4.1 Participantes da Recomendação de Exercícios

Os participantes da arquitetura pedagógica Recomendação de Exercícios são professores e alunos de uma disciplina inicial de programação. Os alunos desenvolvem soluções para os problemas propostos e as enviam via os recursos de um ambiente virtual de aprendizagem. O professor, por sua vez, cadastra problemas de programação, classificados por equivalência, alguns não resolvidos e outros parcialmente ou totalmente resolvidos, para serem usados no processo de recomendação de exercícios que exibiremos a seguir.

### 4.4.2 Etapas da Recomendação de Exercícios

As etapas para a Recomendação de Exercícios são constituídas das seguintes etapas:

- ✓proposta de atividades;
- ✓desenvolvimento de soluções;
- ✓envio das soluções propostas;
- ✓recomendação de exercícios;

Na primeira etapa - *proposta de atividades*, o sistema de recomendação propõe atividades aos alunos, com base no perfil do estudante. Os alunos por sua vez, desenvolvem soluções para os problemas propostos (segunda etapa) passando pelas etapas do método *MCP*, descrito anteriormente.

Na terceira etapa - *envio das soluções propostas*, o aluno envia as soluções das atividades propostas, por meio de um ambiente de aprendizagem. A quarta e última etapa *Recomendação de exercícios*, é feito uma análise das soluções enviadas e é recomendada outra atividade, com base no desempenho obtido pelo aluno, na resolução do problema apresentado a ele.

Caso o aluno encontre dificuldades em resolver o problema recomendado, será recomendado outro exercício equivalente. Se o exercício equivalente não for solucionado, outro exercício equivalente e parcialmente resolvido será recomendado e o estudante terá a possibilidade de completar e testar a solução. Caso o aluno, não consiga resolver o exercício com a solução parcialmente resolvida, outro exercício equivalente com solução completa/correta ou com poucos erros sintáticos, será proposto para que o aluno teste e, eventualmente, corrija a solução. Essas alternativas somente serão concluídas com a entrega de um relatório sobre o processo de construção da solução.

### 4.4.3 Recursos tecnológicos da Recomendação de Exercícios

Os recursos tecnológicos usados na arquitetura pedagógica Recomendação de exercícios foram:

- ✓ recomendador de atividades;
- ✓ analisador de soluções;
- ✓ apresentador de atividades;
- ✓ recebedor de soluções;

- ✓ publicador de soluções;
- ✓ cadastrador de exercícios;
- ✓ apresentador e removedor de exercícios;
- ✓ cadastrador de assuntos;
- ✓ apresentador e removedor de assuntos;
- ✓ acompanhador de aprendizagem;
- ✓ configurador de aprendizagem;
- ✓ enviador de mensagens.

### 4.4.4 Síntese das arquiteturas pedagógicas

Nesta seção, é apresentada a Tabela 10, que apresenta uma síntese dos dados considerados mais importantes das arquiteturas pedagógicas.

Tabela 10 – Síntese das Arquiteturas Pedagógicas

| **Arquiteturas Pedagógicas** | **Objetivo Pedagógico** | **Estratégias Pedagógicas** | **Recursos digitais** |
|---|---|---|---|
| ***CSPL*** | explorar a aprendizagem de programação de forma individual e colaborativa. | ✓ O professor cadastra problemas para a criação de listas de exercícios;<br>✓ O professor cria a lista de exercícios com base nos problemas cadastrados;<br>✓ O professor define se as listas de exercícios serão desenvolvidas pelos alunos de forma individual ou em grupo;<br>✓ Os alunos acessam o ambiente de aprendizagem e acessam as listas de exercícios disponibilizadas pelo professor;<br>✓ Os alunos desenvolvem as listas de exercícios disponibilizadas pelo professor; | ✓ distribuidor de tarefas, individuais ou em grupo;<br>✓ apresentador de listas de exercícios;<br>✓ apresentador de problemas;<br>✓ apresentador de etapas (compreensão do problema, especificação da solução, codificação da solução e especificação da solução);<br>✓ recebedor de etapas;<br>✓ compilador de soluções;<br>✓ comparador do plano de teste com a solução executada;<br>✓ acompanhador de aprendizagem. |
| **Programação em Pares** | explorar a aprendizagem de programação em pares e desenvolver o espírito crítico no aluno | ✓ Cada aluno desenvolve uma solução para o problema proposto;<br>✓ São formados os pares automaticamente que obedeça as restrições estabelecidas pelo professor (percentual mínimo de diferença ou semelhança entre as soluções desenvolvidas);<br>✓ Cada aluno revisa a solução do seu par; | ✓ apresentador de problema;<br>✓ recebedor de solução;<br>✓ medidor da diferença entre duas soluções;<br>✓ comentador da solução do colega;<br>✓ visualizador de comentários;<br>✓ publicador da solução colaborativa;<br>✓ visualizador de soluções em pares; |

| | | | |
|---|---|---|---|
| | | ✓ O par de alunos gera uma nova solução diferente das duas soluções criadas anteriormente. | |
| **Recomendação de exercícios** | acompanhar a aprendizagem de programação de forma individual, recomendando exercícios que auxilie no processo de aprendizagem. | ✓ O aluno escolhe dentre os assuntos disponibilizados pelo professor, o problema a ser desenvolvido;<br>✓ O aluno desenvolve a solução para o problema proposto;<br>✓ Caso o aluno não consiga desenvolver a solução do problema proposto, será recomendado outro exercício equivalente;<br>✓ Se o aluno não conseguir solucionar o exercício equivalente, outro exercício equivalente e parcialmente resolvido será recomendado e o estudante terá a possibilidade de completar e testar a solução. | ✓ distribuidor de atividades recomendadas;<br>✓ analisador de soluções;<br>✓ apresentador de atividades;<br>✓ recebedor de soluções;<br>✓ publicador de soluções;<br>✓ cadastrador de exercícios;<br>✓ apresentador e removedor de exercícios;<br>✓ cadastrador de assuntos;<br>✓ apresentador e removedor de assuntos;<br>✓ acompanhador de aprendizagem;<br>✓ configurador de aprendizagem; |

## 4.5 CONCLUSÕES

A necessidade de personalizar o relacionamento com os alunos está cada vez mais crescente. Este capítulo apresentou a especificação de um ambiente de aprendizagem de programação, construído para dar suporte a arquiteturas pedagógicas para aprendizagem de programação de computadores, no intuito de potencializar a aprendizagem de programação.

# 5 PROTÓTIPO

*"A resposta certa, não importa nada: o essencial é que as perguntas estejam certas."*

***Mário Quintana***
*"As Indagações"*

Neste capítulo serão analisados os aspectos referentes ao suporte computacional para atender às arquiteturas pedagógicas descritas no capítulo IV. Em um protótipo computacional é feito a utilização das Arquiteturas Pedagógicas: *CSPL,* Programação em Pares e Recomendação de Exercícios, para avaliar se os requisitos funcionais foram implementados nos recursos digitais que instrumentalizam as arquiteturas pedagógicas que constituem o *AAP* – Ambiente de Aprendizagem de Programação de Computadores.

A Figura 06 apresenta o Mapa de Navegação do ambiente de aprendizagem de programação de computadores.

Um mapa de navegação, nada mais é que um mapa da arquitetura *web* que inclui todas as páginas e seções de conteúdo. Este por sua vez, é o guia para o desenvolvimento do projeto, desde a criação de *wireframes* até o seu desenvolvimento.

Um *Wireframe* é uma ilustração semelhante ao *layout* de elementos fundamentais na interface, sendo concluídos antes que qualquer trabalho artístico seja desenvolvido. A Figura 07 mostra a integração das arquiteturas pedagógicas no ambiente de aprendizagem.

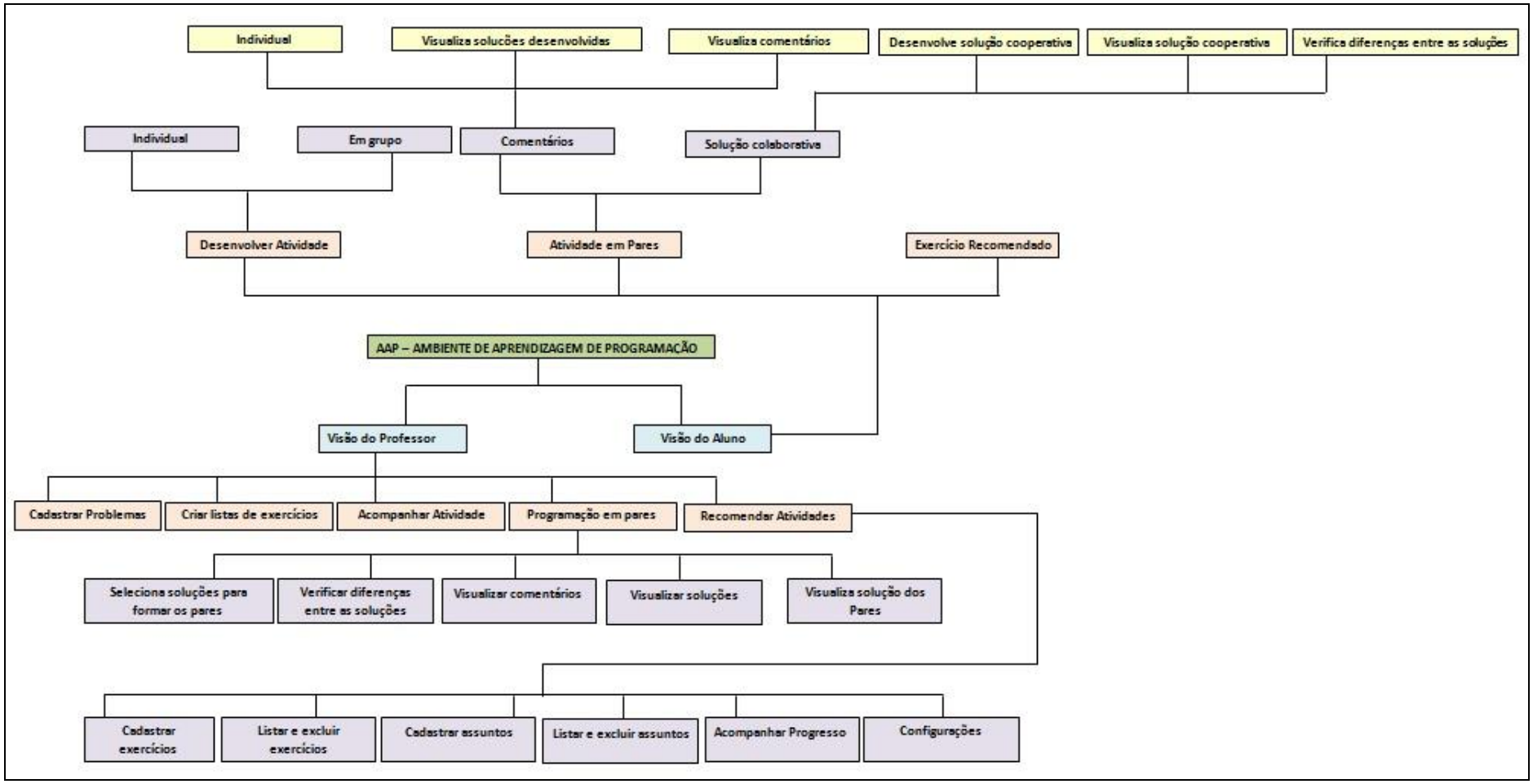


Figura 06 - Mapa de Navegação do ambiente de aprendizagem

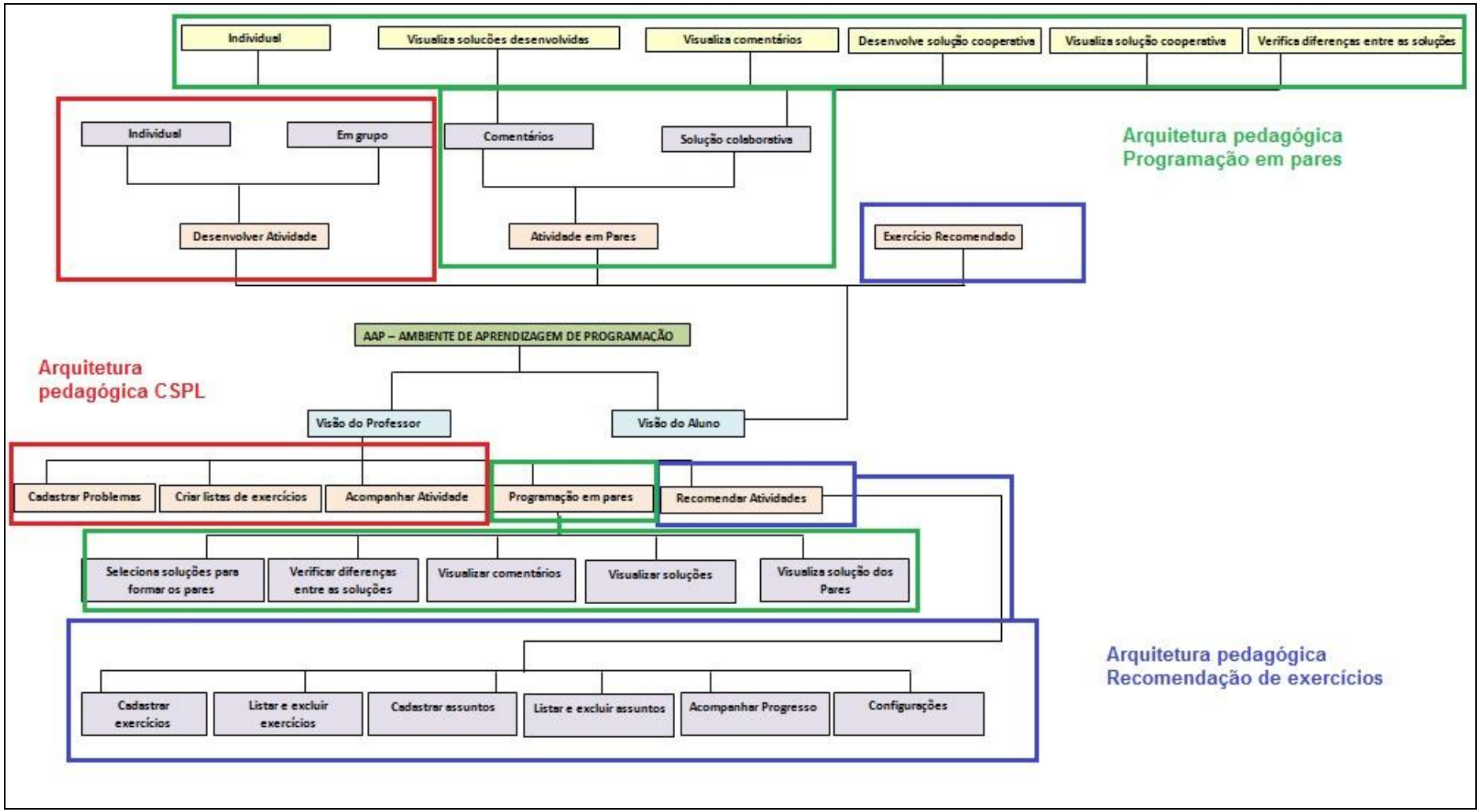


Figura 07 – Identificação das Arquiteturas pedagógicas no ambiente de aprendizagem

A Figura 07 apresenta de forma mais detalhada a apresentação das arquiteturas pedagógicas inseridas no ambiente de aprendizagem. Na cor vermelha é mostrado o espaço que abrange a Arquitetura pedagógica *CSPL* (*Computer Supported Programming Learning*).

A cor verde destaca os componentes da Arquitetura pedagógica Programação em Pares. Em azul, são mostrados os componentes da Arquitetura pedagógica Recomendação de exercícios.

As arquiteturas pedagógicas apresentadas nas ilustrações acima serão detalhadas nas seções posteriores.

A organização deste capítulo se dá conforme a ordem a seguir: na Seção 5.1 são apresentados os recursos digitais necessários para suporte às arquiteturas pedagógicas. A Seção 5.2 trás os requisitos e o suporte computacional utilizados para cada arquitetura pedagógica. Na Seção 5.3 é explicado as tecnologias usadas. A Seção 5.4 encerra este capítulo com as considerações finais.

## 5.1 Recursos digitais para suporte a arquiteturas pedagógicas

As tecnologias da informação e comunicação estão criando novas formas de distribuir socialmente o conhecimento (Salgado & Amaral, 2008). Isso implica em novas formas de ensinar e aprender.

Na defesa desta visão Levy (1998) defende que o professor é o responsável por provocar o “aprender a pensar”. As novas tecnologias estão aí para ajudá-lo nesse processo.  Em prol da aprendizagem, FOSCH( 2010) afirma que o “aprender a pensar” possibilita a construção e o compartilhamento do conhecimento.

Para suporte às abordagens pedagógicas que constituem as três arquiteturas pedagógicas propostas nesta dissertação, foram usados os seguintes recursos digitais:

*Distribuidor de tarefas individuais ou em grupos* – Este recurso digital, faz a distribuição das listas de exercícios criadas pelo professor, aos alunos via um ambiente virtual de aprendizagem *(AVA).* As listas de exercícios são previamente selecionadas pelo docente e disponibilizadas para os alunos.

*Apresentador de listas de exercícios* – As listas de exercícios criadas são apresentadas na visão do aluno para que o mesmo possa optar por alguma delas para dar início à criação de soluções. Em cada lista apresentada é possível visualizar o enunciado de cada problema que compõe a lista.

*O apresentador de problema* – O apresentador de problema mostra para o aluno o enunciado de cada problema proposto na lista de exercícios, para que ele possa escolher o problema que começará a resolver.

*Apresentador de etapas* – durante a solução de um problema proposto, o aluno passa por etapas de construção da solução. Este recurso digital possui a função de apresentar ao estudante, as etapas que ele precisa passar para desenvolver a solução desejada.

*Recebedor de etapas* – O recebedor de etapas tem a função de receber cada informação descrita pelo aluno nas etapas propostas pelo apresentador de etapas.

*Avaliador de soluções* – O avaliador de soluções faz a avaliação (interpretação) dos códigos desenvolvidos pelos alunos e informa para eles, se a solução desenvolvida foi avaliada com sucesso ou se teve algum erro.

Caso ocorra algum erro na interpretação dos códigos, os mesmos são apresentados para os estudantes, para que eles tomem ciência do erro ocorrido e tentem solucioná-lo.

*Comparador do plano de teste com a solução executada* – faz a comparação dos resultados esperados, do plano de teste elaborado pelo estudante, com os resultados obtidos pelo avaliador de soluções.

*Acompanhador de aprendizagem* – O acompanhador de aprendizagem é um recurso digital que auxilia o professor no acompanhamento das atividades desenvolvidas pelos alunos, apresentando ao professor as etapas da construção de soluções desenvolvidas pelos estudantes.

*Medidor da diferença e semelhança entre duas soluções* – o medidor da diferença entre as soluções mede a porcentagem de diferença ou semelhança entre as soluções desenvolvidas pelos alunos.

*Comentador de soluções* - este recurso digital permite que o aluno, faça comentários sobre a solução desenvolvida pelo seu colega, apontando os pontos fortes e os pontos fracos da solução desenvolvida.

*Publicador da solução* - o publicador de soluções permite ao aluno publicar, em seu ambiente de aprendizagem, a solução desenvolvida para um determinado problema proposto.

*Distribuidor de atividades recomendadas* – O recurso digital Distribuidor de atividades recomendadas, faz a distribuição de atividades recomendadas para o aluno, com base em seu perfil de aprendizagem.

*Construtor de solução individual/colaborativa* – É de suma importância o incentivo à colaboração entre os alunos na resolução de problemas, pois esta é uma condição essencial para aprendizagem. Neste sentido foi usado um construtor de soluções para que os alunos pudessem desenvolver as soluções dos problemas propostos de forma individual ou colaborativa.

*Formador de pares* – Este recurso digital forma os pares com base no percentual de diferenças ou semelhanças entre as soluções. Este percentual é escolhido pelo professor, antes da formação dos pares.

*Analisador de soluções recomendadas* - este recurso analisa as soluções desenvolvidas pelos alunos, para posteriormente recomendar outras atividades que possam auxiliar o aluno em seu aprendizado.

*Cadastrador de Exercícios* - este recurso cadastra os exercícios que serão recomendados aos alunos, de acordo com o aprendizado apresentado.

*Apresentador e removedor de exercícios* – com este recurso o professor pode listar ou remover todos os exercícios cadastrados para recomendação.

*Cadastrador de Assuntos* - este recurso digital faz o cadastro de assuntos nos quais os exercícios serão classificados.

*Apresentador e removedor de assuntos* – com este recurso digital o professor pode listar todos os assuntos cadastrados, como também remover os assuntos cadastrados.

*Configurador de aprendizagem* – o configurador de aprendizagem, trás a possibilidade de configurar a quantidade de problemas de um mesmo assunto que devem ser resolvidos corretamente para que o aluno possa passar para outro assunto recomendado.

*Enviador de mensagens* - este recurso envia para o professor uma mensagem informando que um aluno terminou de inserir um comentário ou terminou de enviar uma solução desenvolvida.

*Visualizador de mensagens* – Com este recurso o professor consegue visualizar as mensagens enviadas pelo enviador de mensagens.

## 5.2 Requisitos e o suporte tecnológico

As atividades propostas pelas Arquiteturas Pedagógicas visam criar situações de aprendizagem de programação de computadores, algumas atividades são individuais e outras colaborativas.

### 5.2.1. Suporte à distribuição de atividades

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como professor de Programação I.

**Suporte tecnológico:** Os suportes tecnológicos usados nesta etapa são: cadastro de problemas e criador de listas de exercícios. As Figuras 08 e 09 mostram os locais para o cadastramento de problemas e a criação das listas de exercícios.

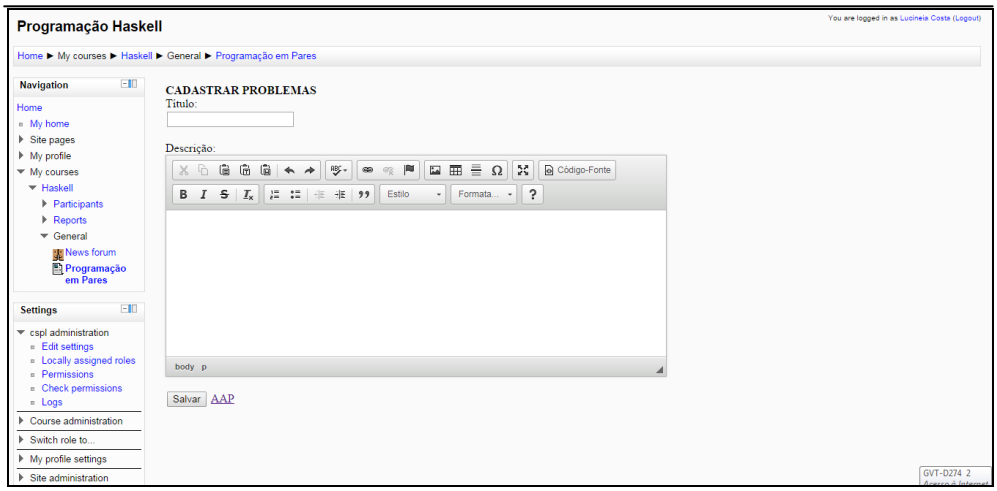


Figura 08. Cadastro de problemas

A Figura 08 apresenta o local em que são cadastrados os problemas a serem desenvolvidos pelos alunos. Neste cadastro são inseridos o título do problema e a descrição do mesmo.

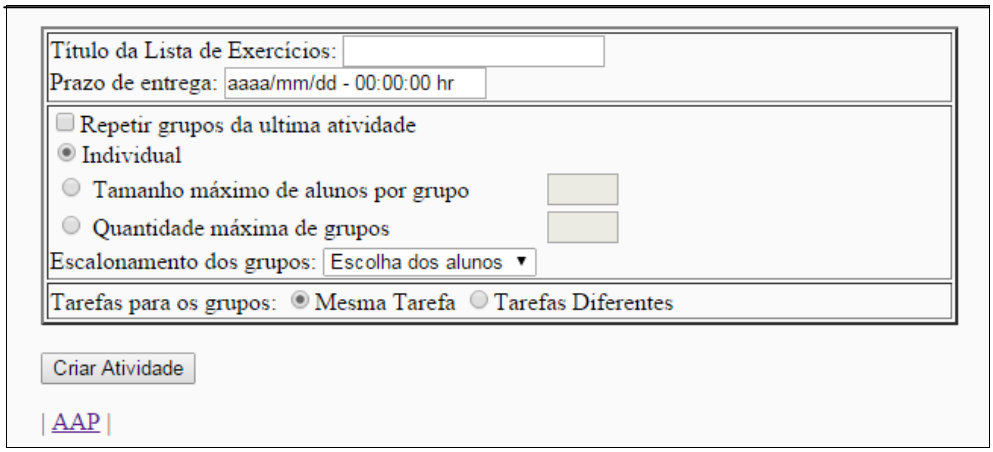


Figura 09. Criação das listas de exercícios

Na Figura 09 mostra a tela da criação das listas de exercícios. Nela é solicitado ao professor que insira informações como:

*Titulo da lista de exercícios:* Neste campo o professor insere o título que dará a nova lista de exercícios a ser criada.

*Prazo de entrega:* onde é informado o prazo que os alunos terão para desenvolver as atividades contidas nesta lista de exercícios. Este prazo é formado pelo dia e a hora em que os alunos precisam entregar as tarefas.

*Repetir grupos da última atividade***:** Este campo deverá ser marcado somente se o professor for enviar a lista de exercícios para os mesmos grupos formados na atividade anterior.

*Individual:* Esta opção é usada quando o professor for criar uma lista de exercícios para ser desenvolvida de forma individual.

Os campos: *Total máximo de alunos por grupo* e *Quantidade máxima de grupos***,** são usados somente se o professor optar por enviar as atividades para serem desenvolvidas em grupo. Neste caso, o *Total máximo de alunos por grupo* indica a quantidade máxima de alunos que um grupo pode ter, para que seja formado e a opção *quantidade máxima de grupos* que indica o número

máximo de grupos que devem ser formados para o desenvolvimento desta lista de exercícios.

A opção *Escalonamento de grupos* informa se os grupos serão formados pela escolha do próprio aluno ou de forma aleatória. Caso o professor faça a opção do próprio aluno escolher o grupo, ao acessar o ambiente, o aluno vê os grupos propostos pelo professor para que possa escolher um desses grupos.

Em *Tarefas para grupos*, temos duas opções: *Mesma tarefa* e *Tarefas diferentes*. Em "*Mesma tarefa*", o professor pode optar por enviar a mesma lista de exercícios para todos os grupos. Já em *Tarefas diferentes* o docente tem a possibilidades enviar atividades diferentes para cada grupo. A Figura 10 apresenta o local em que o professor faz a escolha dos problemas para a criação da lista de exercícios.


Figura 10. Escolha dos problemas

Após o cadastro dos problemas, o professor pode criar a lista de exercícios com os problemas já cadastrados. A Figura 10, apresentada acima, mostra a

forma como os problemas são apresentados ao professor, para que eles sejam inclusos na lista.

O docente pode optar em criar uma lista de exercício com apenas um problema, ou criar uma lista contendo vários problemas. A escolha das atividades contidas nessas listas fica a cargo do professor, que pode criá-las de acordo com o avanço da aprendizagem dos alunos.

### 5.2.2. Suporte à visualização de atividades

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como aluno de Programação I.

**Suporte tecnológico:** Os suportes tecnológicos usados nesta etapa são: distribuidor de tarefas individuais ou em grupos, apresentador de listas de exercícios, apresentador de problema.

**Protótipo**

A Figura 11 apresenta a visualização do aluno no ambiente virtual de aprendizagem. Nela são apresentadas as listas de exercícios sugeridas pelo professor, para que sejam desenvolvidas pelos alunos.

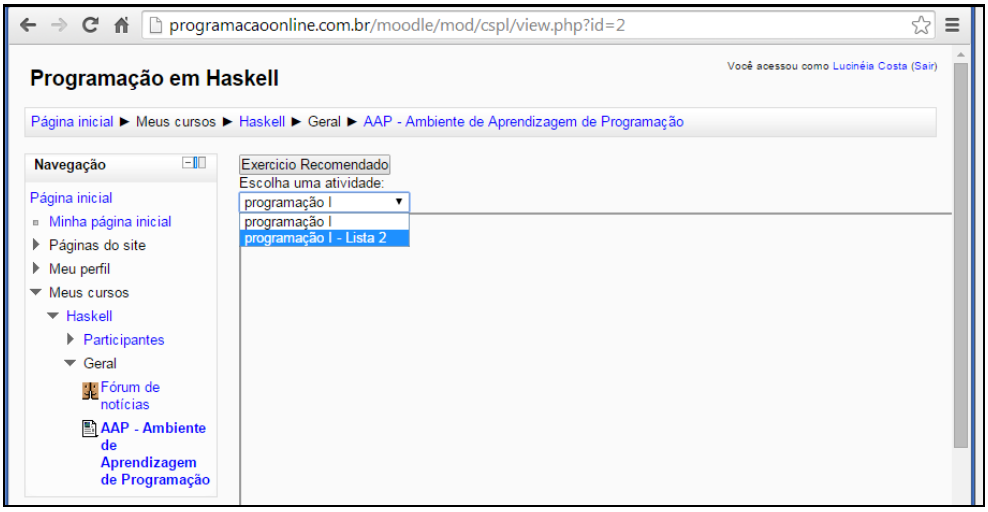


Figura 11. Visão do aluno

O estudante, por sua vez, tem a possibilidade de escolher qual das listas de exercícios apresentadas começará a desenvolver. Esta escolha é feita clicando sobre o enunciado da lista escolhida. Feita a escolha da lista de exercícios, o aluno pode optar, dentre os problemas contidos na lista, aquele que começará a solucionar conforme apresenta a Figura 12.

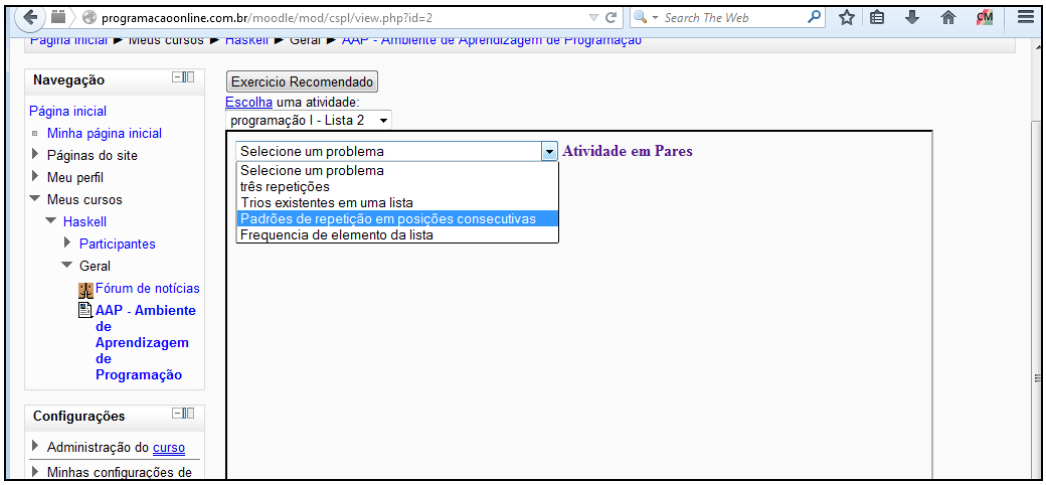


Figura 12. Escolha dos problemas a serem resolvidos

Em seguida o aluno é reportado ao local em que começará a desenvolver as atividades, como: a compreensão do problema proposto, a especificação da solução, a codificação da solução e o plano de teste.

Para que o requisito da visibilidade seja atendido é necessário que o aluno entre no ambiente virtual de aprendizagem, através de seu *login* e senha. Após o acesso será apresentado as atividades propostas no qual deverá desenvolver, através dos recursos tecnológicos descritos anteriormente.

### 5.2.3. Suporte ao desenvolvimento de atividades

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como aluno de Programação I.

**Suporte tecnológico:** Os suportes tecnológicos usados nesta etapa são: apresentador de etapas, recebedor de etapas, avaliador de soluções e comparador entre o resultado esperado no plano de teste com o resultado obtido. A Figura 13 ilustra o local no qual cada aluno pode editar como ele compreendeu o problema proposto pelo professor.

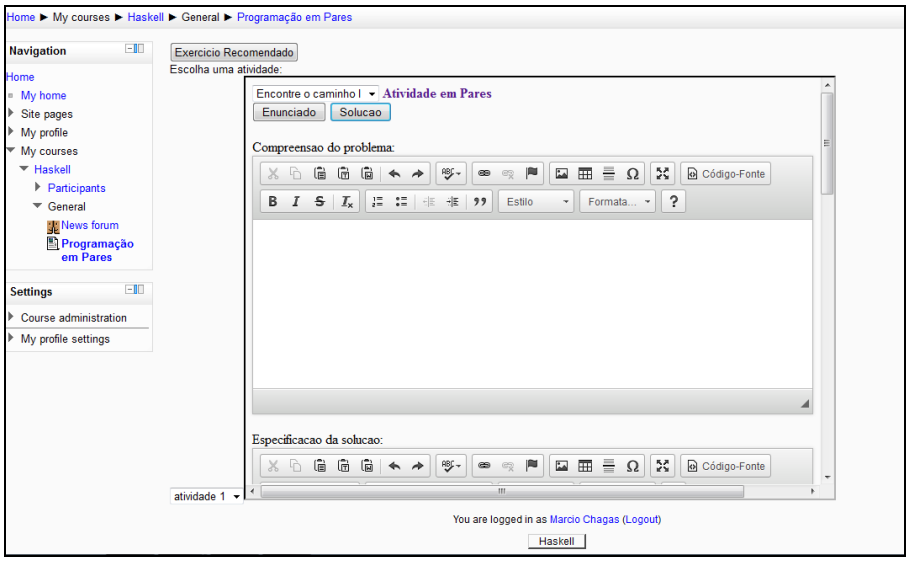


Figura 13. Etapa Compreensão do problema

A próxima etapa a ser desenvolvida pelo aluno é a especificação da solução. Nela o estudante descreve a especificação da solução a partir da composição

dos métodos e funções de mapeamentos usados no plano de teste, como apresenta a Figura 14.

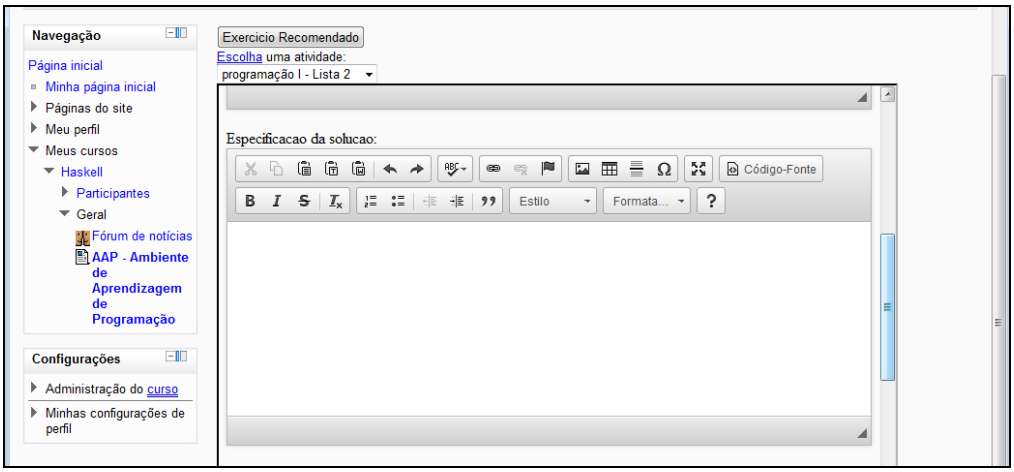


Figura 14. Etapa Especificação da solução

A Etapa a seguir é a codificação da solução, quando o aluno descreve o código da solução proposta. A Figura 15 mostra o local em que o estudante insere o código da solução. O código inserido nesse local é salvo em um arquivo “.txt” para posteriormente ser usado pelo interpretador da linguagem e fazer a comparações entre o resultado esperado e o resultado obtido.


Figura 15. Etapa de codificação

Inserido o código da solução, o aluno pode fazer os testes do plano de testes. O plano de teste é o resultado da etapa em que o estudante planeja o teste da solução especificada e codificada. Para criar o plano de teste, o aluno deve inserir as informações de entrada e saída esperada na tabela de plano de teste, apresentada na Figura 16.

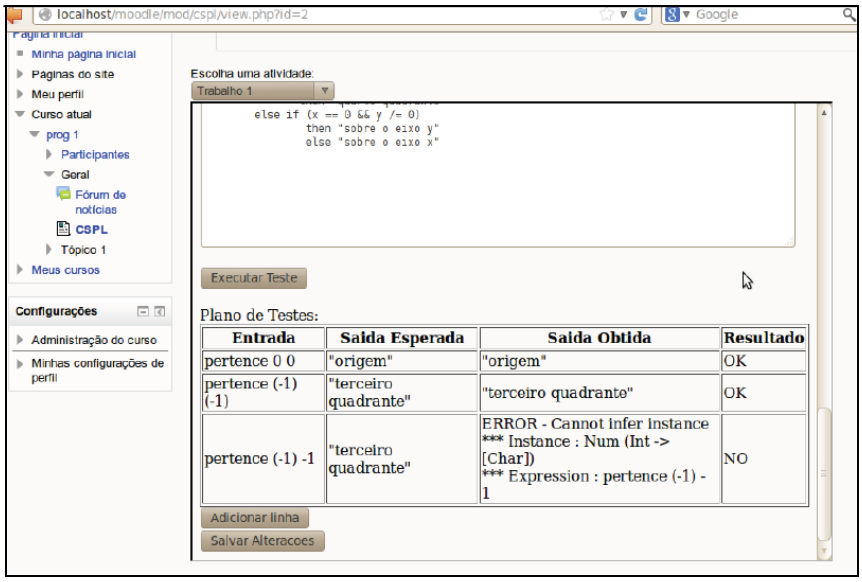


Figura 16. Etapa Planejamento do Teste

No campo destinado a *Entrada* é inserido o dado de entrada a ser processado e no campo de *Saída Esperada* é adicionado o resultado esperado para a avaliação da função (solução) sobre o dado de entrada da linha correspondente.

A *Saída Obtida* é o resultado do processamento do dado de entrada. Neste momento o interpretador da linguagem usa os dados de entrada digitados pelo aluno para fazer o processamento. Em seguida insere no campo Saída obtida, correspondente, o resultado obtido desse processamento.

O campo *Resultado* é o local usado para informar se a saída esperada pelo estudante confere com a saída obtida pelo interpretador da linguagem. Em caso positivo é escrito *OK* no campo resultado, informando que os resultados esperados e obtidos foram iguais. Em caso negativo é informado no campo de resultado *NO* que indica que as saídas esperadas e obtidas são diferentes.

### 5.2.4 Suporte a Análise de soluções e formação de pares

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como professor de Programação I.
**Suporte tecnológico:** Acompanhador de aprendizagem, Medidor da semelhança/diferença entre duas soluções e Formador de Pares.

A análise das soluções é feita através do modelo de espaço vetorial. O modelo de espaço vetorial representa cada documento como um vetor de termos e, cada termo possui um valor associado que indica seu grau de importância (BONFIN, 2009). Os *Termos* são ocorrências únicas nos documentos, neles são atribuídos pesos que especificam o tamanho e a direção de seu vetor de representação.

Os pesos são usados para calcular o grau de similaridade entre consulta e documento. Cada elemento do vetor de termos é considerado uma coordenada dimensional. Assim, os textos podem ser colocados em um espaço euclidiano de *n* dimensões e a posição do texto em cada dimensão é dada pelo seu peso (PAYSAN, 2006), conforme mostra a Figura 17.

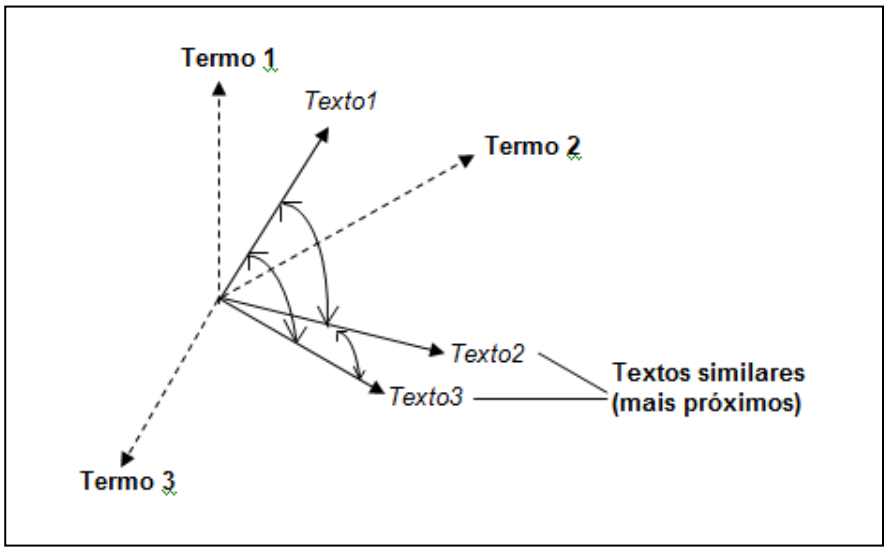


Figura 17. Espaço euclidiano com os textos em “n” dimensões, PAYSAN, 2006

Os textos que possuem os mesmos termos acabam colocados em uma mesma região do espaço, pois se tratam de assuntos similares. Para o cálculo *dos* pesos dos vetores é usado a fórmula do *TF-IDF* (*term frequency–inverse document frequency*). O *TF-IDF*, é uma estatística numérica que reflete o quão importante uma palavra é em um documento de uma coleção de documentos. (RAJARAMAN, 2011)

O valor *TF-IDF* aumenta proporcionalmente com o número de vezes que uma palavra aparece no documento, mas é compensada pela frequência da palavra na coleção de documentos, o que ajuda a controlar o fato de que algumas palavras são geralmente mais comuns do que outras. Abaixo são apresentadas as equações usadas para calcular o *TF-IDF.* Para calcular o termo de frequência, é usada a seguinte equação:

$$tf = \frac{freq_{ti,dj}}{max\ freq_{dj}} \qquad (1)$$

No qual $freq_{ti,dj}$ indica a frequência do termo $t_i$ no documento $d_j$ e $max\ freq_{dj}$ indica a frequência do termo de maior frequência no documento $d_j$. Quanto mais frequentemente um termo ocorrer no texto de um documento, maior será a sua frequência de termo. Para calcular o termo *IDF*, foi usado a seguinte equação:

$$idf = log\frac{N}{n_{ti}} \qquad (2)$$

No qual *N* é o número total de documentos da coleção e, $n_{ti}$ é o número de documentos da coleção que contêm o termo de indexação $t_i$. Após calculados os pesos dos termos, é feito o cálculo da medida de similaridade entre

documentos e consulta. Para este cálculo, fez uso da fórmula do *cosseno*. Apresentada a seguir:

$$similaridade(Q,D) = \frac{\sum_{k=1} W_{qk} \bullet W_{dk}}{\sqrt{\sum_{k=1}^{n} (W_{qk})^2 \bullet \sum_{k=1}^{n} (W_{dk})^2}} \quad (3)$$

Em que, *Q* é o vetor de termos da consulta, *D* é o vetor de termos do documento, $W_{qk}$ são os pesos dos termos da consulta e $W_{dk}$ são os pesos dos termos do texto. O modelo de espaço vetorial foi escolhido, devido a sua capacidade de mensurar um grau de similaridade parcial entre os documentos e pela não necessidade de técnicas probabilísticas avançadas para a recuperação de informação.

## Protótipo

A Figura 18 apresenta o local em que o professor faz a análise das soluções e a formação dos pares. Essa análise é feita de forma automática, basta acessar a opção “Programação de Pares”.


Figura 18. Análise e formação dos pares

Ao acessar no botão *Programação em Pares*, o professor é direcionado a um local, no qual pode:

- ✓ selecionar a lista de exercícios para a formação dos pares;
- ✓ verificar a diferença entre as soluções;
- ✓ visualizar os comentários;
- ✓ visualizar as soluções desenvolvidas
- ✓ visualizar as soluções desenvolvidas em pares;
- ✓ envio de arquivos;
- ✓ mensagens

A Figura 19 apresenta o local no qual o professor é direcionado, ao acessar a Programação em Pares.

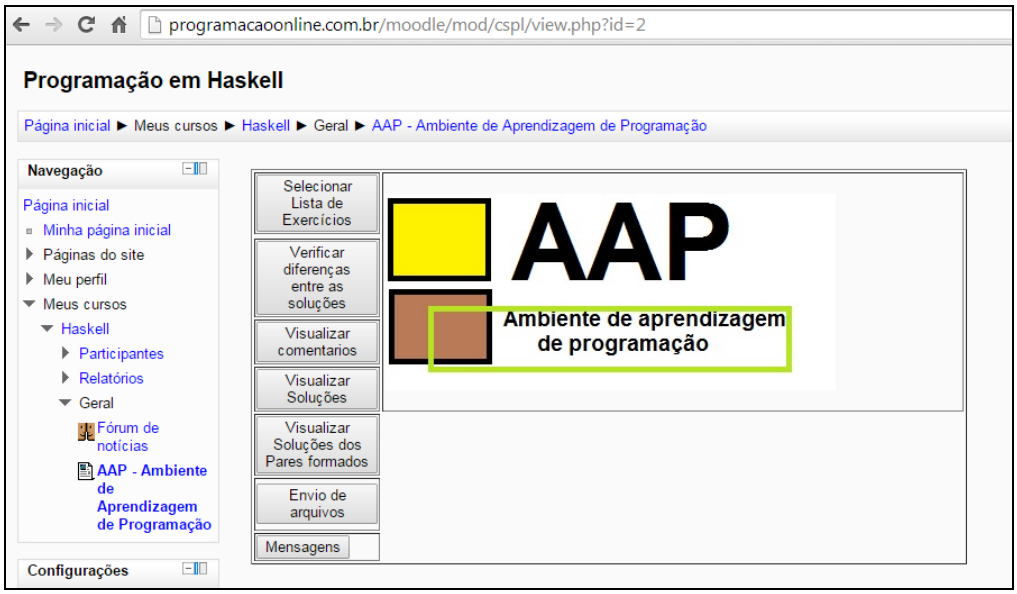


Figura 19. Programação em Pares

Ao acessar *Selecionar Lista de Exercícios* são apresentadas ao docente as atividades encontradas com as soluções desenvolvidas pelos alunos. Conforme apresenta a Figura 20.


Figura 20. Descrição das atividades encontradas

Ainda nesta tela o docente digita, no local apropriado, o nome da Lista de exercícios que deseja escolher para a formação dos pares, em seguida clica no botão *enviar*. Na próxima tela são apresentados ao professor os nomes dos problemas que já foram desenvolvidos dentro da lista escolhida. Conforme apresenta a Figura 21.

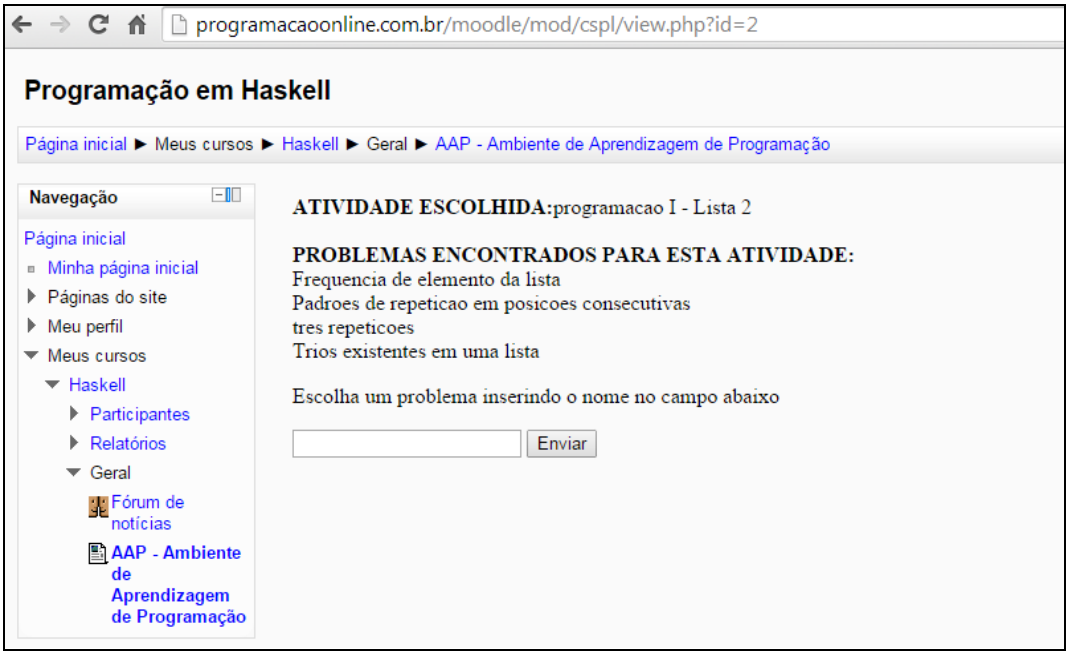


Figura 21. Descrição dos problemas encontrados

Ao escolher um dos problemas, é disponibilizado ao professor uma tela onde o docente indica como deseja formar os pares. Os pares podem ser formados tanto por semelhança como por diferença. Fica a critério do professor optar pela forma como os pares devem ser formados. A Figura 22 apresenta o local no qual o professor informa como os pares deverão ser formados.


Figura 22. Formação dos pares

Após o professor definir a formação dos pares e o percentual de diferença ou semelhança entre as soluções, são mostrados na visão do professor os pares formados, conforme ilustra a Figura 23.

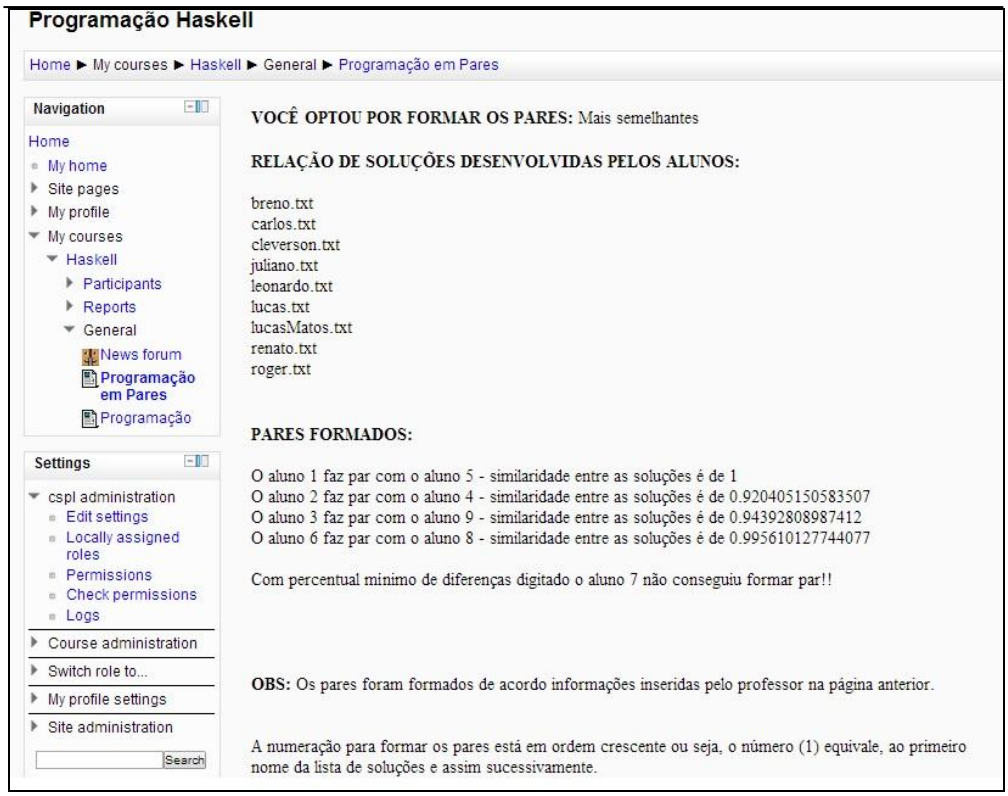


Figura 23. Pares formados

Na Figura 24 foi usada uma pequena amostra de nove soluções desenvolvidas pelos alunos de uma disciplina de Programação I. Para a formação dos pares o professor optou em formá-los por *Mais semelhantes*, com um percentual mínimo de semelhança de 70% entre as soluções.

Desta amostra foram formados quatro pares e apenas um aluno ficou sem par. Neste caso o professor pode optar em alocar este aluno a uma das duplas formadas ou sugerir ao aluno que desenvolva a atividade individualmente.

Para fazer a formação dos pares foi criado um algoritmo que busca na base dados as soluções e faz o processamento destas, retirando de cada solução as palavras reservadas da linguagem de programação e operadores lógicos. Em seguida com base na fórmula do *cosseno* apresentada anteriormente é formada a matriz de similaridade entre as soluções.

Com a matriz de similaridade gerada, é feita a formação dos pares com base no percentual de similaridade ou diferença digitado pelo professor. Para

calcular a similaridade, o algoritmo percorre a matriz buscando as soluções mais semelhantes, faz a comparação delas e forma os pares.

Para fazer o cálculo da diferença, o algoritmo percorre a matriz de similaridade e busca as soluções mais diferentes para formar os pares. Essa formação de pares varia de acordo com o percentual digitado pelo professor.

Por exemplo, se o professor buscar formar pares “mais semelhantes”, com o percentual de similaridades de 70%, o algoritmo irá percorrer a matriz de similaridade, buscando as soluções que sejam no mínimo de 70% mais semelhantes para formar os pares.

Caso o professor desejar formar pares mais diferentes com o percentual de no máximo 70% de diferença. O algoritmo irá percorrer a matriz de similaridade, buscando as soluções mais diferentes, com no máximo 70% de diferença entre as soluções.

Após a formação dos pares, é apresentada ao professor a formação dos pares, em um arquivo pdf, no qual o professor poderá fazer o *download*. Conforme mostra a Figura 24.

Neste exemplo, foi usado apenas 4 soluções desenvolvidas pelos alunos, para a formação dos pares. A formação escolhida foi a “Mais semelhantes”, com no mínimo 40% de semelhança.

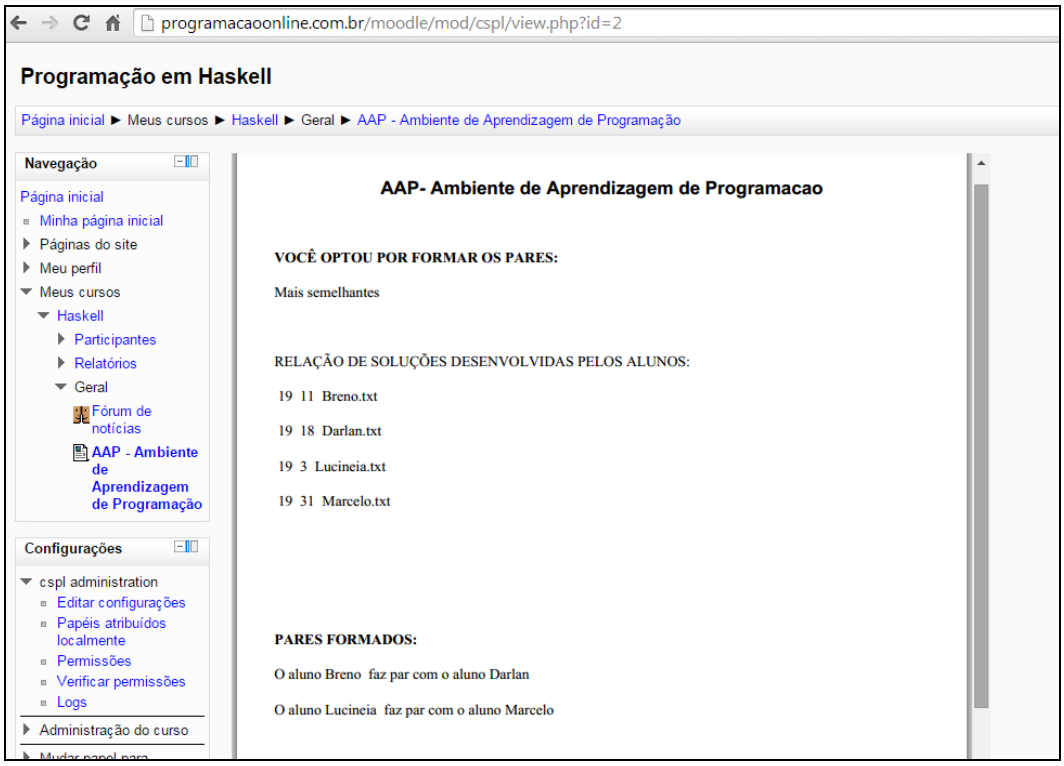


Figura 24.Visualização dos pares formados

Com o pdf gerado, o professor agora pode disponibilizar para os alunos um arquivo com a formação dos pares. A disponibilização deste arquivo pode ser feito no próprio ambiente, através do botão “envio de arquivos”, conforme apresenta a Figura 25.

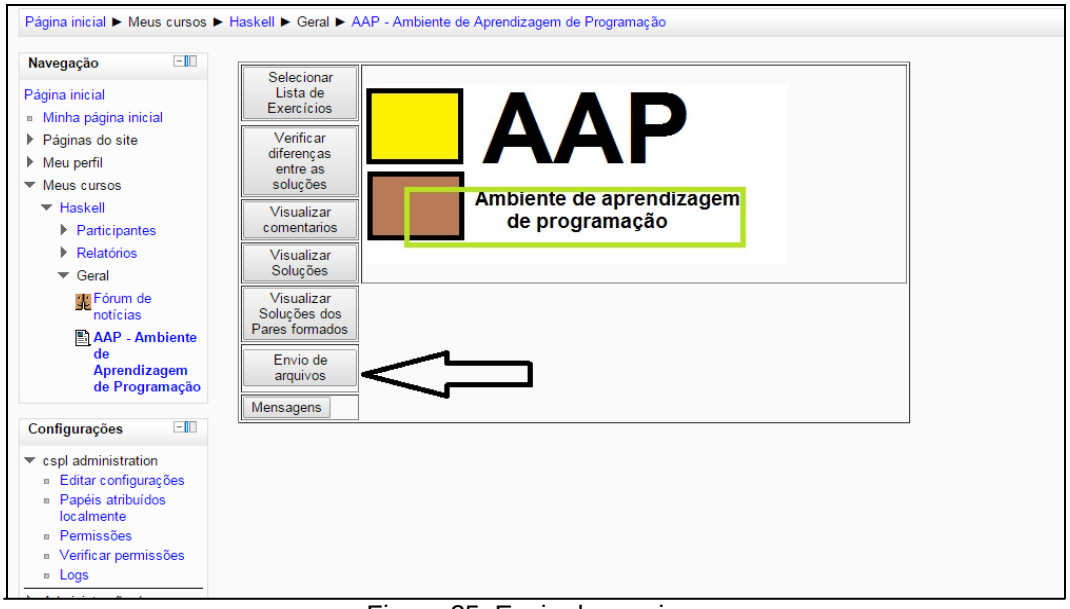


Figura 25. Envio de arquivos

### 5.2.5 Suporte a solução em Pares

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como aluno de Programação I.

**Suporte tecnológico:** Distribuidor de tarefas individuais ou em grupos, apresentador de listas de exercícios, apresentador de problema, apresentador de etapa de solução, recebedor de etapa de solução, avaliador de soluções, comentador da solução do colega, publicador da solução, construtor de soluções individual/colaborativa, verificador de diferenças, enviador de mensagens.

**Protótipo**

Após a formação dos pares, são apresentadas ao aluno três tipos de atividades: comentar as soluções de forma individual e criar uma solução cooperativa e verificar diferenças entre as soluções, além do aluno poder visualizar os pares formados. Conforme mostra a Figura 26.


Figura 26. Atividade em pares

Na atividade *Comentários de Soluções,* os alunos comentam as soluções que foram escolhidas. Neste comentário os alunos apontam as semelhanças e

diferenças que encontraram entre as soluções desenvolvidas. Conforme apresenta a Figura 27.

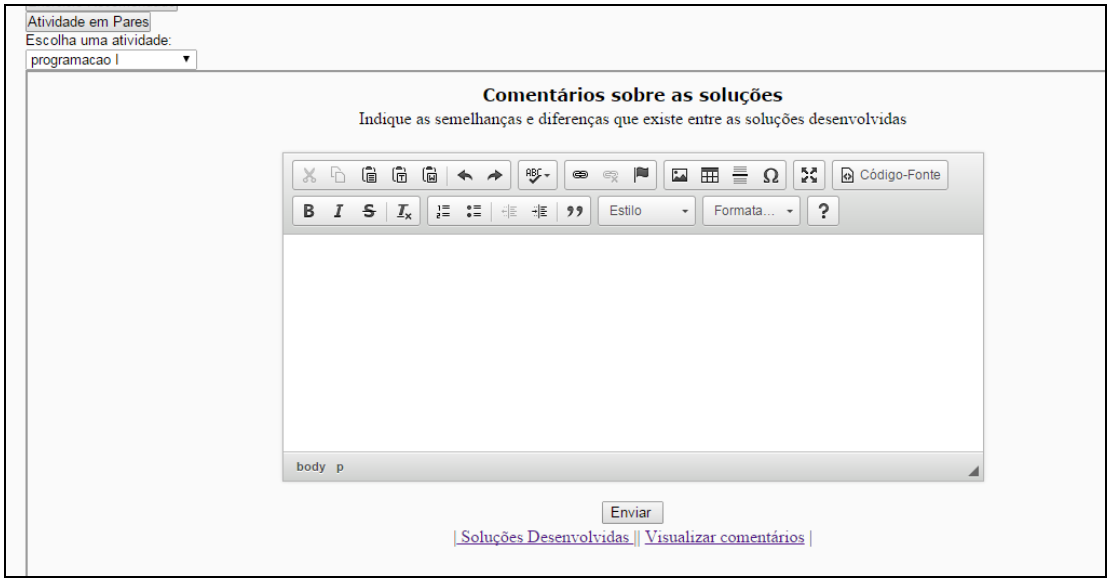


Figura 27. Comentador de soluções

Em *Solução cooperativa* é disponibilizado um local para que o par formado possa desenvolver a nova solução para o problema proposto anteriormente, levando em consideração que a nova solução precisa ter no mínimo 20% de diferença entre ela e as duas outras soluções criadas de forma individual, conforme ilustra a Figura 28.

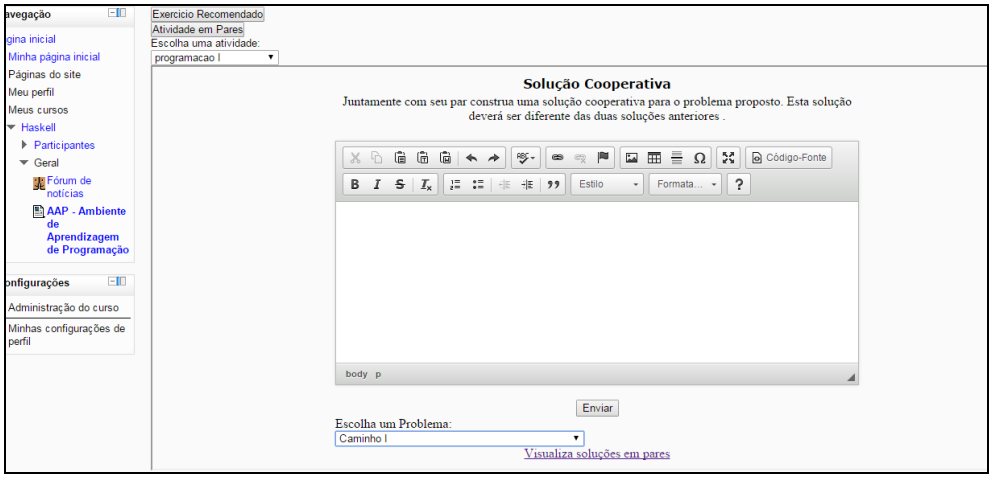


Figura 28. Solução Colaborativa

Na opção *Verificar Diferenças,* os pares submetem as soluções desenvolvidas de forma individual e a solução desenvolvida em pares para verificar se a solução desenvolvida de forma colaborativa possui um percentual de no mínimo 20% de diferenças entre as demais soluções, conforme apresenta a Figura 29.

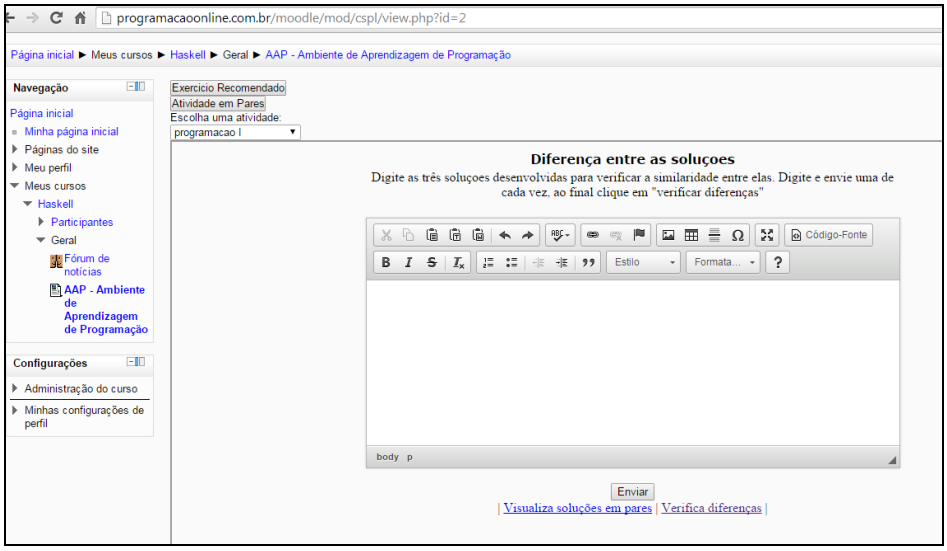


Figura 29.Diferença entre as soluções

Em *visualizar pares formados*, o aluno tem a possibilidade de visualizar e fazer o *upload* dos arquivos enviados pelo professor com os pares formados, conforme apresenta a Figura 30.

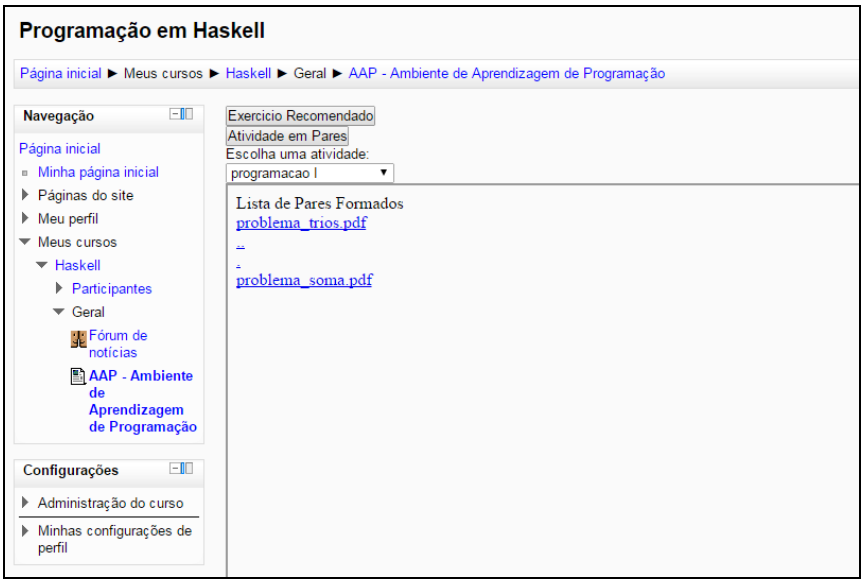


Figura 30. Lista de pares formados

### 5.2.6 Suporte ao envio de soluções colaborativas

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como aluno de Programação I e ter desenvolvido a solução de forma colaborativa.

**Suporte tecnológico:** codificador de solução, publicador da solução.

**Protótipo**

A Figura 31 apresenta o local onde os alunos descrevem a solução colaborativa para o problema proposto. Esta solução colaborativa é desenvolvida pelo par, com diferença mínima de 20%, em relação às soluções desenvolvidas de forma individual pelos membros do par.

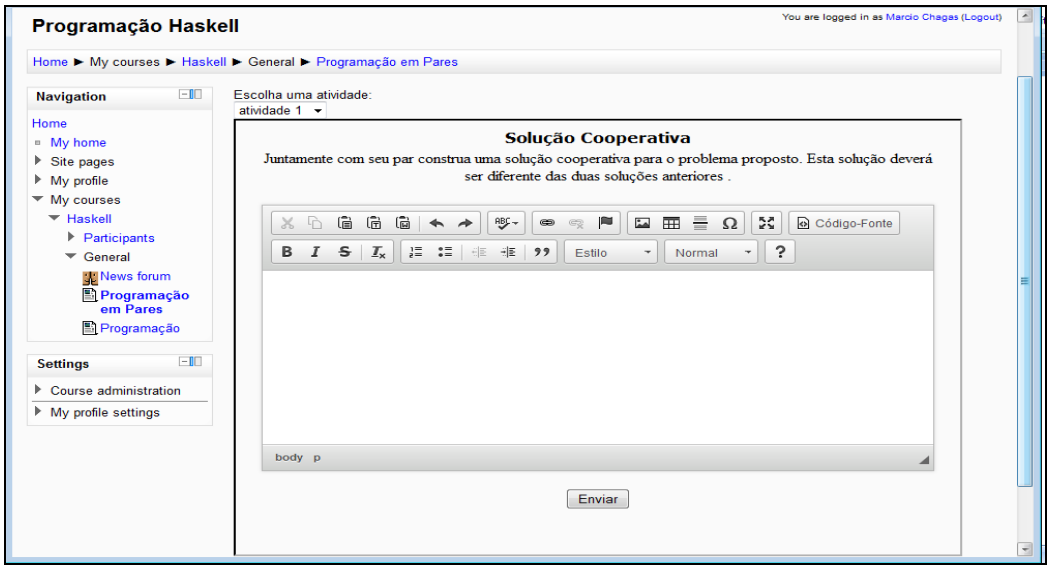


Figura 31. Desenvolvimento e envio das soluções em pares

### 5.2.7 Suporte à comparação entre as soluções

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como professor ou aluno de Programação I.

**Suporte tecnológico:** Medidor da diferença entre duas soluções.

Em ambas as visões: professor e aluno podem fazer a comparação entre a dois códigos. Para a comparação das soluções é usado um medidor de diferença entre as soluções.

Este medidor busca as soluções desenvolvidas pelos pares de forma individual e colaborativa e mede a diferença entre elas. Caso a solução de um aluno não possua o percentual de diferença de 20%, os alunos são convidados a refazer a solução colaborativa até que esta possua o percentual mínimo de diferença exigido para esta atividade.

### 5.2.8 Suporte a Recomendação de exercícios

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como professor de Programação I.

**Suporte tecnológico:** cadastrador de exercícios, apresentador e removedor de exercícios, cadastrador de assuntos, apresentador e removedor de assuntos, acompanhador de aprendizagem, configurador de recomendações e um analisador de soluções recomendadas.

**Protótipo**

A Figura 32 apresenta o local onde o professor faz o cadastro dos exercícios a serem recomendados.

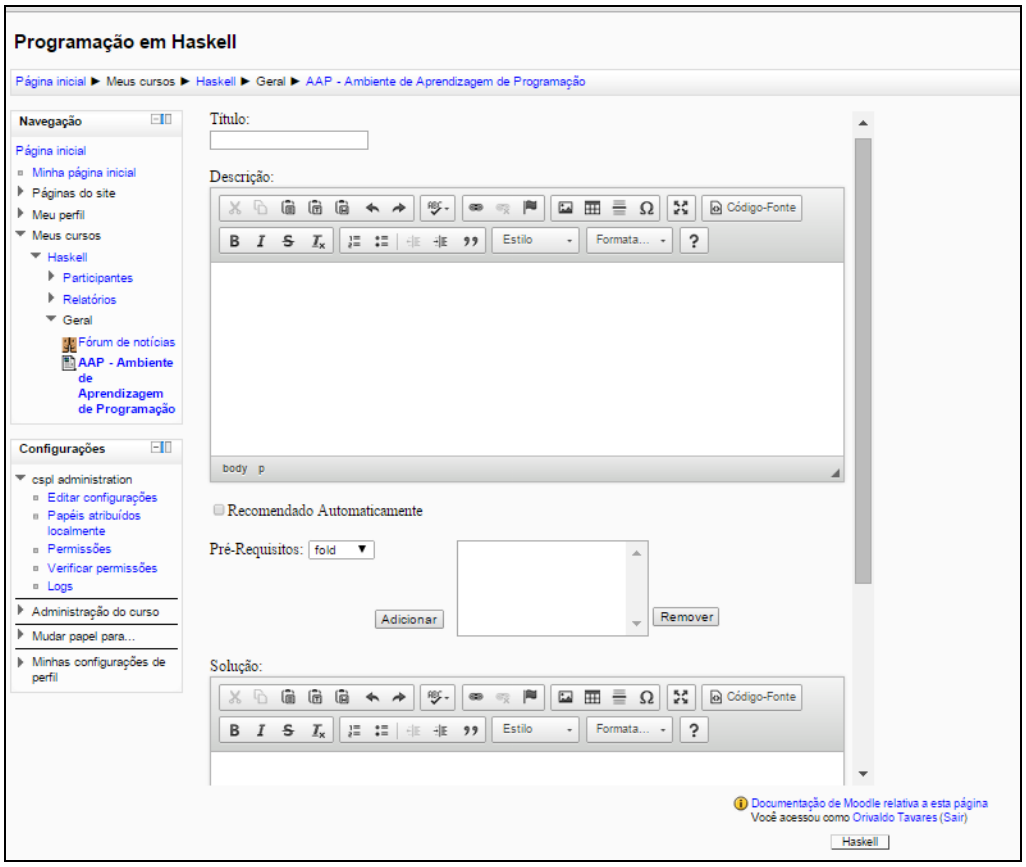


Figura 32. Cadastro de exercícios

Ao cadastrar o exercício para ser recomendado, o professor passa as seguintes informações ao ambiente:

- ✓ título do exercício;
- ✓ a descrição do exercício;
- ✓ informa se o exercício será recomendado de forma automática ou manualmente;
- ✓ indica os pré-requisitos que o aluno precisa ter para ingressar nesta atividade;
- ✓ informa se a atividade possui solução completa ou parcial.

Na Figura 33 é apresentada a lista de exercícios já cadastrados para serem recomendados pelo professor. Uma vez que os exercícios são cadastrados, eles também podem ser removidos, conforme ilustra a Figura 34.

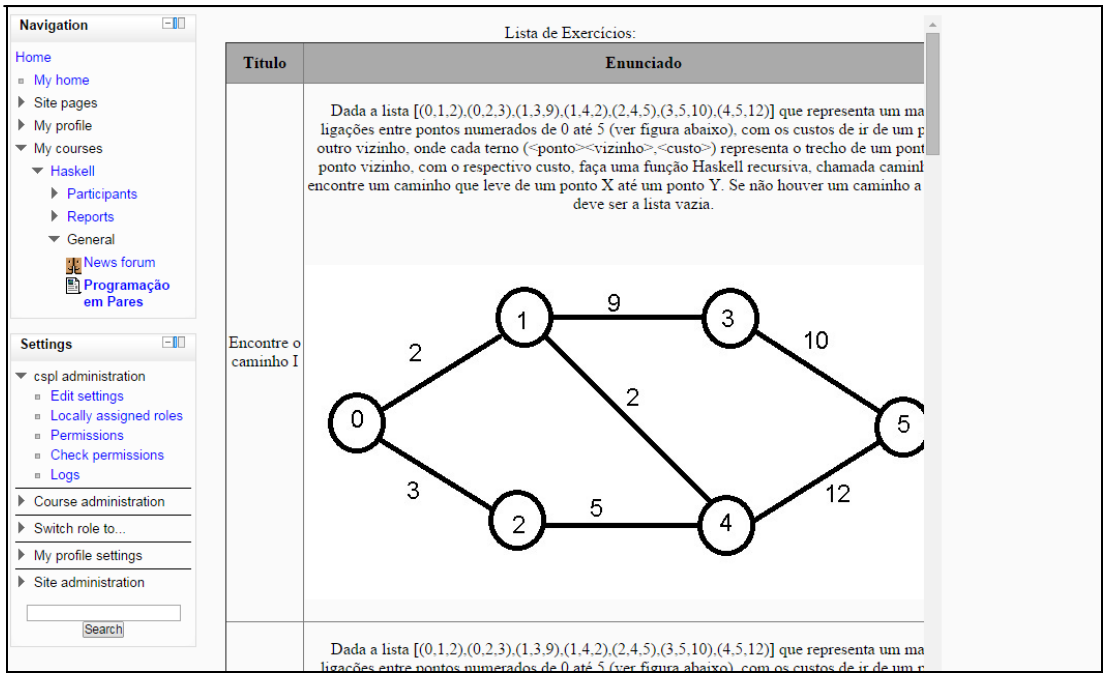


Figura 33. Lista os exercícios cadastrados

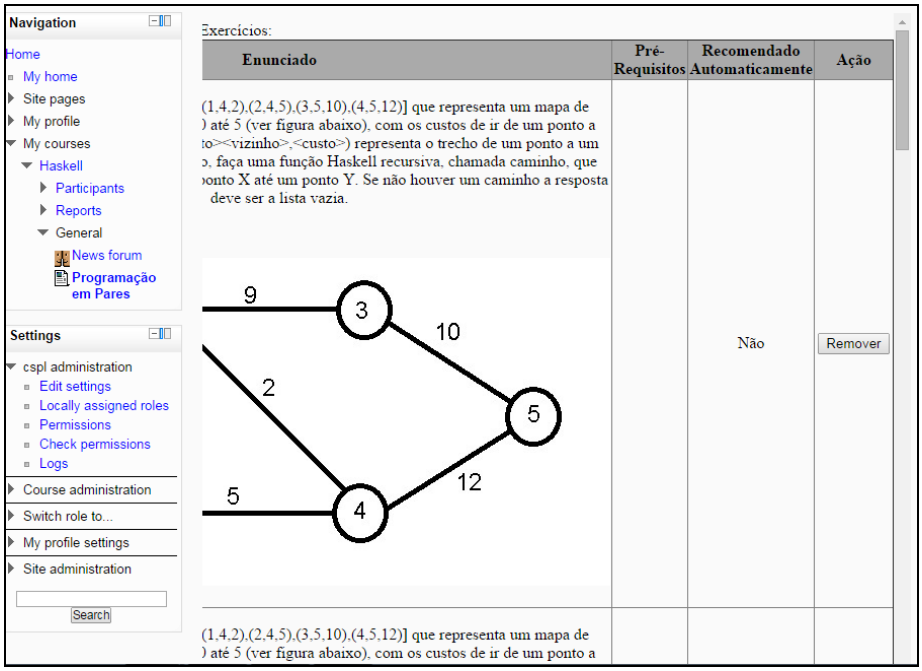


Figura 34. remoção do exercício a ser recomendado

A Figura 35 informa o local onde o professor faz o cadastramento dos assuntos. Nesse local são inseridas informações como o nome do assunto,

pré-requisitos para que o aluno possa desenvolver aquele assunto e a descrição do mesmo.

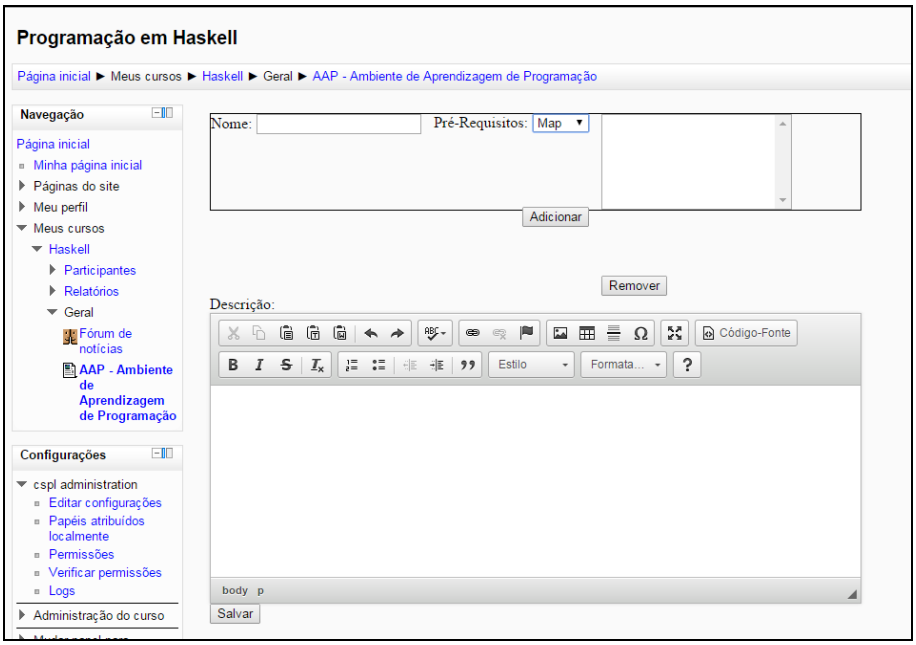


Figura 35. Cadastro de assunto

Ainda referente aos assuntos, o professor pode visualizar ou remover os assuntos, conforme apresenta a Figura 36.

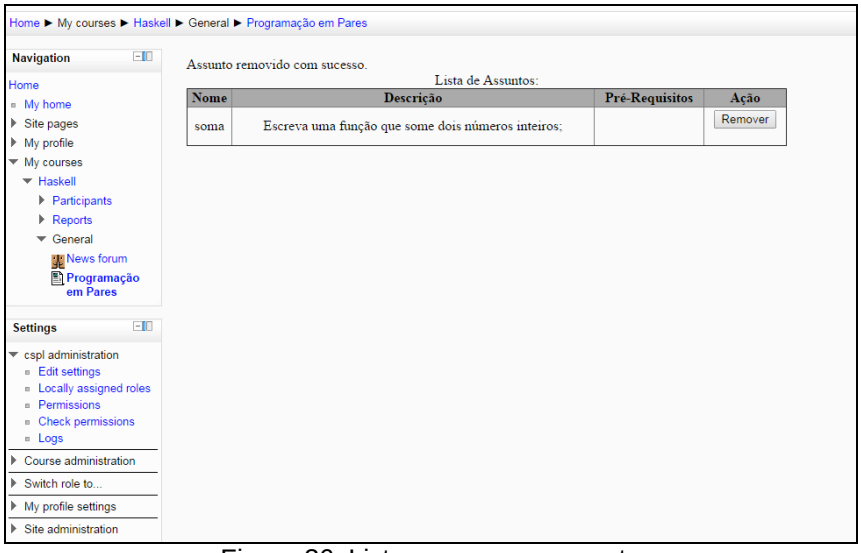


Figura 36. Listar e remover assunto

O próximo recurso a ser abordado é o do acompanhamento dos progressos dos alunos. A Figura 37 ilustra como é feita a escolha de um aluno para ser visualizado o percurso da aprendizagem dele, durante o desenvolvimento das atividades.

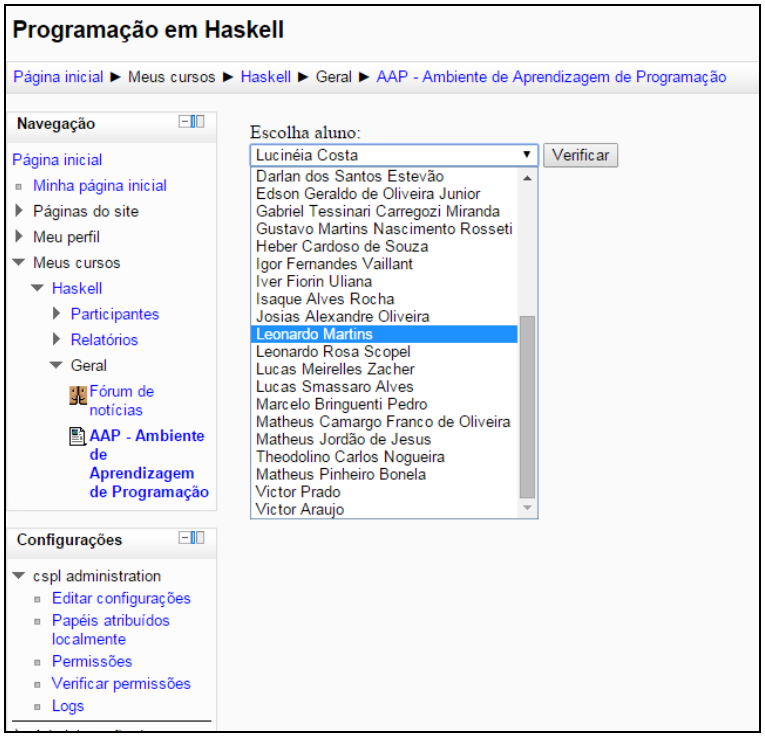


Figura 37. Acompanhar o processo de aprendizagem

Ao acessar em verificar, o professor é redirecionado a página onde constam as informações referentes às atividades desenvolvidas pelo aluno. A Figura 38 mostra o local em que o professor faz a configuração das recomendações.

É neste local que o professor indica quantos exercícios devem ser desenvolvidos de forma correta para que o aluno possa seguir para o próximo nível. O professor também pode indicar a quantidade mínima de linhas que o plano de teste deve ter para ser usado na avaliação de um código da solução.

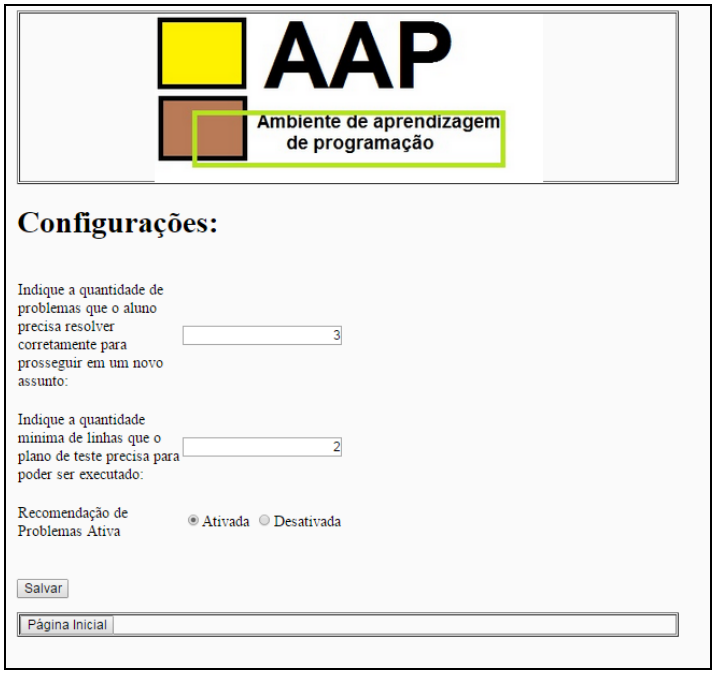


Figura 38. Configurar a recomendação de atividades

### 5.2.9 Suporte ao envio de soluções recomendadas

**Requisito:** estar cadastrado e autenticado no ambiente de aprendizagem como aluno de Programação I.

**Suporte tecnológico:** recebedor de soluções

**Protótipo**

O aluno através da sua visão no ambiente de aprendizagem, visualiza os assuntos pré-cadastrados pelo professor, e faz a escolha de um deles para começar a desenvolver os problemas nele cadastrados. A Figura 39, apresenta o local em que o aluno faz a escolha dos assuntos recomendados.

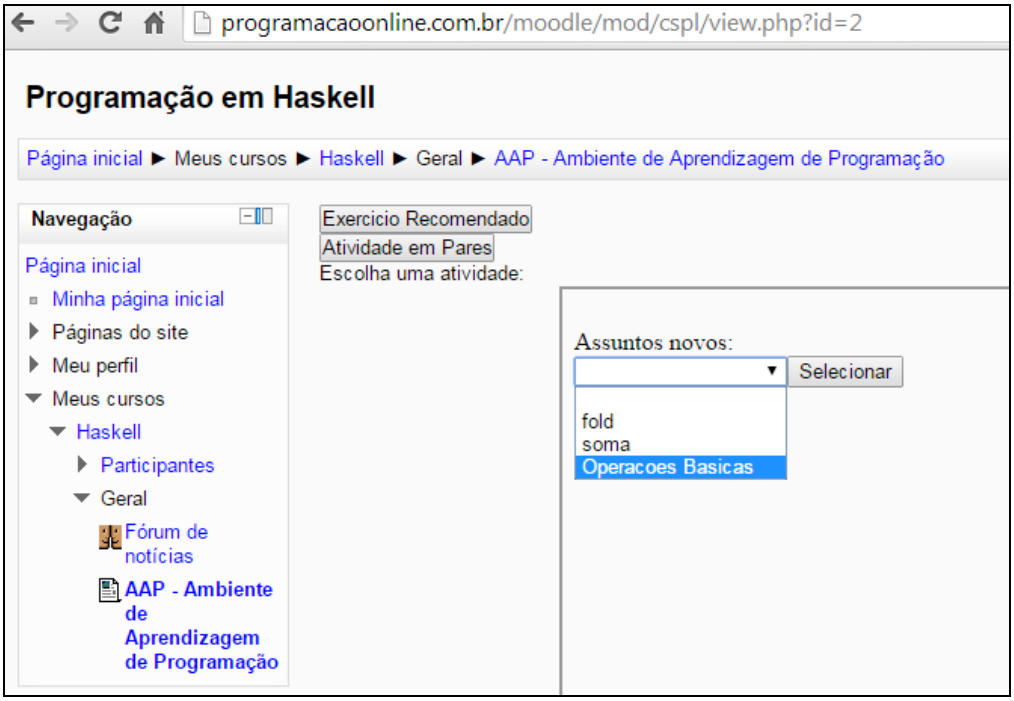


Figura 39. Escolha do assunto

Feita a escolha dentre os assuntos apresentados, o aluno é redirecionado a um janela em que pode desenvolver um dos problemas cadastrados neste assunto, conforme apresenta as Figuras 40 e 41.

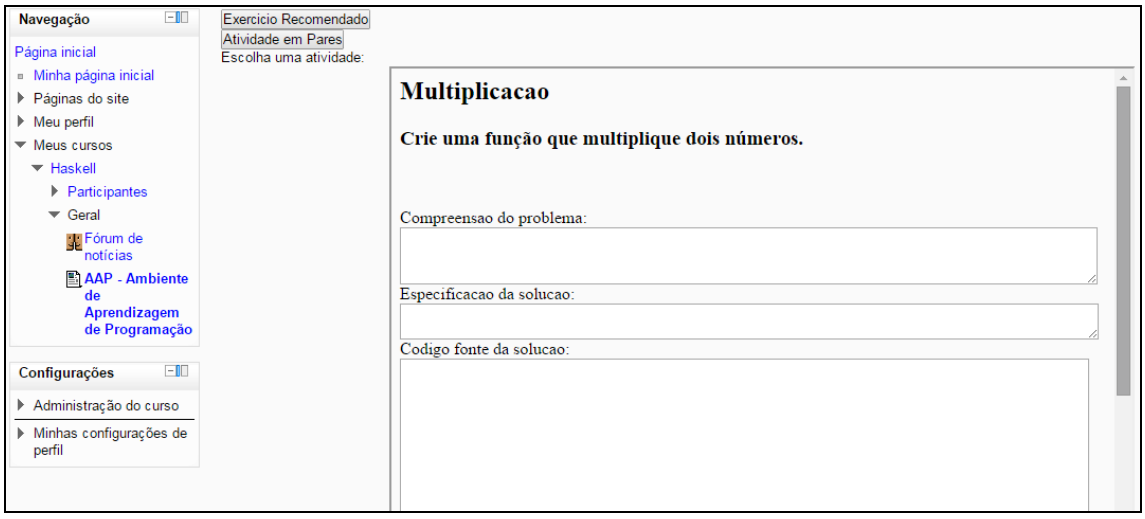


Figura 40. Desenvolvimento da atividade recomendada (a)

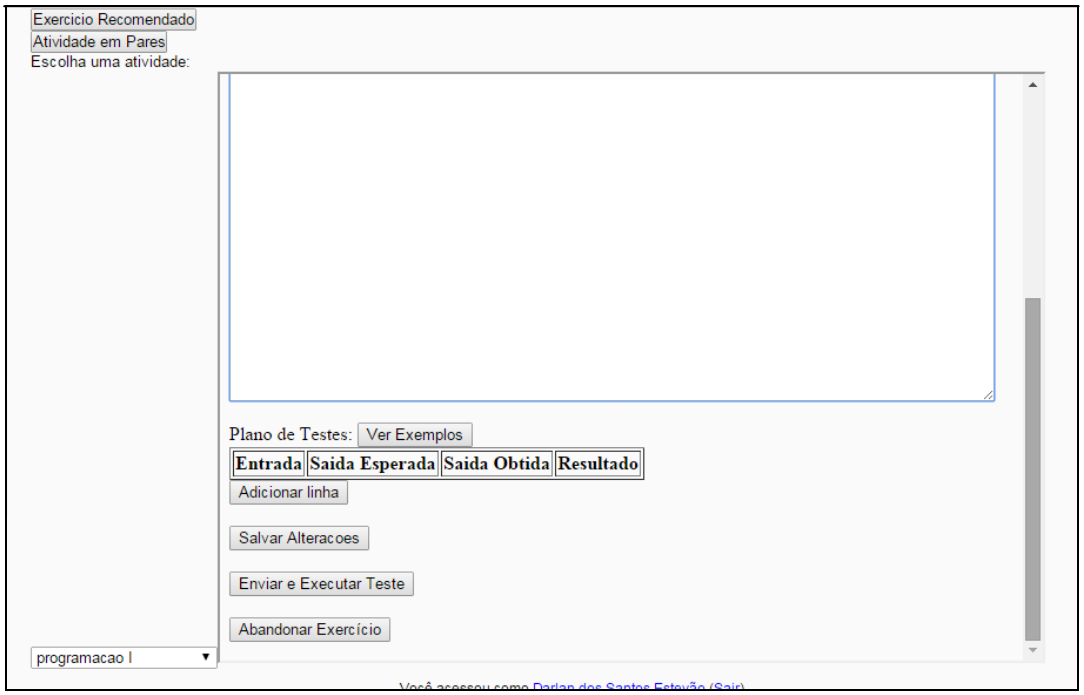


Figura 41. Desenvolvimento da atividade recomendada (b)

O ambiente também apresenta um exemplo de como o aluno insere o código da solução e executa o plano de teste, conforme mostra a Figura 42.

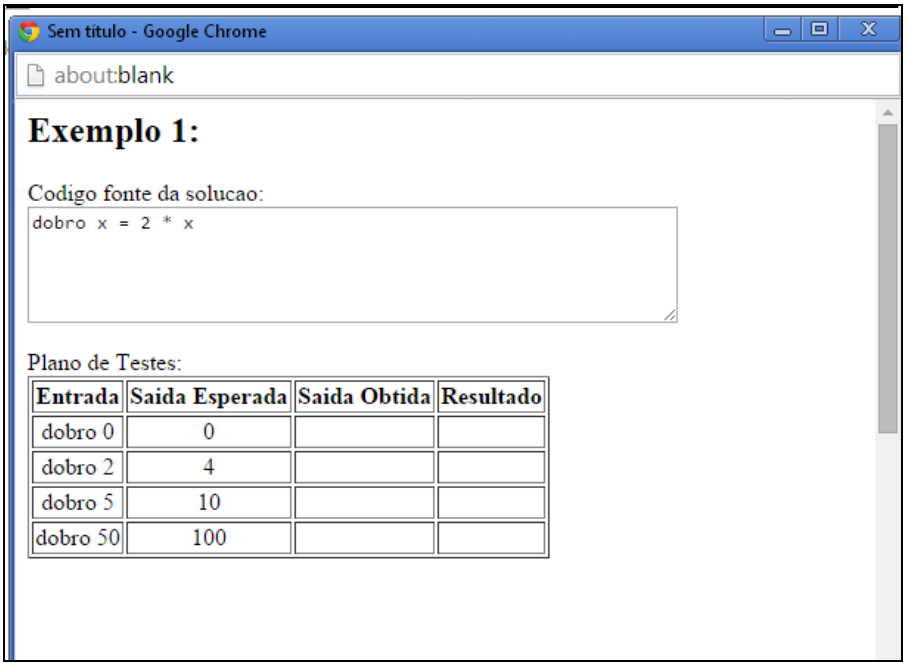


Figura 42. Ver exemplos

Após o desenvolvimento das atividades, o ambiente faz a análise do desempenho do aluno. Caso ele tenha um bom desempenho, poderá passar para outra atividade de um nível maior do que ele está no momento. Caso contrário, o sistema envia outro exercício similar àquele que o discente está desenvolvendo, com uma solução parcialmente concluída.

Se, ainda assim, o aluno não conseguir terminar a solução, o ambiente oferece outra atividade com a solução completa ou quase completa (com poucos erros) para que o aluno possa fazer os testes da solução ou acertar os erros.

O aluno também tem a opção de abandonar o exercícios. Quando o aluno entrar novamente no ambiente, ele receberá outra atividade referente ao mesmo assunto em que parou na sessão anterior.

## 5.3 CONSIDERAÇÕES FINAIS SOBRE O PROTÓTIPO

Este capítulo apresentou a implementação de um ambiente com suporte a arquiteturas pedagógicas para a aprendizagem de programação de computadores. O diferencial deste trabalho esta na possibilidade do professor usar em um mesmo ambiente, várias arquiteturas pedagógicas com abordagens e recursos digitais diferentes, para facilitar a aprendizagem de programação.

O aluno, através do método *MCP*, trabalha as várias etapas da criação de programas, desde a compreensão do problema proposto até o teste da solução desenvolvida, o que proporciona melhorias ao aprendizado.

O ambiente traz ainda possiblidades ao estudante de desenvolver suas tarefas de forma colaborativa (grupo ou pares) ou individualmente, podendo fazer os testes da solução desenvolvida no próprio ambiente, comparar as soluções e receber recomendações de novos exercícios, de acordo com seu desempenho.

Atualmente o protótipo oferece funcionalidades para o professor como: (i) cadastrar problemas, (ii) criar listas de exercícios, (iii) permitir ao docente o acompanhamento das atividades desenvolvidas pelos alunos, (iv) formar pares de alunos para o desenvolvimento de atividades colaborativas, (v) verificar a diferença entre as soluções desenvolvidas, (vi) criar comentários, (vii) desenvolver soluções em pares, (viii) visualizar comentários, (vx) visualizar soluções desenvolvidas em pares, (x) visualizar soluções desenvolvidas individualmente, (xi) cadastrar problemas a serem recomendados, (xii) cadastrar assuntos de acordo com os problemas cadastrados, (xiii) listar exercícios a serem recomendados, (xix) remover exercícios a serem recomendados, (xx) listar assuntos a serem recomendados, (xxi) remover assuntos a serem recomendados.(xxii) fazer a recomendação de atividades com base nos avanços apresentados pelo aluno (xxiii) fazer o acompanhamento dos progressos dos alunos, (xxiv) configurar a quantidade de problemas por assunto a ser desenvolvido, (xxv) enviar mensagens ao professor quando o aluno comenta uma solução, (xxvi) enviar mensagens ao professor, todas as vezes que o aluno abandonar uma atividade recomendada. (xxxvii) enviar arquivos em pdf, com a formação dos pares formados, (xxxviii) gerar pdf com a formação dos pares gerados, (xxxix) enviar mensagens automáticas para os alunos, convidando-os a comentar as soluções dos pares, (xl) enviar mensagens automáticas informando aos alunos que desenvolvam uma solução colaborativa, (xli) visualizar pares formados.

A Figura 43 apresenta a página inicial do professor autenticado no ambiente.

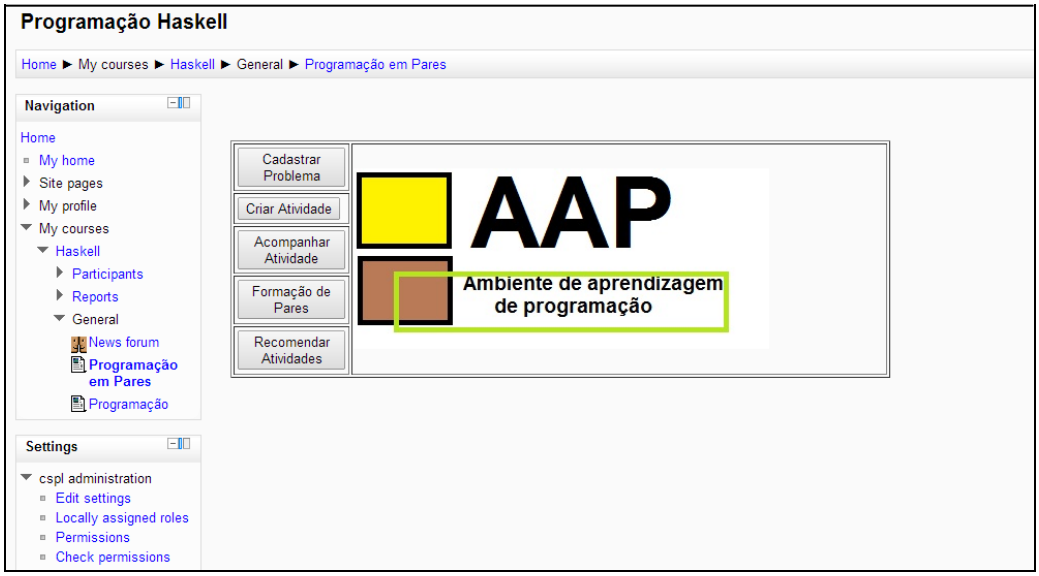


Figura 43. Visão do professor

A visão do aluno, no Ambiente de Aprendizagem de Programação, permite a escolha de atividades a serem desenvolvidas, como também oferece um suporte para que o aluno possa desenvolver atividades em pares, ou receber recomendação de exercícios a serem desenvolvidos.

## 5.4 TECNOLOGIAS USADAS

Esta seção mostra as tecnologias usadas para a implementação das arquiteturas pedagógicas no ambiente de aprendizagem de programação. São elas:

**Xampp 3.2.1** - É um servidor independente de plataforma. *Software* livre que consiste de: base de dados *MySQL*, servidor web Apache e os interpretadores para linguagens de script: *PHP* e *Perl.* O programa está liberado sob a licença GNU e atua como um servidor web livre, fácil de usar e capaz de interpretar páginas dinâmicas.

**Moodle 2.0** - O *Moodle* é uma plataforma de aprendizagem projetada para fornecer a educadores, administradores e alunos, um sistema robusto, seguro e integrado para criar ambientes de aprendizagem personalizados.

**R 3.1.0 -** *R* é uma linguagem e um ambiente de desenvolvimento integrado, para cálculos estatísticos e gráficos. Ele disponibiliza uma ampla variedade de técnicas estatísticas e gráficas, no que inclui a modelação linear e não linear, testes estatísticos clássicos, análise de séries temporais, classificação, agrupamento e outras.

O código fonte do *R* está disponível sob a licença *GNU, GPL* e as versões binárias pré-compiladas são fornecidas para *Windows, Macintosh*, e muitos sistemas operacionais *Unix/Linux.*

***Dreamweaver* 8** - O Adobe *Dreamweaver* é um *software* de desenvolvimento voltado para a *web*. Uma boa funcionalidade do *Dreamweaver* é permitir selecionar a maioria dos navegadores para se ter uma previsão da visualização do *HTML* diretamente no(s) navegador(es) de destino.

***Sed* 4.2.1** – *Sed* significa *Stream EDitor* ou editor de fluxo, é utilizado para realizar transformações de texto básicos em um fluxo de entrada. Ao contrário dos editores convencionais, o *Sed* atua em linha de comando ou *shell script*. No geral, o *Sed* recebe como entrada os dados sobre os quais irá atuar. Aceita expressões regulares, o que lhe confere maior poder e o torna uma excelente ferramenta para administradores de sistemas.

# 6 CONSIDERAÇÕES FINAIS

A conclusão deste trabalho se resume na prestação de contas sobre os resultados alcançados por esta pesquisa.

Em pesquisa bibliográfica conduzida neste trabalho e relatada no Capítulo 3, foi observado que o uso de arquiteturas pedagógicas tem apresentado bons resultados em diversas áreas da educação.

Foi obervado também que as instituições de ensino enfrentam um significativo processo de transformação na educação, por meio da introdução de Tecnologias da Informação e Comunicação (*TICs*).

No protótipo computacional é mostrado que as arquiteturas pedagógicas usadas tem o potencial de ajudar na aprendizagem de programação. As arquiteturas pedagógicas usadas neste protótipo foram:

1. Arquitetura Pedagógica *CSPL,* onde o estudante desenvolve suas atividades através do método *MCP*, que consiste basicamente em sete etapas: compreensão do problema, planejamento de teste, especificação da solução, codificação do programa, teste de solução e avaliação da solução.

Com o uso desta arquitetura pedagógica, o professor faz o acompanhamento do aprendizado do aluno, para identificação dos pontos fortes e correção dos pontos fracos.

2. Programação em pares – nessa arquitetura pedagógica os alunos podem criar várias soluções para o mesmo problema e exercitar o espírito crítico por meio da comparação de soluções diferentes para o mesmo problema.

3. Recomendação de exercícios – esta *AP* leva em consideração os avanços obtidos pelos estudantes, em atividades desenvolvidas anteriormente, para recomendar novos problemas a serem resolvidos e propiciarem a aprendizagem de novos conceitos e estruturas de programação.

O objetivo principal desta pesquisa se resume em apresentar um ambiente de aprendizagem de programação, com três arquiteturas pedagógicas para aprendizagem de programação de computadores.

Quanto à definição dos recursos digitais, consideramos bem sucedida, na medida em que foi apresentado todo o processo de desenvolvimento de todos os recursos criados, de modo a permitir a continuidade deste trabalho por outros pesquisadores.

Na questão da viabilidade das arquiteturas pedagógicas, a implementação de um ambiente computacional que dê suporte a essas arquiteturas foi determinante. Através dele foi possível avaliar os pontos fortes e os fracos, bem como oportunidades de melhoria, conforme descritos nas considerações finais do Capítulo 5.

Em geral, foram encontradas respostas para as principais questões norteadoras desta pesquisa, mostrando arquiteturas pedagógicas úteis para a aprendizagem de programação de computadores.

## 6.1 Trabalhos Futuros

As seguintes alterações poderiam melhorar de imediato alguns dos resultados aqui apresentados:

- ✓ inserção de novas arquiteturas pedagógicas no Ambiente de Aprendizagem de Programação;
- ✓ uso de um protocolo de aceitação de soluções, na *AP* Programação em Pares, de modo a permitir ao aluno a comparação das soluções (individuais e colaborativa) de forma automática, sem que haja a necessidade do aluno inserir as três soluções para que essa comparação ocorra;

- ✓ aperfeiçoamento da interface do ambiente, de modo a melhorar a usabilidade dela;
- ✓ inclusão no *AAP* de um agente pedagógico que auxilie o estudante no desenvolvimento de suas atividades.

# 7 REFERÊNCIAS

A. Ater-Kranov, et al., ***Developing a community definition and teaching modules for computational thinking: accomplishments and Paper presented at the Proceedings*** of the 2010 ACM conference on Information technology education, pp. 143-148, 2010.

ADOMAVICIUS, G. e Tuzhilin, A. (2005), ***Toward the Next Generation of Recommender Systems: A Survey of State-of-the-Art and Possible Extensions***, IEEE Transactions on Knowledge and Data Engineering, v.17 n.6 p. 734-749

ASSIS, João Francisco de. **Evasão escolar no ensino profissionalizante: Um estudo de Caso do Colégio Francisco Carneiro Martins.** UNICENTRO: Revista Eletrônica de pós-graduação Lato Sensu, 2008.

ASTRACHAN, O., & Reed, D. (1995). AAA e CS 1: **A Abordagem de Aprendizagem Aplicada ao CS 1**. Em C. Laxer, CM Branco, JE Miller & J. Gersting (Eds.), *Actas do vigésimo sexto SIGCSE técnico simpósio sobre educação informática.* New York: ACM Press, 1-5.

AUSUBEL, D.P. (1963). **The psychology of meaningful verbal learning.** New York, Grune and Stratton.

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. ***Modern Information Retrieval***. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

BALABANOVIC, M; Shoham, Y. ***Communications of the ACM. Fab: Content-Based, collaborative recommendation***, New York, v.40, n.3, p.66-72, mar.1997.

BEALE, R. & Jackson, T. ***Neural computing: an introduction***. Institute of Physics Publishing, 1994.

BEHAR, P. A.; PASSERINO , L.; BERNARDI, M. **Modelos pedagógicos para educação a distância: pressupostos teóricos para a construção de objetos de aprendizagem**. RENOTE: Revista Novas Tecnologias na Educação, Porto Alegre, v. 5, p. 25-38, 2007.

BONFIN, M. E. **Recuperação de documentos texto usando modelos probabilísticos estendidos**. In: VI - EPCC Encontro Internacional de Produção Científica Cesumar 27 a 30 de outubro de 2009.

BORGES, M. A. F. (2000). **Avaliação de uma Metodologia Alternativa para a Aprendizagem de Programação**. VIII Workshop de Educação em Computação – WEI 2000. Curitiba, PR, Brasil.

BORGES, A. L. S.; Ladeira, M.B.; Oliveira, M. P. V. **A aplicação do modelo de redes Bayesianas para o gerenciamento de risco de ruptura em cadeias de suprimento**. In: Anais do SIMPOI 2008.

BREESE, J. S., HECKERMAN, D., KADIE, C., 1998, ***Empirical Analysis of Predictive Algorithms for Collaborative Filtering***. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43-52, Madison, WI, USA.

BRITO, L. M.; Junior, J. R. G. **Uso de ambientes virtuais de aprendizagem na educação a distância e presencial: o caso do Pvanet**. In: Coletânea de artigos sobre informática na educação – construções em curso – volume 2, ano, 2013.

BURKE, Robin. Hybrid recommender systems: ***Survey and experiments. User Modeling and User Adapted Interaction***, v.12, n. 6, p. 331–370, nov. 2002. Disponível em: <http://josquin.cti.depaul.edu/~rburke/pubs/burke-umuai02.pdf>. Acesso em: 08 de outubro de 2013.

BURKE, Robin. 2007. ***Hybrid Web Recommender Systems.*** Pp. 377–408 in *The Adaptive Web*, vol. 4321, edited by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg.

CABRAL, M. I. C. et al. **Perfil dos cursos de computação e informática no Brasil,** XXVII Congresso da SBC - XV WEI, Rio de Janeiro, 2007.

CABRÉ, G. J. **Filtragem Colaborativa aplicada à recomendação Musical**. Dissertação de Mestrado. Fundação Edson Queiroz – Universidade de Fortaleza – MIA Mestrado em Informática Aplicada. Fortaleza – 2011.

CÁMARA, L. M. S.; Velasco, M. P.; Alcover, C. M.; Iturbide, J. A. V.: ***An evaluation of students' motivation in computer-supported collaborative learning of programming concepts***. In: Computers in Human Behavior, 2013.

CAMPOS, R. L. B. L. ERMC2: **Uma proposta de metodologia para melhoria do ensino-aprendizado de lógica de programação**. XI Congresso Chileno de Educacíon Superior em Computacion (CCESC), (2009). Santiago, Chile, Jornadas Chilenas de Computacion (2009), Santiago, Chile, (2009b), Vol. Único.

CARVALHO, M. J. S.; Nevado, R. A.; Menezes, C. S. : **Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático**. In: XVI Simpósio Brasileiro de Informática na Educação- SBIE – UFJF – 2005.

CARVALHO, Marie J. S., NEVADO, Rosane e MENEZES, Crediné S. **Aprendizagem em Rede na Educação a Distância – estudos e recursos para formação de professores.** Lens – Porto Alegre/RS, 2007.

CASEY, G. Cegielski and Dianne J. Hal(2006): ***What Makes a good programmer?:*** In Communications of the ACM October 2006/Vol. 49, No. 10.

CAZELLA, S. C.: Reategu, E. ***Mini-course: Recommender Systems***. In: ENIA, 2005, São Leopoldo, RG, 2005.

CAZELLA, Sílvio César; NUNES, Maria Augusta S. N.; REATEGUI, Eliseo Berni. ***A Ciência da Opinião: Estado da arte em Sistemas de Recomendação,*** 200-?. Disponível em:

< http://200.17.141.213/~gutanunes/hp/TCI/JAI4.pdf >. Acesso em: 10 de outubro de 2013.

CHAGAS, L.B.C; Oliveira, M.G.: Metodologia ANEA para avaliação Online de Lógica de Programação. In: Anais do XXII SBIE - XVII WIE. Aracaju, 2011.

CHAGAS, R. P. ***Aplicação de redes Bayesianas na previsão de crescimento de fluxos de caixa***. Dissertação de mestrado, Fundação Getúlio Vargas (FGV/EESP) – São Paulo, 2007.

COHEN, A., & Haberman, B. (2007). ***Computer science: a language of technology***. ACM SIGCSE Bulletin, 39(4), 65–69.

CHEN, Gwo - Dong ; Li, Liang - Yi ; Wang, Chin – Yea. ***A Community of Practice Approach to Learning Programming Tecnology***. In Turkish Online Journal of Educational Technology - TOJET, 2012, Vol.11(2), p.15-26.

CORREIA, l. **A. Um Sistema de Recomendação de Promoções Baseado em Posts no Twitter** . In: Trabalho de conclusão de curso. UFP - 2011.

COVER, T. ; Hart, P. ***Nearest Neighbor Pattern Classification*** . In: IEEE Transactions on Information Theory. Janeiro, 1967.

DAVIS, J. C. ***Statistics and data analysis in geology***. New York: Wiley, 1986. 646p

DESHPANDE, Mukund; KARYPIS, George. ***Item-Based Top-N Recommendation Algorithms.*** ACM Transactions on Information Systems**.** New York, v. 22, n. 1, p. 143- 177, jan. 2004.

DIJKSTRA, Edsger W. (1989). ***On the Cruelty of Really Teaching Computing Science***. In Communications of ACM, Issue 12, (vol.32), 1398-1404.

DUNICAN, E. (2002). ***Making The Analogy: Alternative Delivery Techniques for First Year Programming Courses***. In 14th Workshop of the Psychology of Programming Interest Group, Brunel University, p.89- 99, In J. Kuljis, L. Baldwin & R. Scoble (Eds).

ECKHARDT, A. ***Similarity of users' (content-based) preference models for Collaborative filtering in few ratings scenario.*** In: Expert Systems with Applications, 2012.

FECHINE, Joseana Macêdo. ***Inteligência Artificial I - Representação do Conhecimento (Parte IV)***, disponível em <http://dsc.ufcg.edu.br/~joseana/IA_NA15.zip>, acesso em 30 de outubro de 2013.

FERRO, M.R.C., Nascimento Júnior, H. M., Paraguaçu, F., Costa, E. B., Monteiro, L. A. L. (2011) **Um Modelo de Sistema de Recomendação de Materiais Didáticos para Ambientes Virtuais de Aprendizagem**. Simpósio Brasileiro de Informática na Educação, p. 810-819.

FILHO, D.B.F.; Junior, J. A. F.: **Desvendando os mistérios do coeficiente de correlação de Pearson( r ) *.** In: Revista Política de Hoje, Vol 18, Nº 1, 2009.

FILMES NETFLIX, 2013, **Filmes Netflix** homepage http://filmes-netflix.blogspot.com.br/2013/07/guia-rapido-como-netflix-faz.html, acessado em 01/12/2013.

GARCÍA, E. G.; Veiga, E. C. da. **O construtivismo e as funções mentais.** Diálogo Educacional, Curitiba, v. 7, n. 20, jan./abr. de 2007.

GARCÍA, E., Romero, C., Ventura, S., and Castro, C. D. (2009). ***An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering.*** User Modeling and User-Adapted Interaction, 19(1-2):99–132.

GATES, G.W. ***The reduced nearest neighbor rule***. IEEE Transactions on Information Theory 18, 1972, pp 431-433.

GEYER - SCHULZ, A., Hahsler, M., Jahn, M. (2001) ***Educational and Scientific Recommender Systems: Designing the Information Channels of the Virtual University.*** International Journal of Engineering Education.

GOMES, A., Carmo, L., Bigotte, E. and Mendes, A. J. (2006). ***Mathematics and programming problem solving***, in Proceedings of 3rd E-Learning Conference – Computer Science Education, Coimbra,Portugal (CD-ROM).

GOMES, A.; Henriques, J.;Mendes, A. J.:**Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores.** In: Educação, Formação & Tecnologias, vol. 1 (1), Maio 2008.

GOMES, A. e Mendes, A. (2000). **Suporte à Aprendizagem da Programação com o Ambiente SlCAS**. Actas do V Congresso Ibero-americano de Informática Educativa,Viña del Mar, Chile.

GONZALEZ, G., de la Rosa, J.L., and Montaner, M. (2007). ***Embedding Emotional Context inRecommender Systems.*** In The 20th International Florida Artificial Intelligence Research Society Conference-FLAIRS, Key West, Florida.

GOVENDER, I: ***The learning context: Influence on learning to program.*** In Computers & education 53 (2009) 1218-1230.

GROUPLENS, 2013. ***GroupLens homepage***, http://www.grouplens.org, acessado em 25/09/2013.

HART, P.E. ***The condensed nearest neighbor rule***. IEEE Transactions on Information Theory 14, 1968, pp 515-516.

HERLOCKER, J., KONSTAN, J., BORCHERS, J., et al., 1999, ***An algorithmic framework for performing collaborative filtering***. In: Proceedings of the

22nd annual international ACM SIGIR conference on Research and development in information retrieval, p.230-237, August, Berkeley, California, USA.

IBM:http://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao / acessado em 26/09/2013.

J. A., & Sherrell, L. B. (2010). ***Why computational thinking should be integrated into the curriculum***. J. Comput. Small Coll., 25(5), 66-71.

JACOBSEN. G. R. C. ***Redes Bayesianas***. In: Centro de Educação Superior do Alto Vale do Itajaí – CEAVI. Universidade do Estado de Santa Catarina – UDESC. Produção de artigos nas disciplinas 2011/2. http://www.ceavi.udesc.br/?id=387 acessado em 31 de outubro de 2013.

JENSEN, Finn V.; ***Bayesian Networks and Decision Graphs***; 2001; Statistics for engineering and information science; Springer

JENKINS, T. (2002). ***On the difficulty of learning to program***. In Proceedings of 3rd Annual LTSN_ICS Conference (Loughborough University, United Kingdom, August 27-29, 2002). The Higher Education Academy, p.53-58.

KALLES, D. (2008). ***Estudantes que trabalham para estudantes em cursos de programação***. *Computadores e Educação, 50* (1), 91 -97.

KAINULAINEN, J. ***Clustering Algorithms: Basics and Visualization***. In: Helsinki University of Tecnology. Laboratory of Computer and Information Science T-61.195 Special Assignment 1, 2002.

KAZIMOGLU, C.; Kiernan, M.; MacKinnon, L. B. L.: ***Learning Programming at the Computational Thinking Level via Digital Game-Play***. In: International Conference on Computational Science, ICCS 2012.

KERCKHOVE, D. (2003) **A arquitetura da inteligência: interfaces do corpo, da mente e do mundo**. In Arte e vida no século XXI - tecnologia, ciência e criatividade, Edited by D. Domingues. São Paulo: Editora UNESP, p.15-26

KHVILON, E., & Patru, M. (2002). ***Information and communication technology in education: A curriculum for schools and programme of teacher development***. http://unesdoc.unesco.org/images/0012/001295/129538e.pdf. Acessado em 17 de setembro de 2013.

KITCHENHAM, B. A. (2007) ***Guidelines for performing Systematic Literature Reviews*** in Software Engineering.

KOLLING, M., & Barnes, DJ (2004). **Reforço de Aprendizagem Aprendiz-Based de Java.** *ACM SIGCSE Boletim 36* (1), 286-290.

KONSTAN, J., Miller, V, Maltz, D., et al, 1997, ***GroupLens: applying collaborative filtering to Usenet news***, Communications of the ACM, v. 40 , n. 3, pp. 77–87.

KORDAKI, M.: ***A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation***. In: Computers & education 54 (2010) 69-87.

KOREN, B. Y. ***Collaborative Filtering withTemporal Dynamics.*** In: communications of the ac m | april 2010 | vol. 53 | no. 4.

LAU, W. W.F. , Yuen, A. H.K.. ***Modelling programming performance: Beyond the influence of learner characteristics***. In: Computers & Education, 2011.

LAWRENCE, R. D. et al. (2001) ***Personalization of supermarket product recommendations***. Journal of Data Mining and Knowledge Discovery, v.5, n.1/2, January, p.11-32.

LEINO, J. and R¨aih¨a, K.-J. (2007). ***Case amazon: ratings and reviews as part of recommendations. In RecSys '07***: Proceedings of the 2007 ACM conference on Recommender systems, pages 137–140, New York, NY, USA. ACM.

LEVY, Pierre. **A máquina universo: criação, cognição e cultura informática.** Porto Alegre: ArtMed, 1998.

LINDEN, G. Smith, B. York, J. Amazon.com ***Recommendations Item-to-Item Collaborative Filtering.*** IEEE INTERNET COMPUTING Janeiro/Fevereiro 2003.

LUNA, J. E. O. **Algoritmos EM para Aprendizagem de Redes Bayesianas a partir de Dados Incompletos.** Dissertação de Mestrado, UFMGS - Universidade Federal de Mato Grosso do Sul, 2004.

LINDEN, R. **Técnica de Agrupamentos.** Revista de Sistema de Informação da FSMA, n.4 (2009), PP.18-36.

MASETTO, M. T. **Mediação pedagógica e o uso da tecnologia. In: Novas tecnologias e mediação pedagógica**. Campinas:Papirus, 2000.

MILLER, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. (2003). ***Movielens unplugged: experiences with an occasionally connected recommender system***. In IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces, pages 263–266, New York, NY, USA. ACM.

MONTANER, M et al. (2003). ***A Taxonomy of Recommender Agents on the Internet. Artificial Intelligence Review***. Netherlands : Kluwer Academic Publishers, pp. 285- 330, Aug.

MORAIS, S.F.S (2012) . In: **Sistemas de Recomendação em Rapid Miner: um caso de estudo**. Dissertação de Mestrado – Universidade do Porto, 2012.

MOVIELENS, 2013, ***MovieLens homepage***, http://movielens.umn.edu/, acessado em 23/09/2013.

NEVADO, A.R.; Dalpiaz, M. M.; Menezes, C. S. **Arquitetura Pedagógica Colaborativa de Conceituações.** In: Anais WEI, 2009. Disponível em:

http://www.br-ie.org/pub/index.php/wie/article/view/2150/1916. Acessado em 10/04/2014.

NOBRE, I. A. M. N., Menezes, C. S. (2002) **Suporte à Cooperação em um Ambiente de Aprendizagem para Programação (SAmbA)**. XIII Simpósio Brasileiro de Informática na Educação – SBIE 2002. São Leopoldo, RS, Brasil.

NOBRE, I. A. M. Nunes, V. B., Gava. T. B. S.; Fávaro, R. P.; Bazet, L. M. B. (2013) **Informática na Educação - um caminho de possibilidades e desafios**. Serra, 2013.

NUNES, M. A. S. N. (2009). R***ecommender Systems based on Personality Traits:Could human psychological aspects influence the computer decision-making process?***. 1. ed. Berlin: VDM Verlag Dr. Müller. v.1. 140 p.

NUNES, M. A. S. N. ; Cerri, Stefano A. ; Blanc, N. (2008a). ***Towards User Psychological Profile.*** In:Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais, 2008, Porto Alegre. VIII Simpósiosobre Fatores Humanos em Sistemas Computacionais IHC 2008. Porto Alegre : Sociedade Brasiliera da Computação. v. 1. p. 196-203.

OLIVEIRA, L. G. ***Proposal of the methodological structure to implement recommender system***. http://www.intelog.net/ArtigosNoticias/Arquivos/artigo_leonardo_oliveira.pdf. Acesso em: 08 de outubro maio, 2013.

OLIVEIRA, l. G. **Sistema de recomendação de meios de hospedagem baseado em filtragem colaborativa e informações contextuais.** Dissertação de Mestrado. Universidade Federal de Santa Catarina - 2007.

OLIVEIRA, M. G.; CICARELLI, P. M.; OLIVEIRA, E. ***Recommendation of Programming activities by multi-label classification for a formative assessment of students***. In: Expert Systems with Applications, 2013.

OLIVEIRA, M. G. **Núcleos de avaliações diagnóstica e formativa para regulação da aprendizagem de programação.** Tese de Doutorado. Ufes-2013**.**

PARK D.H., Kim H.K., Choi I.Y., Kim J.K. (2012) ***A literature review and classification of recommender systems research***. Expert Systems with Applications, 39 (11), p. 10059-10072.

PAYSAN, T. G. **Sistema WEB Gerenciador de Perfis.** In: Monografia de conclusão de curso. Ufes-2006.

PENNOCK, David. M. et al. ***Collaborative Filtering by Personality Diagnosis: A Hybrid Memory And Model-Based Approach.*** In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1999, Estocolmo. Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. São Francisco: Morgan Kayfmann, 2000, p. 473-480. Disponível em: <http://dpennock.com/papers/pd-uai-00.pdf>. Acesso em: 09 de outubro. 2013.

PERKINS, D. N., SCHWARTZ, S. and SIMMONS, R.; (1988). ***Instructional Strategies for the Problems of Novice Programmers***. In R. E. Mayer (ed.), Teaching and Learning Computer Programming, p.153-178. Hillsdale, NJ: Lawrence Erlbaum Associates.

PERRENOUD, P. **Avaliação: Da Excelência à Regulação das Aprendizagens – Entre Duas Lógicas.** Porto Alegre, RS: Artmed Editora, 1999.

PERRENOUD, P.; Magne, B. C. **Construir: as competências desde a escola**. Porto Alegre: Artmed, 1999.

PIAGET, J.; **Evolução intelectual da adolescência à vida adulta** (Tradução de Tania Beatriz Iwaszko Marques e Fernando Becker do artigo publicado em inglês sob o título: ***Intellectual Evolution from Adolescence to Adulthood***, pela Human Development, 15:1-12, 1972).

PIAGET, J.; **Fazer e compreender**; Editora da Universidade de São Paulo; 1978.

PIMENTEL, E.P.;França, V. F.; Noronha, R. V.; Omar, N.: **Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores**. In IX Workshop de Informática na Escola – WIE-2003.

POZO, J. I. **Teorias cognitivas da aprendizagem**. 3 ed. Porto Alegre: Artes Médicas, 2002.

PRADA, R., MA, S., Nunes, M. A. S. N. (2009) .Personality ***in Social Group Dynamics*** In: International Conference on Computational Science and Engineering- CSE '09, 2009, Vancouver. International Conference on Computational Science and Engineering- CSE '09. v.4. p.607 – 612.

QUALLS, J. A., & Sherrell, L. B. (2010). ***Why computational thinking should be integrated into the curriculum.*** *J. Comput. Small Coll.*, 25(5), 66-71.

QUEIROZ, S. R. M. 2003. ***Group Recommendation Strategies Based On Collaborative Filtering.*** Dissertação de Mestrado, Universidade Federal de Pernambuco.

RAJARAMAN, A.; Ullman, J. D. (2011). "***Data Mining". Mining of Massive Datasets***. pp. 1–17.

REAL, L. M. C. ; Corbellini, Silvana. **Proposta de uso de WIKI como Arquitetura Pedagógica: cooperação.** In: 22º Simpósio Brasileiro de Informática na Educação - 17º Workshop de Informática na Escola, 2011, Aracaju. Anais do XXII SBIE - XVII WIE, 2011. v. Wapsed. p. 1-9.

REATEGUI, E. B.; Cazella, S. C.: ***Sistema de Recomendação***. In XXV Congresso da Sociedade Brasileira de Computação. A Universidade da Computação : Um agente de inovação e conhecimento.São Leopoldo, 2005.

REIS, L. F. M. **Sistema de Recomendação Baseado em Conhecimento**. Dissertação de Mestrado. Universidade de Coimbra, 2012.

REIS, E.; Godói, K. A. A dialogicidade na tutoria a distância: uma contribuição freireana. Anais do VIII Seminário Internacional de Paulo Freire. Recife, 2010.

RESNICK, P. e Varian, H. R. (1997). ***Recommender Systems. Communications*** of the ACM, New York, v.40, n.3, pp. 55-58, Mar.

RESNICK, P., Iacovou, N., Sushak, M., et al, 1994, ***GroupLens: An open architecture for collaborative filtering of netnews***. In: Proceedings of the 1994. ACM conference on Computer supported cooperative work, pp. 175 – 186, Chapel Hill, North Carolina, USA.

RIBEIRO, R.S; Brandão L.O; Brandão; A. A. F.: **Uma visão do cenário nacional do ensino de algoritmos e programação: uma proposta baseada no paradigma de programação visual**. I Congresso de brasileiro de Informática na educação, 2012, Rio de Janeiro.

ROCHA, C. C.: RecDoc: **Um sistema de recomendação para uma biblioteca digital na web.** Tese de mestrado UFRJ, 2003.

ROCHA, T. M. Hybrid Rec – **Protótipo de sistema de recomendação utilizando filtragem híbrida**. Trabalho de conclusão de curso, Gravatí: 2007.

ROMERO, C.; VENTURA,S.;PECHENIZKIY, M.; BAKER, R.S.J.d. ***Handbook of Educational Data Mining,*** 2011.

SANDVIG, J. J. ***A Survey of Collaborative Recommendation and the Robustness of Model-Based Algorithms***. IEEE Data Eng. Bull. 31 (2): 3-13 (2008).

SALGADO, M. U. C; AMARAL, A. L. (Orgs.). **Tecnologias da Educação: ensinando e aprendendo com as TIC. (guia do cursista)** Brasília: MEC, Sec. da educação à Distância, 2008.

SAMPAIO, R. F.; Mancini, M. C.: ***Estudos de revisão sistemática: Um guia para síntese criteriosa da evidência científica***.in: Revista Brasileira de Fisioterapia. São Carlos, v.11, n.1, p. 83-89, jan/fev.2007.

SARWAR, B. et al. ***Item-based collaborative filtering recommendation algorithms***. Proceedings of the 10th international conference on World Wide Web, 2001. ACM. p.285-295.

SCHAFER, J. Ben; Konstan, Joseph; RIEDL, John. ***Recommender Systems***. In: Conference on Electronic Commerce, 2000, Minneapolis. Proceedings...

SCHAFER, J. Ben et al. (2001) ***E-commerce recommendation applications.*** Journal of Data Mining and Knowledge Discovery, v.5, n.1/2, Janeiro, p.115-153.

SERRANO-CÁMARA, L. M. ; Paredes-Velasco, M. ; Alcover, C. M. ; Velazquez,I., J. Ángel. ***An evaluation of students' motivation in computer-supported collaborative learning of programming concepts*** In: Computers in Human Behavior, 2013.

SHARDANAND, U.; Maes, p. ***Social information Filtering: Algorithms for automating "Word of Mouth"***. MIT Media-lab. Cambridge, MA, 1995. Disponível em: http://citeseer.ist.psu.edu/shardanand95social.html. Acesso em 26 de setembro de 2013.

SIBALDO, M. A. A., Sales, T. B. M., Calado, I.A.A.R., Bittencourt, I.I., Costa, E.B. (2007) ***Mobile GraW: Uma Aplicação para Dispositivos Móveis Baseada em Comunidades Virtuais de Aprendizagem Com Suporte A Recomendação.*** Simpósio Brasileiro de Informática na Educação, p. 214-217.

SILVA, E. L., Menezes, E. M.: **Metodologia da pesquisa e elaboração de dissertação**. 3. ed. rev. atual. Florianópolis: Laboratório de Ensino a Distância da UFSC, 121 p. 2001. Disponível em: <http://www.tecnologiadeprojetos.com.br/banco_objetos/%7B7AF9C03E-

C286-470C-9C07-EA067CECB16D%7D_Metodologia%20da%20Pesquisa%20e%20da%20Disserta%C3%A7%C3%A3o%20%20UFSC%202005.pdf>. Acesso em 18 de agosto de 2013.

SOUZA, D. M.; Maldonado, J.C.; Barbosa, E. F: **Aspectos de Desenvolvimento e Evolução de um Ambiente de Apoio ao Ensino de Programação e Teste de Software**. Anais do 23º Simpósio Brasileiro de Informática na Educação (SBIE 2012) - Rio de Janeiro – 2012.

SOUZA, V.S; Programação Funcional usando Haskell. UFPI – Universidade Federal do Piauí. Setembro, 2009. Disponível em: http://www.ufpi.br/subsiteFiles/gaa/arquivos/files/LF_Apostila.pdf. Acesso em 15/08/2014.

STAHL, Marimar M. **A formação de professores para o uso das novas tecnologias de comunicação e informação**. Petrópolis: Vozes, 1995.

TAVARES, A. (2000), **Ambiente de Aprendizagem de uma linguagem de programação**. http://civil.fe.up.pt/people/staff/acruz/ff/ensinoDistancia.pdf (2005-09-16).

TAVARES, O. L.; Menezes, C.S.; Nevado, R.A.: ***Pedagogical architectures to support the process of teaching and learning of computer programming:*** In FIE2012- Frontiers in education conference, 2012.

TAVARES, O. L.; Menezes, C.S.; Aragón, R.; Costa, L. B.: **Uma arquitetura pedagógica auxiliada por tecnologias para ensino e aprendizagem de programação.** In XXXIII Congresso as sociedade brasileira de computação - CSBC 3013.

TERVEEN, L. & Hill, W., 2001, ***Beyond Recommender System: Helping People to Find Each Other***. In: Proceedings of HCI in the New Millennium, Jack Caroll, ed.,Addison-Wesley.

TORRES, R. **Personalização na internet: Como descobrir os habitos de consume de seus clients, fidelizá-los e aumentar o lucro de seu negócio**. São Paulo: Novatec Editora, 2004.

TORRES, Patrícia Lupion. **Laboratório on-line de aprendizagem: uma proposta crítica de aprendizagem colaborativa para a educação**. Tubarão: Ed. Unisul, 2004.

VALENTIM, H: **Um Estudo sobre o Ensino Aprendizagem de Lógica de Programação**, Florianópolis, 2009.

VIGOSTSKY, Lev. S. **A construção do pensamento e da linguagem.** São Paulo, 2001.

WANG, S.-L., & Lin, S. S. J. (2007). ***The effects of group composition of self-efficacy and collective efficacy on computer-supported collaborative learning***. Computers in Human Behavior, 23, 2256–2268.

WILTON, O. B; Pedro, A. Morettin. Estatística Básica. Editora Saraiva, 8ª edição - 2014.

WING, J. M. (2006). ***Computational thinking.*** Communications of the ACM, 49(2), 33-35.

ZAINE, O. R. (2002). ***Building a recommender agent for e-learning systems***.

## ARTIGOS INCLUÍDOS NA REVISÃO SISTEMÁTICA

GOVENDER, I. (2009). ***The learning context: Influence on learning to program****.Computers & Education*, *53*(4), 1218-1230.

HWANG, W. Y., Wang, C. Y., Hwang, G. J., Huang, Y. M., & Huang, S. (2008). ***A web-based programming learning environment to support cognitive development. Interacting with Computers***, *20*(6), 524-534.

HWANG, Wu-Yuin, et al. ***"A pilot study of cooperative programming learning behavior and its relationship with students' learning performance."*** *Computers & Education* 58.4 (2012): 1267-1281.

KAZIMOGLU, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). ***Learning Programming at the Computational Thinking Level via Digital Game-Play.Procedia*** *Computer Science*, *9*, 522-531.

LAW, K. M., Lee, V., & Yu, Y. T. (2010). ***Learning motivation in e-learning facilitated computer programming courses***. *Computers & Education*, *55*(1), 218-228.

MOONS, J.; Backer, C. D.: ***The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism.*** In: Computers & education 60 (2013) 368-384

SERRANO-CÁMARA, L. M. ; Paredes-Velasco, M. ; Alcover, C. M. ; Velazquez,I., J. Ángel. ***An evaluation of students' motivation in computer-supported collaborative learning of programming concepts*** In: Computers in Human Behavior, 2013.