FAI – CENTRO DE ENSINO SUPERIOR EM GESTÃO, TECNOLOGIA E EDUCAÇÃO

CURSO DE SISTEMAS DE INFORMAÇÃO

CELSO DANILO DA MOTA

FRANCISCO DE FARIA CARDOSO

JULIANO COSTA SILVA

MARCELO PEREIRA COSTA

ROBSON DOS SANTOS

DESENVOLVIMENTO DE SISTEMA PARA MELHORAR A MEMORIZAÇÃO A LONGO PRAZO, UTILIZANDO A TÉCNICA DE REPETIÇÃO ESPAÇADA

SANTA RITA DO SAPUCAÍ – MG

2016

FAI – CENTRO DE ENSINO SUPERIOR EM GESTÃO,

TECNOLOGIA E EDUCAÇÃO

CURSO DE SISTEMAS DE INFORMAÇÃO

CELSO DANILO DA MOTA

FRANCISCO DE FARIA CARDOSO

JULIANO COSTA SILVA

MARCELO PEREIRA COSTA

ROBSON DOS SANTOS

DESENVOLVIMENTO DE SISTEMA PARA MELHORAR A MEMORIZAÇÃO A LONGO PRAZO, UTILIZANDO A TÉCNICA DE REPETIÇÃO ESPAÇADA

Projeto final de curso apresentado à FAI – Centro de Ensino Superior em Gestão, Tecnologia e Educação, como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação sob a orientação da profa. Ma.Eunice Gomes de Siqueira.

SANTA RITA DO SAPUCAÍ – MG

2016

FOLHA DE APROVAÇÃO

HISTÓRICO DE REVISÕES

Data	Versão	Autor	Descrição
12/03/2016	001	Equipe	Criação de documento. Descrição de Requisitos.
14/03/2016	002	Juliano	Descrição de Requisitos. Visão Funcional. Visão de Dados.
19/03/2016	003	Juliano	Descrição de Requisitos. Visão Funcional. Visão de Dados. *Storyboards*.
26/03/2016	004	Juliano	Descrição de Requisitos. Visão Funcional. Visão de Dados.
27/03/2016	005	Juliano	Gerência do Projeto.
30/03/2016	006	Celso Danilo	Revisão Bibliográfica.
31/03/2016	007	Juliano	Gerência do Projeto.
31/03/2016	008	Marcelo	Objetivo do Projeto.
01/04/2016	009	Juliano	Gerência do Projeto.
01/04/2016	010	Celso Danilo	Objetivo do Projeto.

01/04/2016	011	Francisco	Introdução.
02/04/2016	012	Francisco	Introdução.
02/04/2016	013	Francisco	Referências.
02/04/2016	014	Juliano	Gerência do Projeto.
02/04/2016	015	Robson	Adequação às normas ABNT.
02/04/2016	016	Celso Danilo	Objetivo do Projeto.
25/05/2016	017	Juliano	Correções de 1ª Entrega
27/05/2016	018	Celso Danilo	Correções da 1ª Entrega
18/06/2016	019	Juliano	Inserção de todos os tópicos a serem entregues; Descrição de requisitos de servidor.
21/06/2016	020	Juliano	Descrição de visão estrutural.
24/06/2016	021	Celso Danilo	Preenchimento da planilha de monitoramento; Descrição de controle de configuração; Descrição de controle de mudanças; Descrição de gestão dos riscos; Descrição de gestão da qualidade.
25/06/2016	022	Juliano	Descrição de projeto de sistemas distribuídos.
25/06/2016	023	Robson	Revisão.

25/06/2016	024	Francisco	Construção de diagramas de sequência.
26/07/2016	025	Juliano e Celso Danilo	Correções de memorial.
22/08/2016	026	Juliano e Celso Danilo	Correções de memorial.
03/09/2016	027	Juliano e Celso Danilo	Correção de sessão “6.7 Projeto de Sistemas Distribuídos”.
04/09/2016	028	Juliano	Correção de capítulos 1 a 4.
05/09/2016	029	Juliano	Correção de capítulo 6.
08/09/2016	030	Celso Danilo e Juliano	Descrição e revisão de seções 6.8.3 e 4.2.3.2.
09/09/2016	031	Marcelo	Descrição de seções 6.5.1, 6.5.2 e 6.8.2.
09/09/2016	032	Celso Danilo	Descrição de seções 6.8.3
10/09/2016	033	Juliano	Descrição de seções 6.2.1.4, 6.4.1 e 6.4.2.
12/11/2016	034	Juliano	Descrição de capítulo 7 e Obras consultadas.
12/11/2016	035	Marcelo	Descrição de capítulo 8, Glossário e Conclusão.
12/11/2016	036	Celso Danilo	Descrição de seção 6.6, Resumo e atualização de apêndices B, M, N e O.
30/11/2016	037	Equipe	Correções da entrega 4.

RESUMO

Este projeto consiste no desenvolvimento de um aplicativo *mobile* para a plataforma Android, nomeado Aprendiz. O aplicativo Aprendiz utiliza a técnica de repetição espaçada que é uma das derivações da aprendizagem espaçada, baseada na curva hipotética de esquecimento. Essa técnica consiste em realizar revisões sistemáticas das informações recém-adquiridas a fim de fixá-las na memória de longo prazo. Segundo Ebbinghaus, psicólogo que estudou os processos adjacentes da memória, se não fizermos revisões dos conteúdos aprendidos, no tempo correto e no período correto, nosso cérebro deixará os conteúdos aprendidas de lado, para poder armazenar outras informações que nos são apresentadas durante o dia a dia. O modelo ideal de estudos seria fazendo revisões no momento em que o nosso cérebro está se preparando para esquecer, dessa forma a informação passa a ser vista como importante e pode ter melhor retenção a longo prazo. Estudar sem se preocupar em revisar o conteúdo pode ser inútil, e uma das melhores maneiras de realmente assimilar e armazenar informação em nossa memória é com revisões planejadas e com o intervalo correto. O aplicativo Aprendiz é um sistema de *Flashcards* interativo e que além da técnica de repetição espaçada, baseia-se no sistema Leitner. Durante a fase de planejamento e desenvolvimento do projeto foram explorados os conhecimentos adquiridos no curso de Sistemas de Informação da FAI – Centro de Ensino Superior em Gestão, Tecnologia e Educação, destacando as disciplinas de Gerência de Projetos e Engenharia de Software.

Palavras-chave: Técnica de repetição espaçada. Aprendizagem espaçada. Curva hipotética de esquecimento. *Flashcards*. Sistema Leitner. Sistemas de Informação.

LISTA DE FIGURAS

LISTA DE QUADROS

ABREVIATURA E SIGLAS

AON - *Activity On Node*

ARM - *Acorn RISC Machine*

ART - Android Runtime

C# - *C Sharp*

E - MAIL - *Electronic mail*

EAP - Estrutura Analítica do Projeto

FAI - Centro de Ensino Superior em Gestão Tecnologia e Educação

GB - *Gigabyte*

GHz - *Gigahertz*

HTTP - *Hypertext Transfer Protocol*

IDE - *Integrated Development Environment*

IIS - *Internet Information Services*

ILF - *Internal Logic Files*

IP - *Internet Protocol*

JVM - *Java Virtual Machine*

MB - *Megabyte*

MER - Modelo Entidade Relacionamento

MHz - *Megahertz*

MVC - *Model View Control*

OSI - *Open Systems Interconnection*

PCU - Pontos por Caso de Uso

PFC - Projeto Final de Curso

RAM - *Random Access Memory*

REST - *Representational State Transfer*

RF - Requisito Funcional

RISC - *Reduced Instruction Set Computer*

RNF - Requisito Não Funcional

SGDB - Sistema de Gerenciamento de Banco de Dados

SI - Sistemas de Informação

SQL - *Structured Query Language*

UML - *Unified Modeling Language*

USB - *Universal Serial Bus*

WBS - *Work Breakdown Structure*

SUMÁRIO

1 INTRODUÇÃO

Atualmente nos preocupamos muito com nossa capacidade de retenção de conhecimento na memória. Nossa sociedade frequentemente valoriza uma boa memória fazendo dela um fator decisivo para avaliar candidatos em provas com diferentes finalidades. Paralelo a isso, temos problemas quanto à eficiência de aprendizagem nas salas de aula de todo o sistema educacional, embora já se tenha conhecimento de diversas técnicas que auxiliam na revisão de conteúdo, poucas ou nenhuma delas são abordadas para auxiliar o lecionamento das disciplinas.

> Se técnicas simples estivessem disponíveis para que professores e alunos pudessem utilizá-las para melhorar a aprendizagem e desempenho dos alunos, você ficaria surpreso se os professores não estiverem apresentando essas técnicas, e se muitos alunos não estiverem as utilizando? E se estudantes foram prejudicados com técnicas de aprendizagem ineficazes ou se não progrediram devido a isso? Eles não deveriam parar de usar estas técnicas e começar a usar aquelas que são mais eficazes? Os psicólogos têm vindo a desenvolver e avaliar a eficácia das técnicas de estudo e de instruções há mais de 100 anos (PRESSLEY et al., 1989).

Pressley et al (1989, p.301) também observaram que "muitos estudantes estão empenhados em estratégias ineficazes... Além disso, a avaliação profissional não é suficiente de técnicas que são recomendadas na literatura, com muitas estratégias propostas obsoletas.”.

Para explicar melhor sobre o tema, vamos mostrar um pouco sobre a teoria cognitivista da aprendizagem. Começando por cognição, o que seria?

“Cognição é o processo através do qual o mundo de significados tem origem. À medida que o ser se situa no mundo, estabelece relações de significação, isto é, atribui significados à realidade em que se encontra.” (BOCK et al., 1999, p. 117)

O cognitivismo está preocupado com a compreensão, transformação, armazenamento e utilização das informações. O processo de integração de material à estrutura cognitiva e a organização das informações é o que os cognitivistas chamam de aprendizagem, e essa abordagem apresenta, referente à qualidade de aprendizagem, duas formas: a mecânica e a significativa.

A aprendizagem mecânica é explicada da seguinte maneira:

“Refere-se à aprendizagem de novas informações com pouca ou nenhuma associação com conceitos já existentes na estrutura cognitiva.” (BOCK et al., 1999, p. 118)

Um exemplo clássico de uma aprendizagem mecânica seria se você simplesmente decorasse um texto de outra língua, ou seja, você iria decorar sem associar o conteúdo a nada que você já possuísse em sua estrutura cognitiva, nesse caso nem entenderia o texto, apenas o decoraria. Esse tipo de aprendizagem é criticado por muitos educadores, pois a informação é passada de forma desconexa, e a pessoa passa a receber uma grande quantidade de informação ao qual ela não consegue enxergar a aplicabilidade, ou seja, apenas decora sem saber usá-la na prática.

Para alguns críticos, esse tipo de aprendizagem não gera conhecimento, podendo até mesmo ser prejudicial à aprendizagem. Como Paulo Freire faz uma comparação dos educandos com sistemas bancários, quando todas as informações são depositadas na cabeça dos educandos de maneira forçada e memorizadas de maneira mecânica:

> A narração, de que o educador é o sujeito, conduz os educandos à memorização mecânica do conteúdo narrado. Mais ainda, a narração os transforma em "vasilhas", em recipientes a serem "enchidos" pelo educador. Quanto mais vá "enchendo" os recipientes com seus "depósitos", tanto melhor educador será. Quanto mais se deixem docilmente "encher", tanto melhores educandos serão (FREIRE, 1970, p. 59).

No caso da aprendizagem significativa o conteúdo já se relaciona com conceitos que a pessoa possui.

> Processa-se quando um novo conteúdo (ideias ou informações) relaciona-se com conceitos relevantes, claros e disponíveis na estrutura cognitiva, sendo assim assimilado por ela. Estes conceitos disponíveis são os pontos de ancoragem para a aprendizagem. Por exemplo, nós estamos aqui apresentando a você um novo conceito – o de aprendizagem significativa. Para que este conceito seja assimilado por sua estrutura cognitiva, é necessário que a noção de aprendizagem apresentada pelos cognitivistas já esteja lá, como ponto de ancoragem. (BOCK et al., 1999, p. 119)

Mesmo após uma aprendizagem significativa infelizmente o conteúdo aprendido não ficará na memória por muito tempo. Estamos constantemente aprendendo e se uma informação não é utilizada durante um longo período de tempo a tendência é que nosso cérebro a deixe de lado para armazenar outros conhecimentos que são aprendidos. Mas existe uma forma de manter o conhecimento adquirido sempre "fresco" na memória, e isso é possível fazendo revisões.

Seria muito inviável ter que reaprender algo sempre que esse determinado conteúdo seja requisitado. Manter a estabilidade do que foi aprendido é uma meta importante, e podemos averiguar se o que foi aprendido se mantém na memória do indivíduo.

Esse trabalho apresenta o aplicativo Aprendiz, que auxilia no processo de revisão e estabilidade do que foi aprendido, por meio da técnica de repetição espaçada. O aplicativo é destinado a estudantes que desejam revisar conteúdos e a professores que buscam uma ferramenta para identificar pontos de deficiência de aprendizagem de cada um de seus alunos, dispondo de funções táticas para auxiliar o professor a traçar seu plano de ação para a sala de aula.

Foi em 1885 que Hermann Ebbinghaus revolucionou a Psicologia Experimental com a publicação de um artigo intitulado *"Úber das Gedachtnis"* ("Em Memória"). Nesse trabalho, ele enunciou princípios sobre a retenção da memória e demonstrou que as memórias têm diferentes tempos de duração, além de avaliar a capacidade e a facilidade de recuperação do material memorizado.

Entre seus experimentos, ele elaborou uma longa lista de sílabas aleatórias (ex.: daus, dor, gim, ke4k, etc.), memorizando esta lista durante um ano, bem como prestando atenção em seus progressos e variando os métodos de aprendizagem. Para certificar-se dos resultados, ele repetiu a mesma experiência 3 anos mais tarde. Dessas experiências, nasceram as primeiras noções da curva de aprendizagem, do esquecimento e da técnica de repetição espaçada para memorização.

Nos capítulos inicias deste trabalho, são apresentados alguns conceitos da técnica de repetição espaçada, assim como o trabalho com *flashcards*. E ainda são listadas algumas das principais ferramentas do mercado que fazem uso dessas técnicas.

Na sequência é apresentado o problema identificado para o processo de memorização e descrito o objetivo da solução apresentada neste trabalho e nos capítulos que se seguem é abordado o planejamento e execução dessa solução.

Por fim, convergimos na conclusão avaliando a solução apresentada para o problema.

Complementam este documento os apêndices de A a R.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo encontram-se pesquisas, embasamentos teóricos e relatos obtidos de artigos científicos, documentários, dentre outros documentos sobre a hipotética curva do esquecimento e as técnica de aprendizagem espaçada e repetição espaçada.

Visando a concretização do objetivo proposto, as atividades tiveram início com a busca de aplicações reais do uso da técnica de repetição espaçada, nas quais foi identificado o uso da técnica nos sistemas Duolingo e *Anki*.

2.1 A CURVA HIPOTÉTICA DO ESQUECIMENTO

A curva do esquecimento descreve a capacidade que o ser humano possui em reter as informações/conhecimentos recém-adquiridas e de consolidá-las na memória de longo prazo.

"O esquecimento sempre acontece de forma progressiva, este processo é conhecido como 'Curva do Esquecimento'" (EBBINGHAUS, 1962).

Nesse contexto, surgiu a necessidade de conhecer e estudar a hipotética curva do esquecimento proposta por Ebbinghaus, exibida na FIGURA 1 a seguir, que demonstra o funcionamento da memória de longo prazo humana.

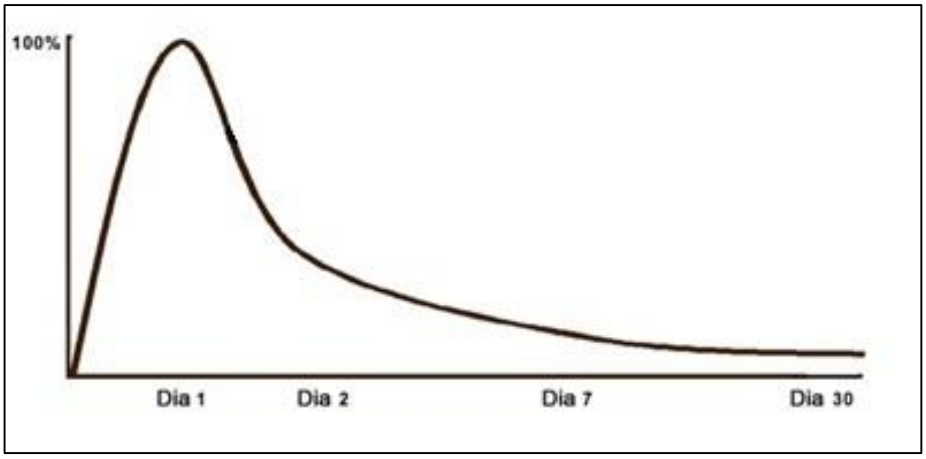

FIGURA 1 - Hipotética curva do esquecimento de Ebbinghaus
FONTE: ADAPTADO DE DELL'ISOLA (2008)

A FIGURA 1 apresenta um gráfico onde o eixo vertical representa a quantidade de informação memorizada pelo indivíduo, e o eixo horizontal a quantidade de tempo passada em relação ao dia

em que o material foi estudado. O ponto mais alto da curva no eixo vertical representa o momento em que o indivíduo finalizou o estudo de um material qualquer, quando a porcentagem hipotética de memorização do conteúdo neste momento é de 100%. Nas primeiras 24 horas após o estudo - entre o primeiro e segundo dia na FIGURA 1 -, é possível verificar o período de maior declínio da curva, significando que a maior parte do conteúdo estudado no dia anterior foi esquecido. O declínio da curva se mantém ao longo do gráfico, diminuindo gradualmente a velocidade de esquecimento.

2.2 APRENDIZAGEM ESPAÇADA

A aprendizagem espaçada é um método de estudo sistemático que segundo Ana Lopes - do programa Mais Aprendizagem - consiste em três etapas, sendo elas: estudo, revisão e aplicação do conteúdo, sempre com intervalos de 10 minutos durante cada mudança de fase. (LOPES, 2012)

Essas etapas podem ocorrer no mesmo dia ou até mesmo durar algum período de tempo, sendo feitas revisões constantemente, podendo até mesmo ser utilizada a técnica de repetição espaçada. "O efeito de espaçamento refere-se à melhoria na retenção em longo prazo de apresentações espaçadas do mesmo material escolar ou de outro tipo de material em relação a apresentações compactas ou maciças" (PINTO, 2001, p. 3).

Na FIGURA 2 a seguir são exibidas as atividades genéricas da aprendizagem espaçada.

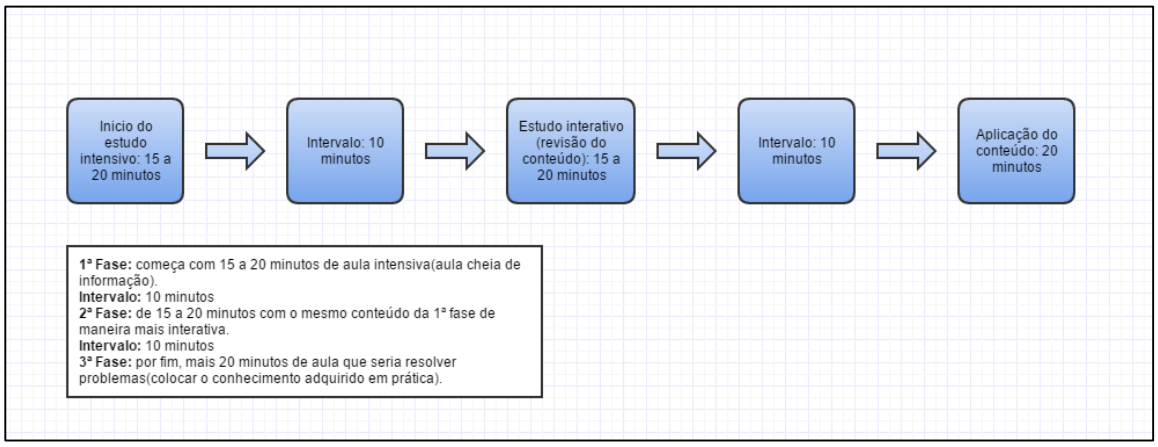

FIGURA 2 - Atividades genéricas da aprendizagem espaçada

2.3 REPETIÇÃO ESPAÇADA

A técnica de repetição espaçada é derivada da aprendizagem espaçada, que por sua vez, é baseada na curva do esquecimento de Ebbinghaus. Essa técnica consiste em realizar repetidas revisões de um material em intervalos de tempo e duração pré-definidos, com a ideia de restabelecer todo o material estudado na memória de longo prazo, nos momentos em que o indivíduo está mais suscetível a esquecê-lo.

> É evidente que a técnica de Repetição Espaçada cita, basicamente, apenas que há um momento ideal para as revisões, ela não considera as questões individuais de aprendizagem e nem as características de estudo de cada indivíduo, mas serve como parâmetro para que seja sempre lembrado um determinado assunto no momento em que o estudante está mais suscetível em esquecê-lo (SILVA; CARNIELLO; CARNIELLO, 2014, p. 2).

A memorização é importante pelo fato de que o objetivo de estudar um material na maioria das vezes é retê-lo na memória de longo prazo, e a não retenção demanda a necessidade de estudar o material novamente, uma ou mais vezes, se tornando o estudo cada vez mais dispendioso e cansativo. Nesta situação, a técnica de repetição espaçada pode ser utilizada para aumentar a eficiência da retenção de informações na memória de longo prazo.

A FIGURA 3 a seguir apresenta o gráfico da curva do esquecimento de Ebbinghaus apresentado na FIGURA 1 na seção 2.1, com a adição de uma nova curva laranja, representando a nova porcentagem de memorização do indivíduo com a utilização da técnica de repetição espaçada durante os primeiros 30 dias.

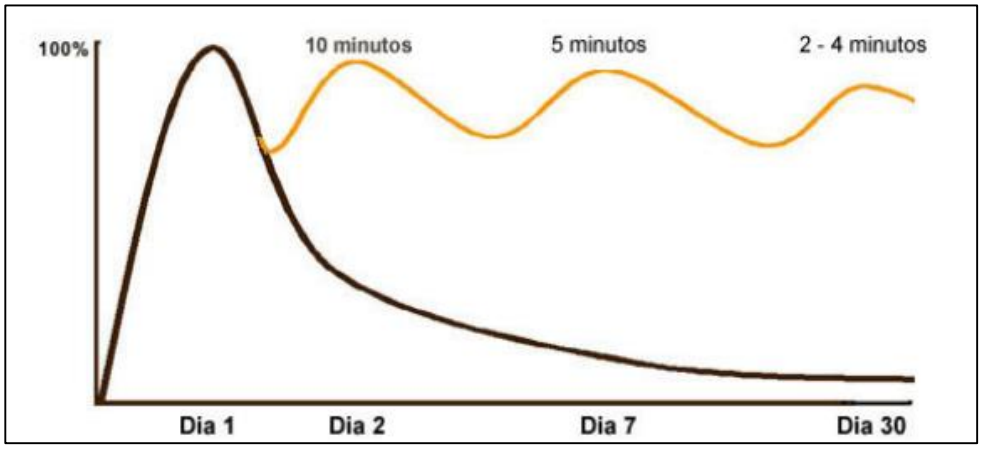

FIGURA 3 - Hipotética curva do esquecimento de Ebbinghaus com o uso da técnica de repetição espaçada
FONTE: DELL'ISOLA (2008)

A técnica de repetição espaçada funciona da seguinte maneira: no primeiro dia o indivíduo realiza o estudo inicial de um material qualquer, retendo hipoteticamente 100% das informações na memória, como pode ser visualizado no primeiro pico da curva de aprendizagem na FIGURA 3. Após 24 horas do estudo inicial o indivíduo deve realizar a primeira revisão com duração de 10 minutos para cada hora do primeiro estudo, que é o suficiente para restabelecer a memória de todo o material estudado inicialmente. Após 7 dias do estudo inicial o indivíduo deve realizar a segunda revisão com duração de 5 minutos para cada hora do primeiro estudo. E após 30 dias do estudo inicial o indivíduo deve realizar a terceira revisão com duração de 2 a 4 minutos para cada hora do primeiro estudo, atingindo mais uma vez a porcentagem hipotética de 100% de memorização do material estudado inicialmente. Desta forma o indivíduo deve continuar realizando revisões do material em intervalos de tempo cada vez mais espaçados. A cada revisão realizada, a velocidade do esquecimento é diminuída, ou seja, o tempo para que o material estudado seja esquecido aumenta, desta forma aumentando também o intervalo de tempo para a próxima revisão.

2.4 FLASHCARDS

O termo *Flashcards* vem do inglês, *Flash* (rápido/instantâneo) e *Card* (Cartão) que consiste em pequenos cartões que auxiliam no processo de memorização de um conteúdo. Acredita-se que os primeiros *flashcards* foram idealizados pela inglesa Favell Lee no ano de 1834 (PRUZAN, 2008, p. 5).

Em 1972, foi criado o sistema Leitner, baseado no uso de *flashcards*, que consiste na utilização de caixas enumeradas para o armazenamento dos cartões, e quanto melhor memorizado um cartão estiver, maior é o número da caixa em que este fica armazenado.

Os *flashcards* – ou simplesmente cartões – normalmente são feitos de cartolina, criados pelo próprio estudante, que escreve de um lado do cartão uma pergunta relacionada ao material a ser estudado e no verso a resposta. No cartão pode conter figuras e cores para auxiliar no processo de aprendizagem. Não existe um padrão de criação dos cartões, ficando a critério do estudante escolher o melhor modelo para seus estudos.

Deck (em português: baralho) é um conjunto de *flashcards* organizados por categoria, por exemplo, por disciplina, por assunto, dentre outras. Não existe limite para a quantidade de *flashcards* em um *deck*.

A contribuição das tecnologias para educação tem melhorado e muito a qualidade do ensino. Um dos motivos consideráveis é a possibilidade de colocar em prática todo conceito teórico aprendido mesmo com a falta dos recursos físicos, devido ao avanço tecnológico, que possibilita simular estes recursos virtualmente.

2.5 APLICATIVOS DE REPETIÇÃO ESPAÇADA

Como referência de *softwares* que fazem o uso das técnicas de repetição espaçada e *flashcards*, pode-se citar o sistema de ensino de idiomas Duolingo e o sistema Anki.

No Duolingo, o acesso para o uso da técnica de repetição espaçada, encontra-se na opção “Palavras” do menu principal, mostrado na FIGURA 4. O sistema mostra todas as palavras aprendidas durante o estudo, sendo possível rever as palavras aprendidas, verificar a tradução, data da última prática e o grau de confiança de cada palavra.

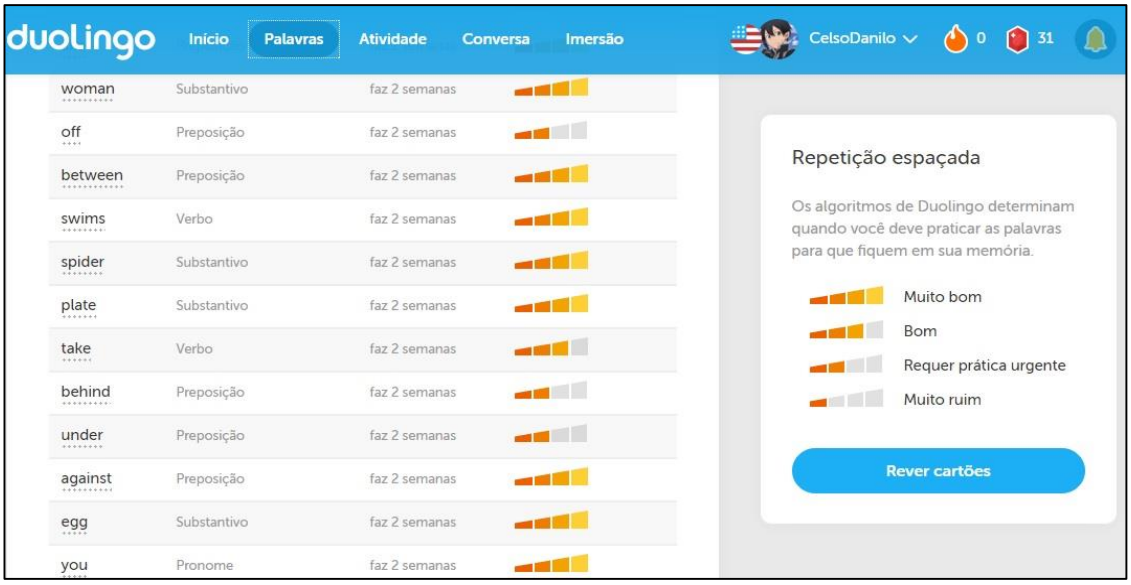

FIGURA 4 - Repetição espaçada no sistema Duolingo
FONTE: DUOLINGO (2016)

O sistema Anki permite a criação e o compartilhamento de questionários para a prática de memorização, podendo exportar este questionário criado pelo usuário e ser importado por outro usuário. Os *flashcards* podem conter mídias, como imagem e áudio.

Algumas das principais características do Anki são: o sistema é *open source*, possibilita a sincronização dos questionários em diversos dispositivos, flexível para personalização dos cartões e cada questionário pode conter até 100.000 *flashcards.* A FIGURA 5 mostra um exemplo de um *flashcard* no Anki.

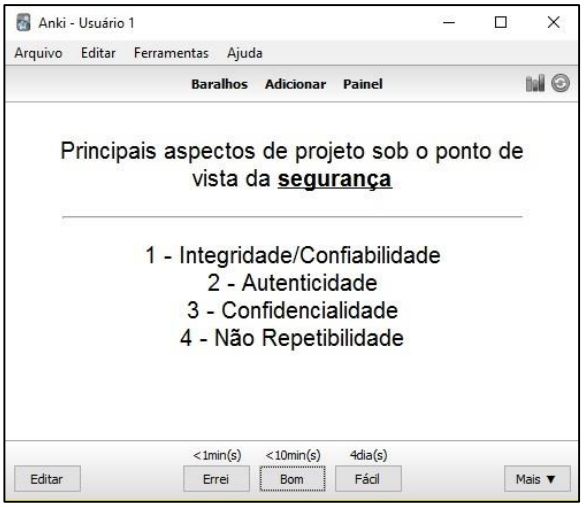

FIGURA 5 - Exemplo de um *flashcard* no sistema Anki
FONTE: Anki (2016)

3 OBJETIVO DO PROJETO

Neste capítulo são apresentados os problemas encontrados a serem solucionados, os objetivos do projeto, o público-alvo, o porquê do desenvolvimento de um sistema que utiliza a técnica de repetição espaçada e, por fim, os níveis de decisão e grupos funcionais atendidos.

3.1 FORMULAÇÃO DO PROBLEMA

Desenvolvida a partir de diversos experimentos pelo alemão Hermann Ebbinghaus, a curva hipotética do esquecimento demonstra o funcionamento da memorização de longo prazo humana, segundo a qual nas primeiras 24 horas, acontece um grande declínio na porcentagem de memorização de um conteúdo estudado, continuando a diminuir gradualmente com o tempo.

> Um dos principais objetivos de um estudante é o de memorizar o máximo possível o conteúdo das disciplinas estudadas. Porém, com o passar do tempo, o esquecimento é inevitável e o estudante acaba tendo que estudar todo o conteúdo novamente caso seja necessário a utilização deste conteúdo (SILVA; CARNIELLO; CARNIELLO, 2014, p.1).

Considerando a importância de serem retidos na memória de longo prazo determinadas informações, surgiu a necessidade da contribuição com um aplicativo a fim de despertar no estudante o interesse de trabalhar mais a memória, até a informação ser consolidada na memória de longo prazo e não ser mais esquecida.

3.2 OBJETIVOS

O objetivo geral do projeto é desenvolver um aplicativo denominado Aprendiz, que faça o uso da técnica de repetição espaçada para melhorar a eficiência de retenção de informações na memória de longo prazo do usuário.

Os objetivos específicos são:

a) auxiliar nos estudos por meio de uma ferramenta computacional simples e de boa usabilidade;

b) ampliar conhecimentos do uso da técnica de repetição espaçada e aplicá-las ao sistema;

c) auxiliar professores na aplicação de métodos de ensino mais dinâmicos;

d) criar um método interativo de estudo, para revisar conteúdos a fim de não esquecê-los.

3.3 JUSTIFICATIVA

Estudar sem se preocupar em revisar o conteúdo, pode ser inútil. Com um uso da técnica de repetição espaçada, é possível melhorar a eficiência da retenção de informações na memória de longo prazo, baseando-se na curva do esquecimento de Ebbinghaus.

De acordo com Ebbinghaus (1885), se não fizermos essas revisões, nosso cérebro vai deixar a informação de lado para poder armazenar outras informações que nos são apresentadas durante o dia a dia. O modelo ideal de estudos seria, então, fazer revisões no momento em que o cérebro está se preparando para esquecer, dessa forma, a informação passa a ser vista como importante e passa de nossa memória de trabalho para a memória de longo prazo.

3.4 PÚBLICO-ALVO

Estudantes em geral poderão usufruir da aplicação, desde estudantes de idiomas, do ensino fundamental, médio, superior ou mesmo pessoas que estejam estudando para vestibulares ou concursos. Além disso, professores também poderão utilizar a aplicação para elaborar aulas mais dinâmicas e coletar resultados de grupos de estudo - a aplicação terá funções que possibilitará o estudo coletivo - e ter uma visão mais ampla sobre o conhecimento de cada aluno.

3.5 NÍVEIS DE DECISÃO E GRUPOS FUNCIONAIS ATENDIDOS

O nível de decisão atendido pelo projeto é o nível Operacional. A FIGURA 6 ilustra os níveis e seus respectivos grupos funcionais.

A atuação do usuário no nível operacional se dá através da criação de *flashcards* e *decks* para revisão dos conteúdos que foram aprendidos, remoção desse material de revisão criado pelo usuário e a própria revisão em si.

Atualmente o usuário não possui recursos gerenciais no aplicativo, mas pretende-se que seja implementado alguns requisitos desejáveis, como por exemplo, a criação de salas de estudo, e a partir dessa sala, o usuário possa fazer análises de seu desempenho e de outros usuários da sala, a fim de analisar o desempenho durante o período em que esteja sendo aplicado o processo de memorização com a repetição espaçada.

FIGURA 6 - Níveis de decisão do projeto

4 GERÊNCIA DO PROJETO

Neste capítulo são apresentados os processos necessários para o planejamento e gestão do projeto.

4.1 PLANO DE PROJETO

Nesta seção são apresentadas as partes interessadas no projeto, o modelo de ciclo de vida a ser adotado, os recursos necessários e as estimativas de tempo para conclusão do projeto.

4.1.1 Partes interessadas

O projeto possui quatro partes interessadas, detalhadas a seguir:

a) Instituição de ensino superior, FAI - Centro de Ensino Superior em Gestão, Tecnologia e Educação (FAI), com o papel de orientar e avaliar os alunos durante a execução do PFC do curso de Sistemas de Informação.
   Interesse: baixo.
   Poder: alto.

b) Equipe do projeto Aprendiz (Celso Danilo da Mota, Francisco de Faria Cardoso, Juliano Costa Silva, Marcelo Pereira e Robson dos Santos), com o papel de desenvolver o sistema.
   Interesse: alto.
   Poder: alto.

c) Educadores, com o papel de experimentar o uso do sistema como ferramenta de auxílio no ensino, com finalidade de aumentar a eficiência da memorização de informações de seus alunos.
   Interesse: baixo.
   Poder: baixo.

d) Estudantes, com o papel de experimentar o sistema, com finalidade de aumentar a eficiência na memorização de informações.
   Interesse: baixo.
   Poder: baixo.

A FIGURA 7 a seguir apresenta as partes interessadas, classificadas na matriz de poder x interesse.

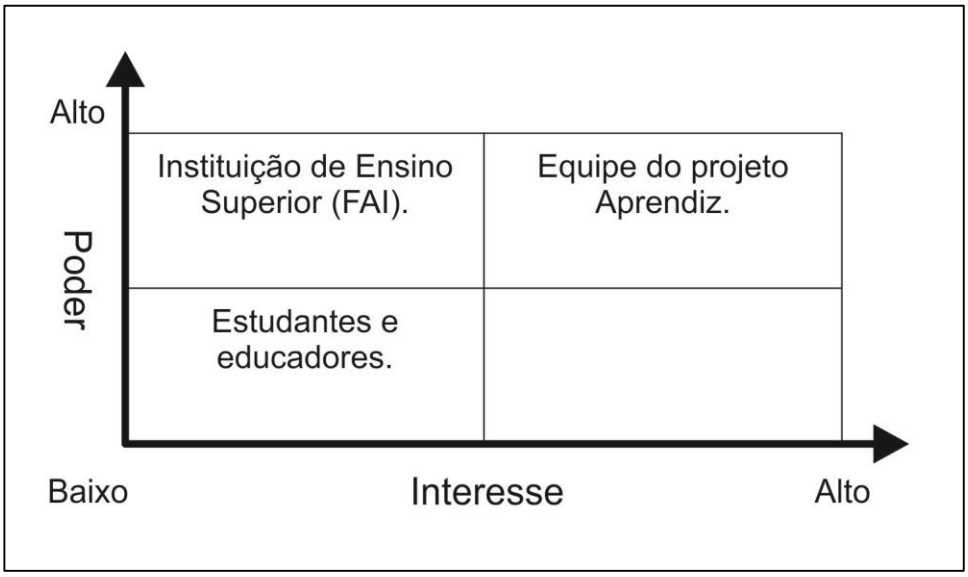

FIGURA 7 - Matriz poder x interesse do projeto

4.1.2 Modelo de ciclo de vida

O modelo de processo adotado para o desenvolvimento do projeto é o Incremental, que consiste em um processo de desenvolvimento iterativo entre as atividades de arcabouço do projeto. Neste processo, o projeto é dividido em etapas, denominadas incrementos, e desta forma o sistema é construído etapa por etapa, sendo entregue uma versão funcional do sistema na finalização de cada etapa/incremento.

Em cada incremento é realizado todo o ciclo de desenvolvimento de *software*, do planejamento aos testes do sistema em funcionamento. As entregas do projeto estão divididas em 4, detalhadas a seguir.

a) Fase 1 - 02 de Abril de 2016
   i. Capítulo 1 – Introdução.
   ii. Capítulo 2 – Revisão bibliográfica.
   O Capítulo 2 deverá ter no mínimo sete obras citadas como embasamento (Referências), com os seguintes conteúdos:

a. Fundamentos necessários para o entendimento do trabalho, com a citação de pelo menos cinco obras conceituadas (artigos, teses, livros) que subsidiem a apresentação desses fundamentos.
b. Trabalhos relacionados (mínimo de dois trabalhos, com a citação das fontes de referência) que apresentam soluções já disponíveis sobre o tema tratado.

iii. Capítulo 3 – Objetivos do projeto e as suas seções.
iv. Capítulo 4 – Gerência do projeto
    a. Plano de projeto e suas subseções “Partes Interessadas”, “Modelo de Ciclo de Vida” e “Recursos Necessários”. Na primeira fase é opcional a entrega das “Estimativas por Pontos de Função” e “Pontos de Casos de Uso”.
    b. Áreas de conhecimento e as subseções Escopo e Tempo.

    Na primeira fase é opcional a entrega das seções “Gestão da Qualidade” e “Gestão dos Riscos”. Quanto à “Gestão da Integração”, pelo menos a planilha com Relatório de Desempenho precisa ser entregue, como apêndice.

v. Capítulo 5 – Especificação e análise de requisitos e as suas seções.
vi. Referências.

b) Fase 2 – 25 de Junho de 2016
    i. Correções da 1a. fase.
    ii. Completar o Capítulo 4 com:
        a. Estimativas Por Pontos de Função e de Pontos de Casos de Uso
        b. Gestão dos Riscos
        c. Gestão da Qualidade
        d. Integração
    iii. Capítulo 6 – Arquitetura e Projeto e suas seções
        a. Visão Estrutural.
        b. Visão Comportamental.
        c. Visão de Dados.
        d. Projeto de Sistemas Distribuídos.

e. Projeto da Interação Humano-computador (somente Perfis de usuários).

Na segunda fase é opcional a entrega das seções “Visão Física e de Distribuição”, “Padrões de Projeto”, “Análise de complexidade” e “Projeto da Interação Humano-Computador (Aspectos visuais e heurísticas de usabilidade)”.

iv. Pelo menos 30% dos casos de uso essenciais devem estar codificados para a primeira apresentação oral e controle de integração implantado.
v. Referências atualizadas.

c) Fase 3 – 10 de Setembro de 2016
   i. Correções das Fases 1 e 2.
   ii. Completar o Capítulo 6 "Arquitetura e Projeto" com as seções “Visão Física e de Distribuição”, “Padrões de Projeto” e “Projeto da Interação Humano-Computador”.

   Na terceira fase é opcional a entrega da “Análise de Complexidade Algorítmica”.

   iii. Pelo menos 60% dos casos de uso essenciais devem estar codificados para a segunda apresentação oral.
   iv. Referências atualizadas.

d) Fase 4 - 12 de Novembro de 2016
   i. Correções das Fases 1, 2 e 3.
   ii. Resumo.
   iii. Capítulo 6 – Adicionar “Análise de Complexidade Algorítmica”.
   iv. Capítulo 7 – Plano de Testes.
   v. Capítulo 8 – Plano de Implantação.
   vi. Capítulo 9 – Conclusão.
   vii. Referências, Obras Consultadas e Glossário.
   viii. Casos de uso essenciais codificados e concluídos.

A documentação do projeto deve seguir as normas indicadas nas “Diretrizes para elaboração de trabalhos científicos - 4a. edição”, disponível para consulta no site da FAI e na Biblioteca.

4.1.3 Recursos necessários

Nesta seção são apresentados os recursos necessários para a execução do projeto, como recursos humanos, de hardware e de software.

4.1.3.1 Recursos humanos

Para a execução do projeto, a divisão de responsabilidades é a seguinte:

a) Celso Danilo da Mota, pesquisador e gerente de projeto;

b) Francisco de Faria Cardoso, pesquisador e desenvolvedor;

c) Juliano Costa Silva, pesquisador e arquiteto de sistema;

d) Marcelo Pereira, pesquisador e revisor do memorial;

e) Robson dos Santos, pesquisador e desenvolvedor.

O projeto também conta com a coordenação da professora da instituição de ensino FAI, Eunice Gomes de Siqueira.

4.1.3.2 Recursos de hardware

Os recursos de hardware necessários para a fase de desenvolvimento do sistema são:

5 Computadores (ou *notebooks*) para desenvolvimento, com configuração mínima de *hardware*:

i. Processador: Intel Pentium III 800MHZ ou equivalente.

ii. Memória RAM: 1 GB.

iii. Memória Permanente: 160 GB.

iv. 4 entradas padrão USB.

v. 1 teclado, com entrada USB.

vi. 1 *mouse*, com entrada USB, com no mínimo 3 botões - incluindo o botão *scroll*.

vii. 1 *headset*, para reuniões virtuais – desejável.

Para as fases de teste e implantação, são necessários:

a) 1 servidor para hospedagem, com configurações de *hardware* mínima:

   i. Processador: 1GHz.

   ii. Memória RAM: 2GB.

   iii. Memória Permanente: 100GB.

b) Pelo menos 1 aparelho com sistema operacional *Android*, versão 4.0 ou superior.

4.1.3.3 Recursos de software

Os recursos de *software* necessários para o desenvolvimento são:

a) Navegador (Google Chrome e/ou Mozilla Firefox), para realização de pesquisas na *internet* e utilização de *softwares web*.

b) Pacote Office, para documentação do projeto.

c) Google Drive, para edição simultânea e compartilhamento dos artefatos do projeto pela *internet*.

d) *Visual Paradigm*, para criação dos diagramas do projeto.

e) *Microsoft Project*, para gestão do cronograma do projeto.

f) IDE Eclipse, para o desenvolvimento da aplicação cliente.

g) IDE Visual Studio, para o desenvolvimento da aplicação servidora.

h) Postman, aplicativo do Google Chrome para realização de testes da aplicação servidora.

i) *BlueStacks*, para realização de testes da aplicação cliente (simulação de plataforma *Android*).

j) PostgreSQL, para o gerenciamento da base de dados.

k) PgAdmin, para o gerenciamento do Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL.

l) *Software* para criação de imagens.

m) RiouxSVN, para controle de versão das aplicações cliente e servidora.

4.1.4 Estimativas de tempo

A estimativa de tempo para o desenvolvimento do sistema foi dividida para as aplicações cliente conforme o QUADRO 1 e a servidora conforme o QUADRO 2. As métricas utilizadas foram por pontos de função e por pontos de casos de uso. Os documentos com os cálculos podem ser consultados no Apêndice P, e os resultados das estimativas podem ser visualizadas nos quadros a seguir.

4.1.4.1 Aplicação cliente

Métrica	2ª. Fase	3ª. Fase (Reestimativa)	4ª. Fase (Reestimativa)
Pontos por função	2721 HH	2721 HH	1763 HH
PCU - *Schneider e Winters*	1695 HH	1695 HH	1034 HH
PCU - *Karner*	1211 HH	1211 HH	739 HH
PCU - *Sparks*	1695 HH	1695 HH	1034 HH

QUADRO 1 - Estimativa de esforço da aplicação cliente

Uma alteração foi feita entre a 3ª e 4ª fase – sendo a retirada de praticamente todas as entidades que seriam armazenadas no banco de dados da aplicação cliente, o que causou uma grande diferença nas estimativas. Também foi feita alteração na pontuação da concorrência, considerando a necessidade da criação de *threads* para o controle das notificações, e foi removido o caso de uso “Avaliar questionário”, diminuindo a estimativa de esforço da aplicação cliente.

4.1.4.2 Aplicação servidora

Métrica	2ª. Fase	3ª. Fase (Reestimativa)	4ª. Fase (Reestimativa)

Pontos por função	2792 HH	2792 HH	2465 HH
PCU - *Schneider e Winters*	1034 HH	1034 HH	1021 HH
PCU - *Karner*	739 HH	739 HH	729 HH
PCU - *Sparks*	1034 HH	1034 HH	1021 HH

QUADRO 2 - Estimativa de esforço da aplicação servidora

As estimativas da aplicação servidora se mantiveram estáveis. Pequenos ajustes foram feitos nos fatores de complexidade técnica e ambiental da estimativa por pontos de casos de uso, aproximando a complexidade para a realidade do projeto. E também o caso de uso “manter avaliação de questionário” foi removido na 4ª fase, diminuindo as estimativas de esforço.

4.2 ÁREAS DO CONHECIMENTO

Nesta seção são apresentadas as áreas de conhecimento para a gestão do projeto, contemplando a gestão de escopo, do tempo, da integração, da qualidade e dos riscos do projeto.

4.2.1 Gestão de escopo

Nesta seção é apresentada a estrutura analítica do projeto (EAP) e o dicionário da EAP para o gerenciamento do escopo do projeto.

4.2.1.1 Estrutura analítica do projeto

A Estrutura Analítica do Projeto (EAP) consiste em um diagrama que apresenta o projeto como um todo, dividido em partes no formato de um organograma. Cada retângulo representa uma entrega, e os retângulos que se encontram no fim da hierarquia são chamados de pacote de trabalho. Cada pacote de trabalho engloba um conjunto de atividades que será necessário ser realizado para concluí-lo. A EAP é utilizada para a definição e gerenciamento do escopo do projeto.

A EAP deste projeto pode ser visualizada na FIGURA 8 a seguir.

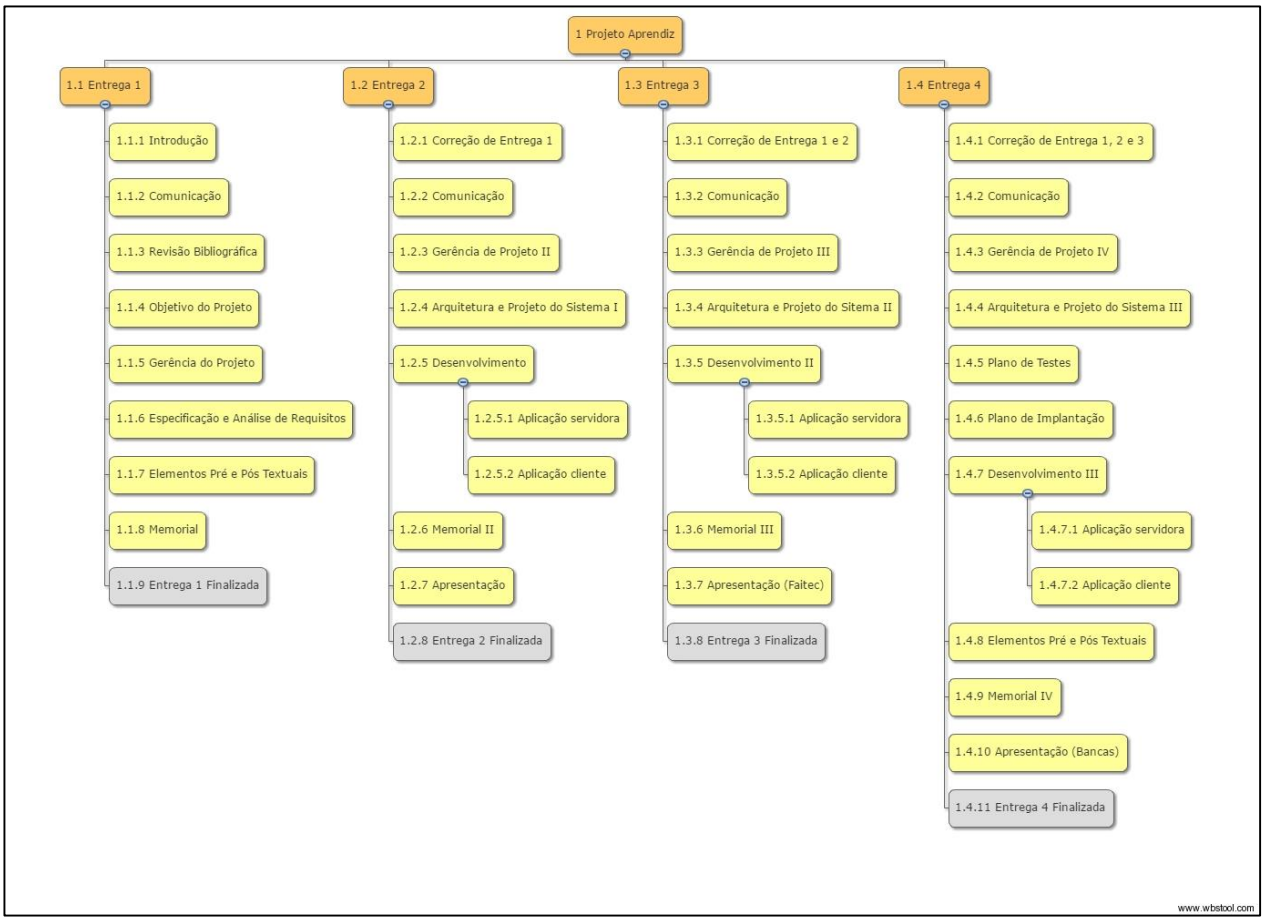

FIGURA 8 - EAP do projeto

4.2.1.2 Dicionário EAP

O dicionário da EAP fornece a descrição de cada pacote de trabalho. O dicionário da EAP deste projeto pode ser consultado no Apêndice A.

4.2.2 Gestão do tempo

Nesta seção são apresentados os processos necessários para o gerenciamento do tempo do projeto, como a lista de atividades a serem realizadas, diagrama de rede das atividades, cronograma das atividades e o quadro resumo.

4.2.2.1 Lista de atividades

A lista de atividade apresenta todas as atividades a serem feitas durante o projeto. A lista de atividades deste projeto pode ser consultada no Apêndice A.

4.2.2.2 Diagrama de rede

O diagrama de rede utilizado no projeto é o *Activity On Node* (AON), onde cada atividade do projeto é representado em um quadro (*node*), e a dependência entre as atividades é representada por uma seta interligando uma atividade à outra.

O diagrama de rede do projeto pode ser consultado no Apêndice A.

4.2.2.3 Cronograma de atividades

O cronograma do projeto apresenta a data de início, término, duração e responsáveis por cada atividade do projeto. Também apresenta a dependência entre as atividades.

O Cronograma do projeto pode ser consultado no Apêndice A.

4.2.2.4 Quadro-resumo

O QUADRO 3 a seguir apresenta a comparação de tempo de realização de cada etapa do projeto. Nela é possível visualizar as estimativas de tempo:

a) *ad-hoc*, sendo a quantidade de horas estimadas antes da criação das atividades e cronograma do projeto;

b) dicionário EAP, sendo o total de horas das atividades previsto nos pacotes de trabalho; e

c) efetivo realizado, sendo o total de horas gastas para a realização das atividades.

	1ª. Fase	2ª. Fase	3ª. Fase	4ª. Fase
Estimativa *ad-hoc*	100	200	200	200
Dicionário EAP	184	232	173,7	224,9
Efetivo realizado	~120/120	~100/220	~200/420	~250/670

QUADRO 3 - Esforço planejado x realizado do projeto

É possível visualizar no QUADRO 3 que as estimativas *ad-hoc* e do dicionário EAP em média ficaram próximas uma da outra. E o tempo efetivo realizado foi bastante inferior aos tempos

estimados. Uma possível explicação é o tempo disponível limitado dos integrantes da equipe para dedicação ao projeto.

4.2.3 Gestão da integração

Nesta seção são apresentados os processos necessários para o gerenciamento da integração do projeto, como o monitoramento do desempenho do projeto, controle de configuração e controle de mudanças do projeto.

4.2.3.1 Monitoramento

Com finalidade de monitorar o andamento do projeto e para que os coordenadores e avaliadores (parte interessada) tenham conhecimento do andamento do projeto, é utilizado o relatório de desempenho, que apresenta de forma superficial o estado do projeto nas seguintes categorias:

a) Documentação.

b) Software.

c) Escopo.

d) Tempo.

e) Desempenho na fase.

O relatório de desempenho deste projeto pode ser consultado no Apêndice B.

4.2.3.2 Controle de configuração

Toda equipe do projeto é responsável pela gerência de configuração, sendo assim, é de responsabilidade de todos manterem os artefatos em redundância - armazenado na nuvem (Google Drive) e no computador. Os artefatos mantidos no Google Drive são: arquivos de banco de dados, diagramas, documentos de apoio, entregas do memorial e pesquisas, como pode ser visualizado na FIGURA 9 a seguir.

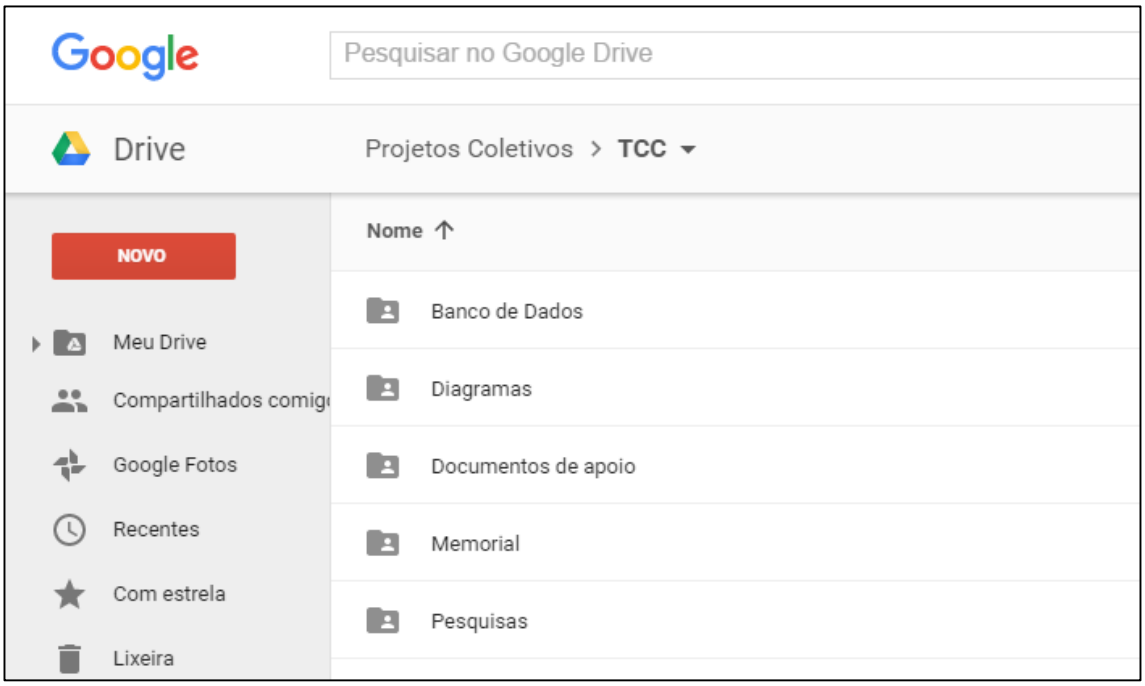

FIGURA 9 - Pasta raiz compartilhada do repositório do projeto no Google Drive

A cada entrega é criada uma subpasta dentro da pasta Memorial, identificada com a nomenclatura: <nº da entrega>, como pose ser visualizado na FIGURA 10 a seguir.

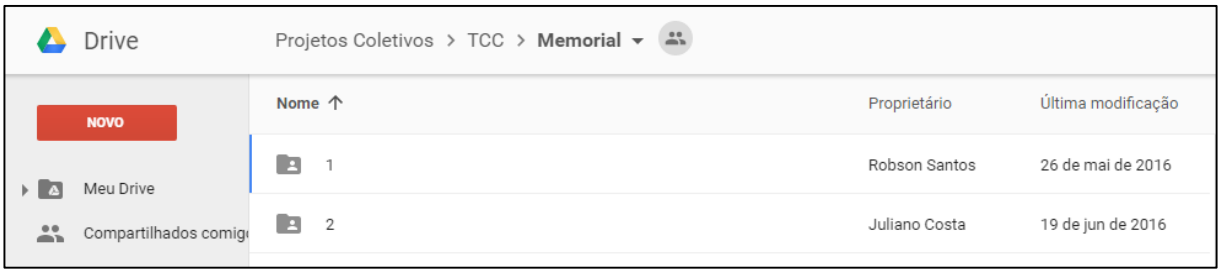

FIGURA 10 - Identificação das entregas do projeto no repositório

Além da pasta compartilhada no *Google Drive*, é utilizado o sistema online *One Drive* para a edição compartilhada do memorial do projeto. Após a finalização de cada versão do memorial – para entrega, é realizada uma cópia para o *Google Drive* e computadores, mantendo sempre *backups* das versões do documento. A FIGURA 11 a seguir apresenta a edição online e simultânea do memorial.

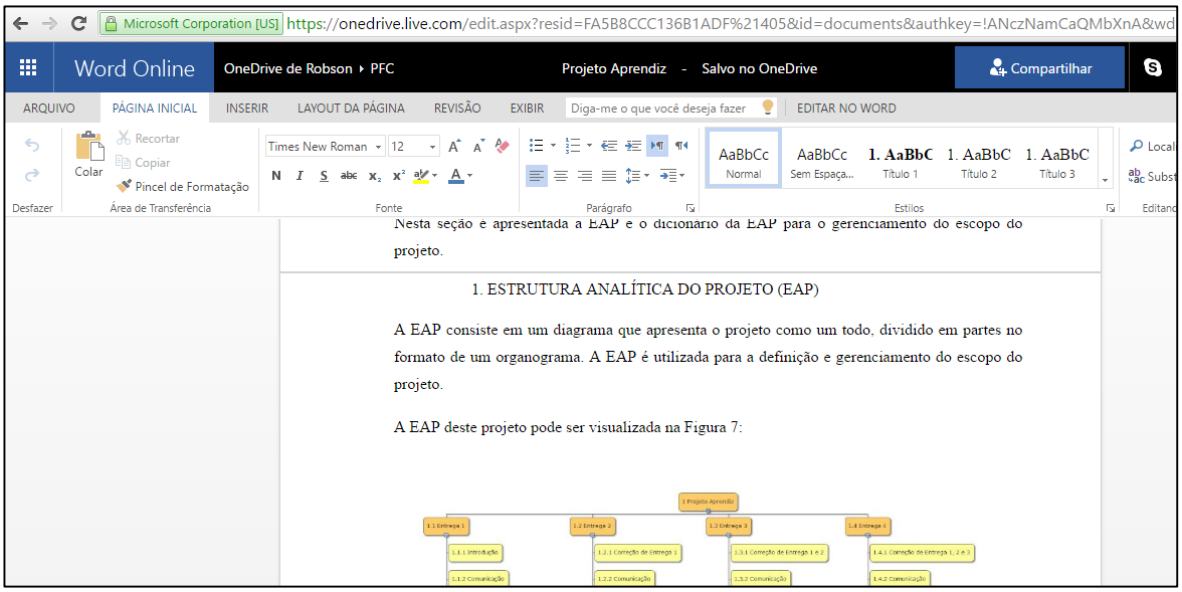

FIGURA 11 - Repositório do memorial do projeto no One Drive

RiouxSVN é o repositório e controlador de versão utilizado pela equipe de desenvolvimento do projeto. Esse repositório disponibiliza gratuitamente um total de 200MB de armazenamento – divido em pacotes de no máximo 50MB. Na FIGURA 12 é possível visualizar os recursos oferecidos pelo RiouxSVN.

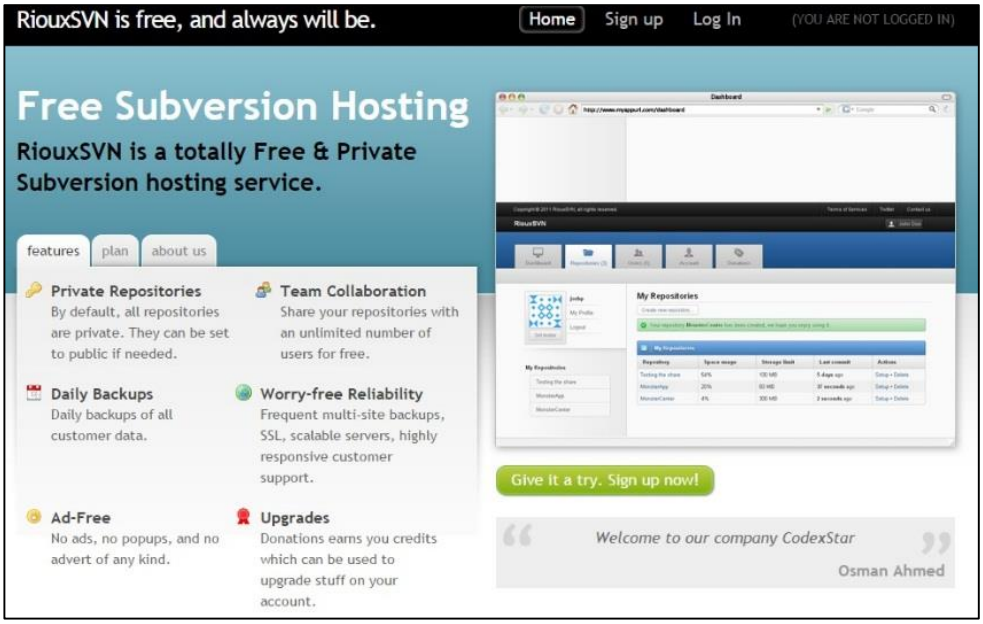

FIGURA 12 - Recursos oferecidos pelo RiouxSVN

O repositório da aplicação cliente está organizado de maneira que “Aprendiz Mobile” seja a pasta principal/raiz, onde dentro da mesma se encontra uma subpasta chamada “Aprendiz” com todos os artefatos da construção da aplicação cliente, como pode ser visualizado na FIGURA 13 a seguir.

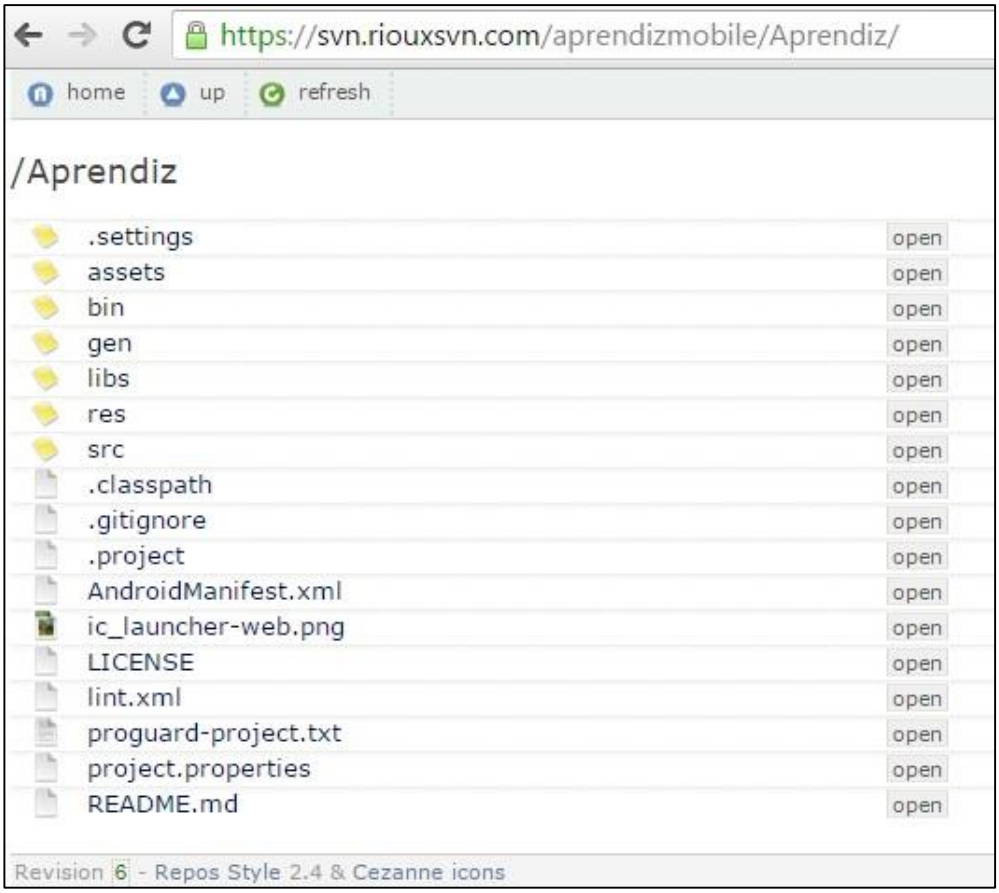

FIGURA 13 - Repositório da aplicação cliente

O repositório da aplicação servidora está organizado de maneira que “APRENDIZWS” seja a pasta principal/raiz, onde dentro da mesma se encontra uma subpasta chamada “aprendizws” com todos os artefatos da construção da aplicação servidora, como pode ser visualizado na FIGURA 14 a seguir.

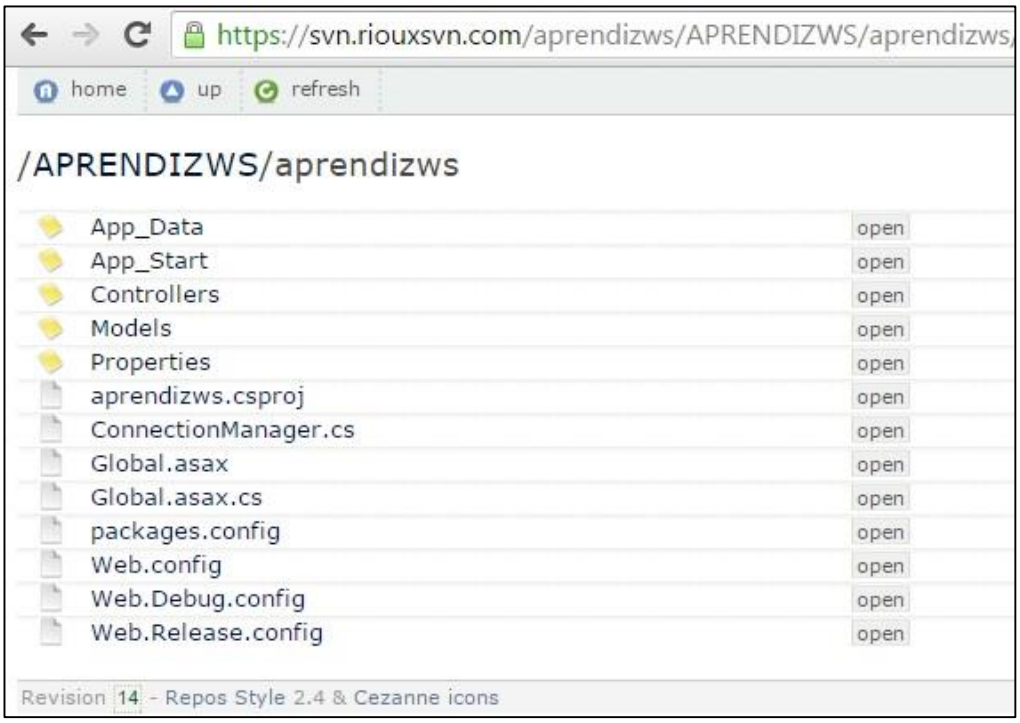

FIGURA 14 - Repositório da aplicação servidora

Além da pasta dos artefatos da construção da aplicação servidora, existe também uma pasta chamada “TESTES”, que possui os artefatos produzidos durante testes realizados na fase de desenvolvimento.

4.2.3.3 Controle de mudanças

No controle de mudanças, ocorrem as revisões de todas as solicitações de mudança ou modificações nos documentos do projeto, entregas, linhas de base ou no plano de gerenciamento do projeto, sendo as mudanças aprovadas ou rejeitadas.

As solicitações de mudanças podem ser feitas por todos *stakeholders*, seguindo um ciclo formal de comunicação. O processo de mudanças é iniciado com uma solicitação formal - exigindo uma explicação clara e detalhada sobre as alterações no escopo do projeto, em seguida é realizada uma análise para avaliar possíveis impactos técnicos e gerenciais para verificação se será aprovada ou não. Após a análise, caso a mudança seja aprovada, é realizado um replanejamento de forma que inclua as alterações no escopo do projeto, e em seguida todas as partes interessadas são

formalmente informadas para dar início a execução das mudanças. A FIGURA 15 mostra as etapas principais deste processo.

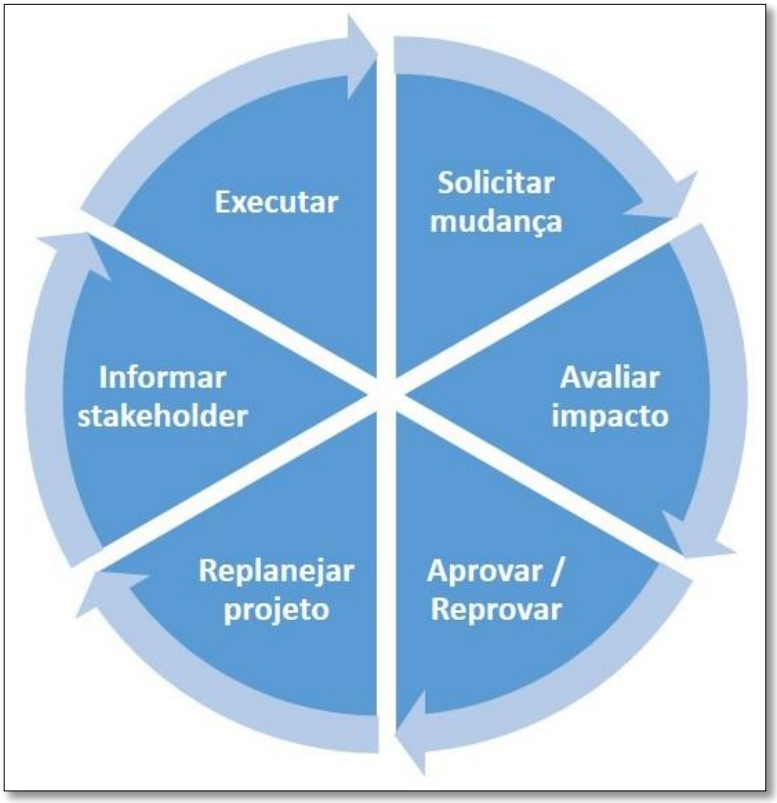

FIGURA 15 - Etapas do controle de mudanças do projeto

Por fim, o controle de mudanças é executado de forma que atenda aos requisitos pré-estabelecidos (prazos, custos, qualidade, riscos, etc.). O documento de controle de mudanças deste projeto pode ser consultado no Apêndice C.

4.2.4 Gestão da qualidade

O gerenciamento da qualidade é realizado pelos integrantes e gerentes de projeto Celso Danilo da Mota e Juliano Costa Silva, que realizam o monitoramento e controle dos processos afim de garantir que o projeto atenda aos requisitos de funcionalidade, de produto e organizacionais, bem como atende as restrições de escopo, prazos, custos e riscos do projeto.

> O gerenciamento da qualidade do projeto inclui os processos e as atividades da organização executora que determinam as políticas de qualidade, os objetivos e as responsabilidades, de modo que o projeto satisfaça às necessidades para as quais foi empreendido (PMI, 2013, p. 227).

O documento que contém os itens a serem gerenciados neste projeto pode ser consultado no Apêndice D.

4.2.5 Gestão dos riscos

Segundo o PMI, o plano de gerenciamento dos riscos é de fato fundamental na comunicação e deve ser iniciado no início do projeto, ainda na fase de planejamento, pois aumenta a probabilidade da exatidão das demais atividades do projeto (PMI, 2013, p. 314).

A gestão dos riscos visa explorar alguns aspectos que geram impactos no objetivo do projeto e gerenciá-los, podendo ser esses riscos de espécie ameaçadora ou de oportunidade.

> Planejar o gerenciamento dos riscos é o processo de definição de como conduzir as atividades de gerenciamento dos riscos de um projeto. O principal benefício deste processo é que ele garante que o grau, tipo, e visibilidade do gerenciamento dos riscos sejam proporcionais tanto aos riscos quanto à importância do projeto para a organização (PMI, 2013, p. 313).

O documento de gerenciamento dos riscos deste projeto pode ser consultado no Apêndice E.

5 ESPECIFICAÇÃO E ANÁLISE DE REQUISITOS

Neste capítulo são apresentados os requisitos funcionais e não funcionais, a visão funcional e a visão de dados do projeto.

5.1 DESCRIÇÃO DE REQUISITOS

Nesta seção são apresentadas as funcionalidades da aplicação cliente e servidora, com a definição de entrada, processamento, saída, restrições e prioridade de cada requisito.

5.1.1 Requisitos funcionais

5.1.1.1 Aplicação cliente

[RF-1] Manter usuário

O sistema deve fornecer a função de cadastro de usuário na tela inicial do sistema.

No cadastro deve ser exibido os campos: “Nome”, “*Eletronic-mail* (*E-mail)*”, “Nome de Usuário”, “Senha”, “Confirmação de Senha”, e as opções “Cadastrar” e “Cancelar”. Após o usuário preencher os campos e selecionar a opção “Cadastrar”, o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve confirmar o cadastro e redirecionar para a página inicial de usuário;

b) caso os dados forem inválidos, o sistema deve permanecer na tela de cadastro e informar o usuário quais campos estão preenchidos com dados inválidos.

Após o cadastro, o sistema deve oferecer as funções de acesso (*login* e *logout*), atualização e remoção da conta.

Prioridade: essencial.

[RF-2] Acessar página inicial de usuário

Após a realização de *login*, o sistema deve redirecionar-se para a página inicial do usuário, com as seguintes funcionalidades:

a) Menu lateral (visível apenas quando solicitado):
    i. Informações.
    ii. Editar perfil.
    iii. Sair.
b) Acessar opção de questionários:
    i. Acessar questionários.
    ii. Adicionar questionário.
c) Acessar opção de salas de estudo:
    i. Acessar sala de estudo.
    ii. Adicionar sala de estudo.
d) Acessar aba de notificações:
    i. Aceitar/Recusar convite de sala de estudo.
    ii. Acessar questionário para revisão.

Prioridade: essencial.

[RF-3] Solicitar recuperação de senha

O sistema deve fornecer a função de recuperação de senha ao usuário. Ao acionar a funcionalidade, o sistema deve exibir uma tela com os campos: "*E-mail*" e as opções "Enviar" e "Cancelar". Após o usuário preencher os campos e selecionar a opção "Enviar", o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve enviar um *e-mail* com o código de alteração de senha para o *e-mail* do usuário, com os passos para a alteração da senha (descritos no requisito RF-20);

b) caso os dados forem inválidos, o sistema deve permanecer na tela atual e informar o usuário quais campos estão preenchidos com dados inválidos.

Prioridade: essencial.

[RF-4] Manter questionário

O sistema deve oferecer a opção para que o usuário crie questionários. Na criação, devem ser exibidos os campos: “Nome”, “Público/Privado”, e as opções “Criar” e “Cancelar”. Após o usuário preencher os campos e selecionar a opção “Criar”, o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve confirmar o cadastro e redirecionar para a tela de criação de cartões;

b) caso os dados forem inválidos, o sistema deve permanecer na tela de cadastro e informar o usuário quais campos estão preenchidos com dados inválidos.

Após a criação do questionário, o sistema deve oferecer as funções de acesso, atualização e remoção do questionário (juntamente com cartões associados).

Na busca de questionários públicos, os resultados devem ser apresentados em ordem de melhor avaliação, calculada pela soma dos indicadores.

Prioridade: essencial.

[RF-5] Manter cartão

Após a criação de um questionário, o sistema deve oferecer a opção para que o usuário crie cartões dentro do questionário. Na criação, devem ser exibidos os campos: “Nome”, “Pergunta”, “Resposta”, “Ativado” e as opções “Criar” e “Cancelar”. Após o usuário preencher os campos e selecionar a opção “Criar”, o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve confirmar o cadastro, e retornar para a tela anterior;

b) caso os dados forem inválidos, o sistema deve permanecer na tela de cadastro e informar o usuário quais campos estão preenchidos com dados inválidos.

Após a criação do cartão, o sistema deve oferecer as funções de acesso, atualização e remoção.

Prioridade: essencial.

[RF-6] Realizar estudo/revisão de questionário

Após o usuário acessar um questionário, deve existir a função de estudo/revisão do questionário. O estudo consiste na apresentação dos cartões (perguntas e respostas) que o usuário precisa revisar.

Durante a exibição dos cartões, devem ser mostrados na tela:

a) Pergunta do cartão.

b) Opção “lembro”.

c) Opção “Não lembro”.

A cada vez que o usuário selecionar uma das opções, a resposta do cartão deve ser exibida, e logo em seguida o próximo cartão deve ser exibido.

Após todos os cartões serem exibidos, deve ser exibida uma tela com o resultado do estudo/revisão com as informações:

a) Número de respostas lembradas.

b) Número de respostas não lembradas.

Após a exibição dos resultados do estudo/revisão, o sistema deve redirecionar-se para a tela de lista de questionários.

A cada vez que o usuário selecionar a opção de realizar o estudo do questionário, a aplicação deve fazer uma análise dos estudos anteriores do usuário e selecionar os cartões que são necessários serem revisados.

O algoritmo utilizado para a revisão foi baseado no "Sistema Leitner" criado por Sebastian Leitner em 1972. Nesse sistema Leitner propôs o controle das revisões com espaçamentos invariáveis e com quantidade fixa.

> No sistema Leitner, os flashcards ficam em caixas. Quando é respondido corretamente, o cartão passa para a caixa seguinte. Os cartões da caixa seguinte são revistos em intervalos maiores do que os da primeira caixa, e assim sucessivamente. Os cartões respondidos de forma errada voltam para a caixa anterior (ZNP, 2016)

Com base nesse sistema será desenvolvido um algoritmo que apresenta as revisões nos seguintes intervalos: após 1 hora, após 1 dia, após 1 semana, após 1 mês e após 1 trimestre, e ainda deverá realizar a correção do intervalo, diminuindo 10% a cada erro. Essa correção do intervalo será necessária após ser considerado a informação que se tem hoje de que a facilidade de memorização de informações varia de acordo com a complexidade do conteúdo, fatores ambientais, dentre outros fatores.

Prioridade: essencial.

[RF-7] Acessar estatísticas do questionário

Cada questionário deve possuir estatísticas públicas e cada usuário deve possuir suas estatísticas privadas - relacionadas ao questionário que está estudando.

a) Estatísticas públicas (questionários públicos):

   Informações públicas acessadas por qualquer usuário.

   i. Número de usuários que acessaram o questionário.

   ii. Número de usuários que visualizaram todos os cartões do questionário.

   iii. Número de salas de estudo que usaram/usam o questionário.

   iv. Porcentagem de acerto/erro geral dos usuários.

b) Estatísticas privadas (usuário que estuda o questionário):

Informações acessadas pelo usuário que está estudando o questionário, e também pelo supervisor da sala de estudo (caso o usuário pertença a uma sala de estudo, e o questionário em

questão esteja vinculado à sala de estudo). Cada usuário deve possuir suas estatísticas em cada questionário.

i. Tempo estudado.

ii. Número de cartões lembrados/não lembrados.

iii. Porcentagem de cartões lembrados/não lembrados.

iv. Cartões memorizados.

v. Cartões não memorizados.

vi. Número de revisões realizadas.

Prioridade: desejável.

[RF-8] Manter sala de estudo

O sistema deve fornecer a função de criação de sala de estudo pelo próprio usuário. Na criação da sala, devem ser apresentados os campos: “Nome”, “Disponibilizar visualização de estatísticas de participantes” e as opções “Criar” e “Cancelar”. Após o usuário preencher os campos e selecionar a opção “Criar”, o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve confirmar o cadastro, e redirecionar o usuário para a sala de estudo;

b) caso os dados forem inválidos, o sistema deve permanecer na tela de cadastro e informar o usuário quais campos estão preenchidos com dados inválidos.

Após a criação da sala, o sistema deve oferecer as funções de acesso, atualização e remoção da sala.

Prioridade: desejável.

[RF-9] Inserir usuário em sala de estudo, por convite

Após a criação de uma sala de estudo, o sistema deve fornecer a opção de inserção de usuários na sala, por meio de envio de convite.

Deve ser exibida uma tela com os campos: "*E-mail/Nome de Usuário*" e as funções "Convidar" e "Cancelar". Após o usuário preencher os campos e selecionar a opção "Convidar", o sistema deve validar os dados de entrada, e:

a) caso os dados forem válidos, o sistema deve enviar o convite para o usuário informado, e retornar para a tela anterior;

b) caso os dados forem inválidos, o sistema deve permanecer na tela atual e informar o usuário quais campos estão preenchidos com dados incorretos.

Prioridade: desejável.

[RF-10] Entrar em sala de estudo pelo convite

Após o usuário receber um convite para entrar em uma sala de estudo, este convite deve ser visível para o usuário na tela de notificações. Ao visualizar o convite, o usuário terá as opções de:

a) aceitar (o sistema deve confirmar a vinculação do usuário à sala de estudo);

b) recusar.

Após o usuário escolher uma das opções, o convite não deve ser mais visível para o usuário.

Prioridade: desejável.

[RF-11] Inserir usuário na sala de estudo, por geração de código

Após a criação de uma sala de estudo, o sistema deve fornecer a opção de inserção de usuários na sala, pela geração de um código temporário, com duração definida pelo usuário.

Após a solicitação de geração do código, o criador da sala divulga o código para os usuários que queiram entrar na sala, e então, cada usuário insere este código na área de cadastro de sala de estudo, confirmando o cadastro. (Um exemplo é um professor exibindo o código em uma sala escolar física, para que os alunos possam se cadastrar na sala de estudo do sistema).

Prioridade: desejável.

[RF-12] Entrar em sala de estudo pelo código

O sistema deve oferecer a função de o usuário se cadastrar em uma sala de estudo - previamente criada por outro usuário – a partir do código divulgado pelo criador da sala. O sistema deve exibir uma tela com os campos: “Código da sala de estudo” e as opções “Entrar” e “Cancelar”. Após o usuário preencher os campos e selecionar a opção “Entrar”, o sistema deve verificar se existe uma sala de estudo vinculada ao código inserido,

a) caso exista, o sistema confirma o cadastro do usuário na sala e redireciona para a sala de estudo; e

b) caso não exista, o usuário é informado de que o código inserido não está vinculado a nenhuma sala de estudo.

Prioridade: desejável.

[RF-13] Remover usuário de sala de estudo

Após a inserção de um ou mais usuários em uma sala de estudo, deve existir uma função para a remoção do usuário da sala de estudo, em que somente o criador da sala possui permissão.

Prioridade: desejável.

[RF-14] Inserir questionário na sala de estudo

Após a criação de uma sala de estudo, o sistema deve fornecer a opção de inserção de questionários na sala, exibindo a opção “Inserir questionário”. Após o usuário selecionar a opção, o sistema deve exibir uma tela com os campos: “Busca de questionário” e as opções “Adicionar” e “Cancelar”. Após o usuário selecionar o questionário e a opção “Adicionar”, o sistema deve vincular o questionário à sala de estudo.

Prioridade: desejável.

[RF-15] Remover questionário de sala de estudo

Após a inserção de um ou mais questionários em uma sala de estudo, deve existir uma função para a remoção do questionário, em que somente o criador da sala possui permissão.

Prioridade: desejável.

[RF-16] Acessar notificações

O sistema deve oferecer a função de notificações, para:

a) convite para entrar em uma sala de estudo;

b) necessidade de revisão de conteúdo, baseado no algoritmo de repetição espaçada, descrito no requisito funcional “Realizar estudo/revisão de questionário”.

Prioridade: importante.

[RF-17] Acessar informações úteis da aplicação

O sistema deve fornecer uma área de informações gerais sobre a aplicação. Informações que explicam:

a) Curva do esquecimento.

b) Técnica de repetição espaçada.

c) *Flashcards*.

d) Sistema Leitner.

e) Funcionamento do aplicativo.

Prioridade: desejável.

[RF-18] Realizar comunicação com aplicação servidora

O sistema deve realizar solicitações de serviços para a aplicação servidora e tratar as respostas recebidas, fazendo o uso do padrão de comunicação *Representational State Transfer* (REST).

Prioridade: essencial.

5.1.1.2 Aplicação servidora

[RF-19] Manter usuário

A aplicação servidora deve possibilitar a inclusão, acesso, alteração e remoção de cadastro de usuário na base de dados. Na inclusão e alteração de cadastro, deve ser mantida a restrição de que os campos “usuário” e “*e-mail*” sejam únicos na base de dados.

Prioridade: essencial.

[RF-20] Recuperar senha

O servidor, ao receber a solicitação de recuperação de senha, deve verificar se o valor de “*e-mail*” enviado pelo usuário está vinculado a algum usuário cadastrado na base de dados.

a) Caso positivo, a aplicação servidora deve gerar um código de atualização de senha, gravar este código no cadastro do usuário e enviar as seguintes instruções para o *e-mail* do usuário:

    i. Na tela de alteração de senha da aplicação, insira o *e-mail* em que possui cadastro na aplicação.

    ii. No campo “código de atualização de senha” insira o código “XXX...”.

    iii. No campo “nova senha” insira sua nova senha.

    iv. Selecione a opção de confirmação para concretizar a alteração da senha.

    Após o usuário ter realizado os passos acima, e a aplicação servidora ter recebido a solicitação de atualização da senha, a aplicação servidora deve verificar se os valores “*e-mail*” e “código de alteração de senha” existem no cadastro do usuário, na base de dados.

    i. Caso positivo, deve ser gerado um código *hash* da nova senha e este deve ser inserido no campo “senha” do usuário. E então a aplicação servidora deve retornar uma mensagem de sucesso na atualização da senha para a aplicação cliente.

    ii. Caso negativo, a aplicação servidora deve retornar uma mensagem de insucesso para a aplicação cliente, informando os dados incorretos.

b) Caso negativo, a aplicação servidora deve retornar uma mensagem a aplicação cliente, informando que o *e-mail* informado não existe na base de dados.

Prioridade: essencial.

[RF-21] Manter questionário

A aplicação servidora deve possibilitar a inclusão, acesso, alteração e remoção de cadastro de questionário na base de dados.

Prioridade: essencial.

[RF-22] Manter cartão

A aplicação servidora deve possibilitar a inclusão, acesso, alteração e remoção de cadastro de cartão na base de dados.

Prioridade: essencial.

[RF-23] Manter histórico de usuário

Quando o usuário realizar revisões/estudos de cartões pela aplicação cliente, esses dados devem ser enviados para a aplicação servidora.

Prioridade: essencial.

[RF-24] Manter sala de estudo

A aplicação servidora deve possibilitar a inclusão, acesso, alteração e remoção de cadastro de sala de estudo na base de dados.

Prioridade: desejável.

[RF-25] Manter usuário em sala de estudo

A aplicação servidora deve possibilitar a inclusão, acesso e remoção de vínculo entre usuário e sala de estudo na base de dados.

Prioridade: desejável.

[RF-26] Manter questionário em sala de estudo

A aplicação servidora deve possibilitar a inclusão, acesso e remoção de vínculo entre questionário e sala de estudo na base de dados.

Prioridade: desejável.

5.1.2 Requisitos não funcionais

Nesta seção são apresentadas as restrições do sistema, por parte do produto e da faculdade (FAI).

5.1.2.1 Aplicação cliente

[RNF-1] Sistema operacional da aplicação cliente.

A aplicação cliente deve ser desenvolvida para a plataforma *Android*, a partir da versão 4.0, pelas seguintes características:

a) Possibilidade de utilizar o sistema por uma dispositivo móvel, não dependendo do computador.

b) Possibilidade da aplicação gerar notificações para o usuário em segundo plano.

[RNF-2] Tamanho limite de memória permanente da aplicação cliente.

Levando em conta o espaço limitado de memória permanente dos aparelhos móveis e também o tempo de *download*, o aplicativo cliente - na instalação inicial - deve ocupar no máximo 15 MB de memória permanente.

[RNF-3] Linguagem de programação da aplicação cliente.

Levando em conta que a aplicação cliente deve ser desenvolvida para a plataforma *Android*, a linguagem que deve ser usada é Java.

5.1.2.2 APLICAÇÃO SERVIDORA

[RNF-4] Sistema operacional da aplicação servidora.

A aplicação servidora deve ser desenvolvida para o sistema operacional Windows.

[RNF-5] Linguagem de programação da aplicação servidora.

A linguagem que deve ser usada para a aplicação servidora deve ser a C#.

[RNF-6] Servidor Web da aplicação servidora.

O servidor web da aplicação servidora deve ser o *Internet Information Services* (IIS).

[RNF-7] Tempo de disponibilidade da aplicação servidora.

Considerando que a maior parte das funcionalidades do sistema depende do correto funcionamento do servidor web, existe a necessidade de alta disponibilidade deste. Portanto, este deve permanecer disponível por pelo menos 23 horas e 30 minutos por dia (~98% do tempo).

5.2 VISÃO FUNCIONAL

Nesta seção é apresentado o diagrama de casos de uso e suas descrições.

5.2.1 Diagrama de casos de uso

O diagrama de casos de uso apresenta a visualização das funcionalidades do sistema, com a decomposição de cada requisito funcional em um ou mais casos de uso. Apresenta também o relacionamento entre os casos de uso e as funcionalidades disponíveis para cada ator do sistema.

Os diagramas de casos de uso da aplicação cliente e servidora podem ser consultados no Apêndice J.

5.2.2 Descrição dos casos de uso

A descrição de um caso de uso apresenta informações sobre as formas possíveis de utilização deste caso de uso, chamadas de cenários. Cada cenário de um caso de uso apresenta informações como pré-condição para que o caso de uso seja executado, pós-condição após a finalização do cenário, atores envolvidos, fluxo de eventos entre os atores e o sistema, etc.

Os cenários dos casos de uso deste projeto podem ser consultados no Apêndices K.

5.3 VISÃO DE DADOS

Nesta seção é apresentado o projeto lógico do sistema.

5.3.1 Projeto lógico

O modelo entidade relacionamento (MER) é uma abstração esquemática que apresenta as entidades do mundo real sobre as quais o sistema deverá armazenar dados, bem como o relacionamento entre estas entidades.

O Modelo ER deste sistema pode ser consultado no Apêndice L.

6 ARQUITETURA E PROJETO DO SISTEMA

Neste capítulo são apresentadas as visões estrutural, comportamental, de dados, física e de distribuição, padrões de projeto, análise de complexidade dos algoritmos, projeto de sistemas distribuídos e projeto da interação humano-computador.

6.1 VISÃO ESTRUTURAL

Nesta seção são apresentados os diagramas de pacotes, diagramas de classes e diagramas de objetos do sistema.

6.1.1 Diagrama de pacotes

O diagrama de pacotes apresenta a arquitetura adotada para o sistema, cada pacote representa um agrupamento de componentes. A arquitetura adotada para as aplicações cliente e servidora do projeto é a *Model View Control* (MVC), onde os componentes de controle, negócio e visualização são separados em camadas. Os diagramas de pacote das aplicações cliente e servidora podem ser visualizados nas figuras a seguir.

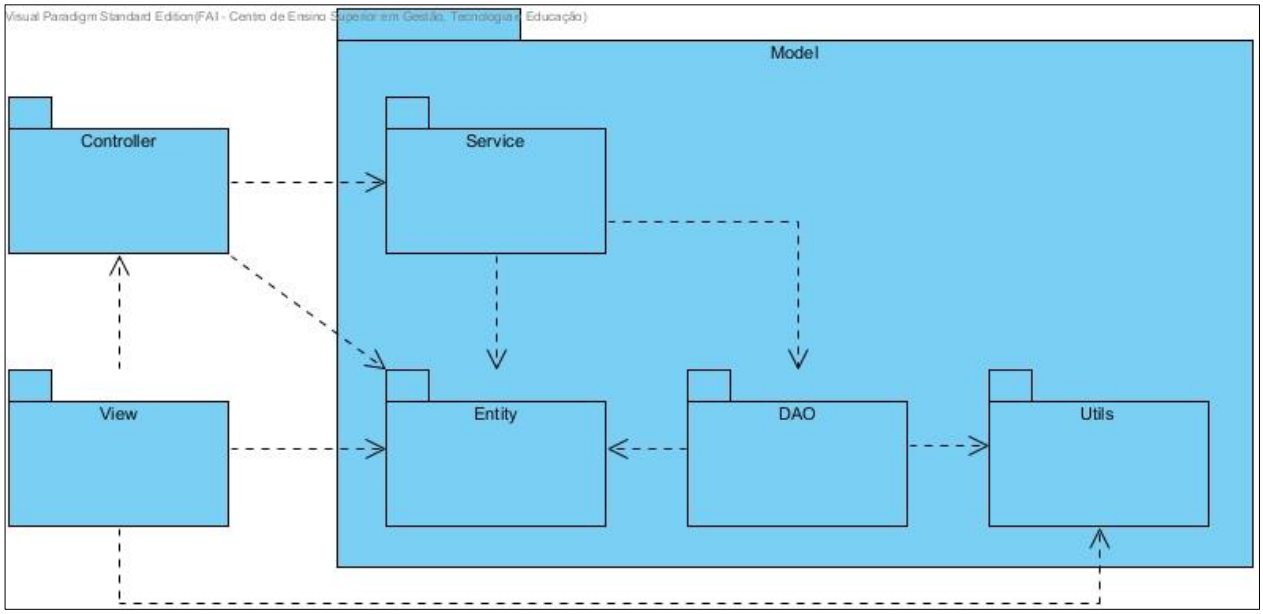

FIGURA 16 - Diagrama de pacotes da aplicação cliente

Na FIGURA 16, onde é mostrado o diagrama de pacotes da aplicação cliente, o pacote View contém as classes responsáveis por montar e exibir as telas da aplicação, e também por repassar as entradas do usuário para o pacote Controller. O pacote Controller contém as classes responsáveis por receber as solicitações, fazer o uso dos pacotes Service e Entity e retornar as respostas para o solicitante. O pacote Service contém as classes responsáveis por realizar serviços locais e fazer o uso dos pacotes DAO e Entity . O pacote DAO contém as classes responsáveis por solicitar os serviços para a aplicação servidora, manipular os dados locais e fazer o uso dos pacotes Entity e Utils. O pacote Entity possui as classes de entidade da aplicação cliente. O pacote Utils contém as classes utilitárias.

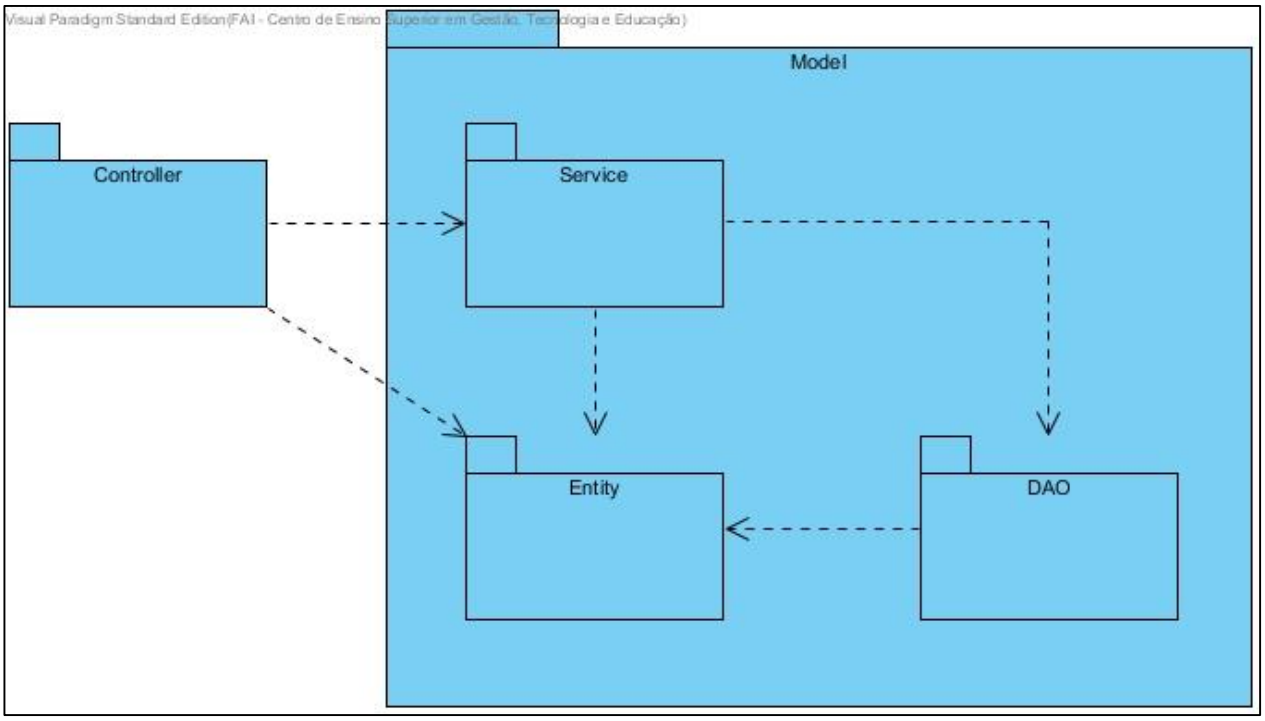

FIGURA 17 - Diagrama de pacotes da aplicação servidora

Na FIGURA 17, onde é mostrado o diagrama de pacotes da aplicação servidora, o pacote Controller contém as classes responsáveis por receber as solicitações, fazer o uso dos pacotes Service e Entity e retornar as respostas para o solicitante. O pacote Service contém os serviços do sistema, como realizar *login*, recuperar senha, criar sala de estudo, dentre outros serviços, e faz o uso do pacote DAO e Entity. O pacote DAO contém as classes de acesso ao SGBD e faz o uso do pacote Entity. O pacote Entity possui as classes de entidade da aplicação servidora.

6.1.2 Diagramas de classes

Diagramas de classes apresentam as classes que contribuem entre si a fim de cumprir os requisitos do sistema, também apresentam os relacionamentos, atributos e operações de cada classe. Os diagramas de classes do sistema podem ser consultados nos Apêndices M e N.

6.1.3 Diagramas de objetos

Diagramas de objetos apresentam os objetos do sistema com os atributos preenchidos com valores reais, simulando o estado da aplicação em um determinado momento. Os diagramas de objetos deste projeto podem ser consultados no Apêndice O.

6.2 VISÃO COMPORTAMENTAL

Nesta seção é apresentado o projeto das interações do sistema e o diagrama de máquina de estados.

6.2.1 Projeto das interações

Nesta seção são apresentados os diagramas de sequência e de visão geral de interação.

6.2.1.1 Diagramas de sequência

O diagrama de sequência permite a visualização da troca de mensagens na ordem pré-estabelecida entre os objetos do sistema. Os diagramas de sequência de cada cenário do sistema podem ser consultados nos Apêndices Q e R.

6.2.1.2 Diagrama de visão geral de interação

O diagrama de visão geral de interação possibilita a visualização dos possíveis caminhos de execução das tarefas do sistema. Consiste na visão geral das interações entre os cenários do sistema. Na FIGURA 18 a seguir é apresentado o diagrama de visão geral de interação do caso de uso “Manter usuário” da aplicação cliente.

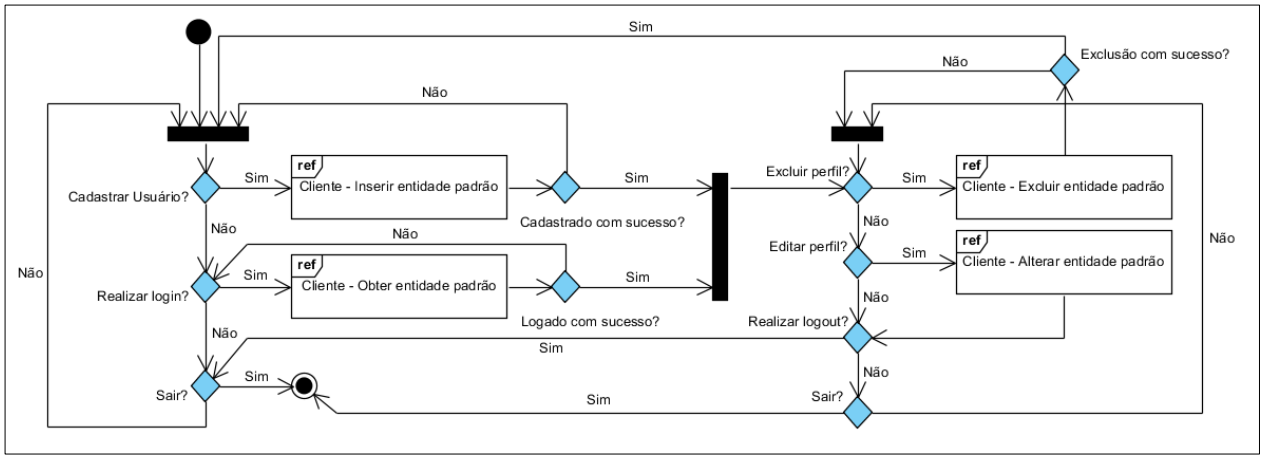

FIGURA 18 - Diagrama de interação geral do caso de uso “Manter usuário” da aplicação cliente

6.2.2 Diagrama de máquina de estados

O diagrama de máquina de estados apresenta o funcionamento de determinada parte de um sistema, alternando entre os estados a partir de eventos ocorridos. Na FIGURA 19 a seguir é apresentado o diagrama de máquina de estados do objeto cartão na aplicação cliente.

Durante a criação de um cartão o estado é de “Em criação”. Após a confirmação de criação do cartão, o estado passa para “Ativado”. Ou seja, cada cartão pode estar ou não ativado para estudo, e por padrão, após a criação de um cartão, este é ativado para estudo. Caso o usuário desative o cartão, este passará para o estado de “Desativado”. Durante um estudo o usuário escolhe uma das opções para cada cartão exibido: “lembro” ou “não lembro”, e neste momento, caso o usuário selecione a opção “não lembro” o cartão passa para o estado de “Não memorizado”, e caso o usuário selecione a opção “lembro” o cartão passa para o estado de “Memorizado”. Ao atingir o tempo para a revisão do cartão, este volta para o estado de “Ativado” indicando que está disponível para ser revisado. O usuário terá a funcionalidade de excluir cartões, e no final desta ação o estado do cartão passa para o estado de “Excluído” chegando ao fim do diagrama de máquina de estados do cartão.

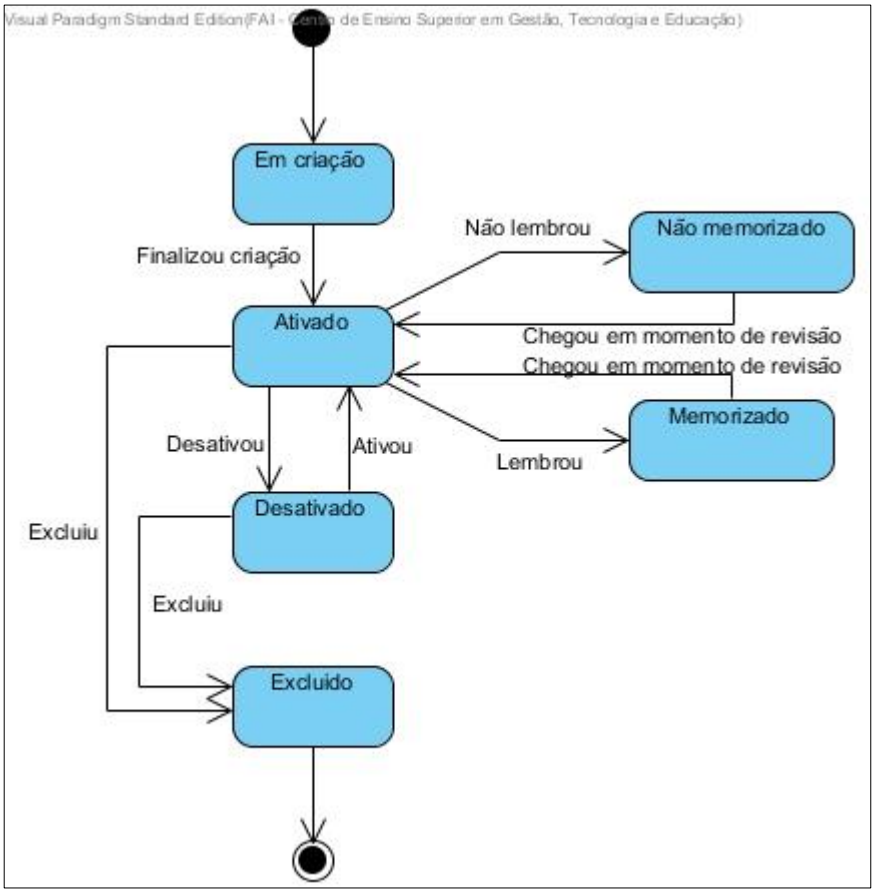

FIGURA 19 - Diagrama de máquina de estados do cartão na aplicação cliente

6.3 VISÃO DE DADOS

Nesta seção é apresentado o modelo operacional do sistema, o dicionário de dados do modelo operacional e o processo de elaboração do projeto físico do sistema.

6.3.1 Modelo operacional

O modelo operacional é utilizado para apresentar as entidades do mundo real em que se deseja armazenar dados e os relacionamentos entre as entidades, de forma que seja possível cumprir os requisitos do sistema.

O sistema possui apenas um modelo operacional, sendo da aplicação servidora. Este pode ser consultado no Apêndice L.

6.3.2 Dicionário de dados

O dicionário de dados oferece informações detalhadas do modelo operacional. O dicionário de dados do projeto pode ser consultado no Apêndice F.

6.3.3 Processo de elaboração do projeto físico

O projeto físico de dados do projeto Aprendiz segue as três formas normais a seguir:

a) A primeira forma normal diz que todos os atributos devem estar definidos em domínios que contêm valores atômicos.

b) A segunda forma normal diz que a tabela deve estar na primeira forma normal e todos os atributos não chave devem ser funcionalmente dependentes da chave na sua totalidade e não apenas de parte da chave.

c) A terceira forma normal diz que, além da tabela estar na segunda forma normal, nenhum atributo não chave depende funcionalmente de nenhum outro atributo não chave.

A nomenclatura das tabelas e dos atributos segue a mesma convenção:

a) Nomes possuem apenas letras minúsculas.

b) Nomes compostos são separados pelo caractere “_”.

6.4 VISÃO FÍSICA E DE DISTRIBUIÇÃO

Nesta seção é apresentado o diagrama de componentes e o diagrama de distribuição do sistema.

6.4.1 Diagrama de componentes

O diagrama de componentes apresenta os componentes do sistema, como bibliotecas, documentos, arquivos executáveis, dentre outros, e também o relacionamento entre eles. O diagrama de componentes do sistema Aprendiz pode ser visualizado na FIGURA 20 a seguir. Na figura é possível visualizar as duas aplicações, servidora e cliente. Os componentes da aplicação servidora são: base de dados “SGBD”, arquivo do servidor web “Aprendiz.dll”, pacotes de arquivos fonte, “Model”, “Controller”, “Service”, “Entity” e “DAO”. Os componentes da

aplicação cliente são: arquivo instalador do aplicativo “Aprendiz.apk” e pacotes de arquivos fonte, “Model”, “View”, “Controller”, “Service”, “Entity” e “DAO”.

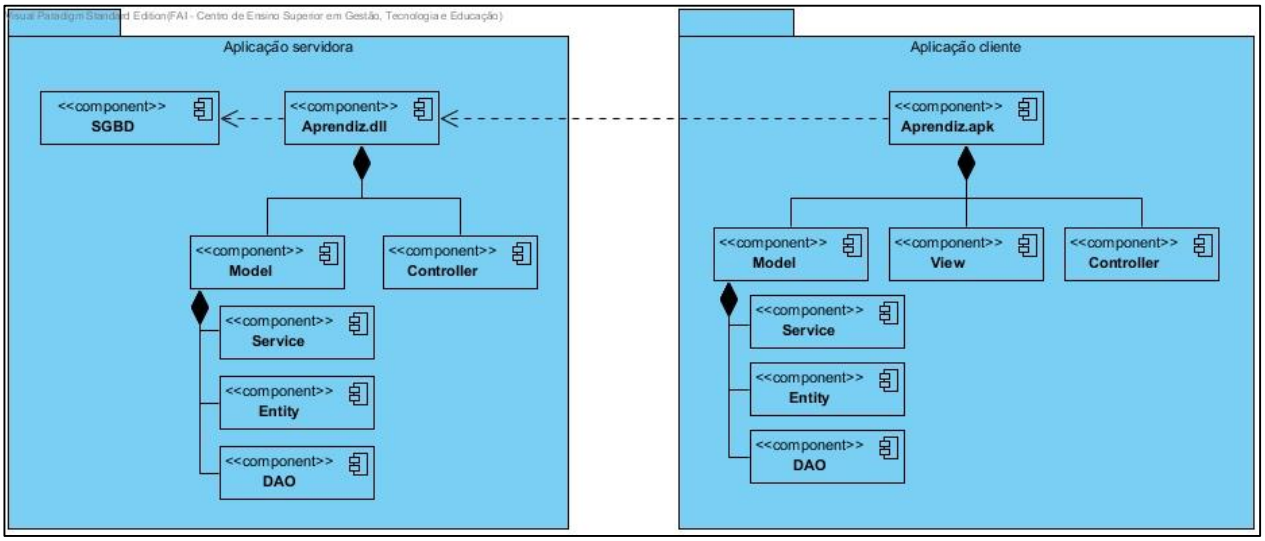

FIGURA 20 - Diagrama de componentes do sistema

6.4.2 Diagrama de distribuição

O diagrama de distribuição apresenta as partes do sistema que se interagem a fim de cumprir os requisitos do sistema. Essas partes são compostas dos recursos computacionais físicos, chamados de nós, e os recursos de software, chamados de componentes. O diagrama de distribuição do sistema pode ser visualizado na FIGURA 21 a seguir.

Na FIGURA 21 é possível visualizar a representação de dois nós, sendo o servidor à esquerda e o cliente à direita. Poderá existir apenas uma instância do servidor e zero ou mais instâncias do cliente. Os componentes do nó servidor são: sistema operacional Windows versão 7, plataforma interpretadora dotNET versão 4.5, servidor web IIS versão 7, aplicação servidora, SGBD PostgreSQL versão 9.4 e base de dados. Os componentes do nó cliente são: sistema operacional Android versão 4.0, interpretador JVM Dalvik ou *Android Runtime* (ART) e aplicativo Aprendiz.

A linha contínua interligando os nós representa a comunicação entre eles, o rótulo “<<HTTP>>” indica que a comunicação entre os nós servidor e cliente é através do protocolo HTTP. As setas tracejadas sem rótulo entre os componentes representam as dependências entre eles, como pode ser visualizado no nó servidor, o componente servidor web IIS depende do interpretador dotNET

para ser executado. E as setas tracejadas com o rótulo “<<use>>” representa a utilização que um componente faz do outro, como pode ser visualizado no nó servidor, o componente SGBD PostgreSQL faz uso do componente base de dados.

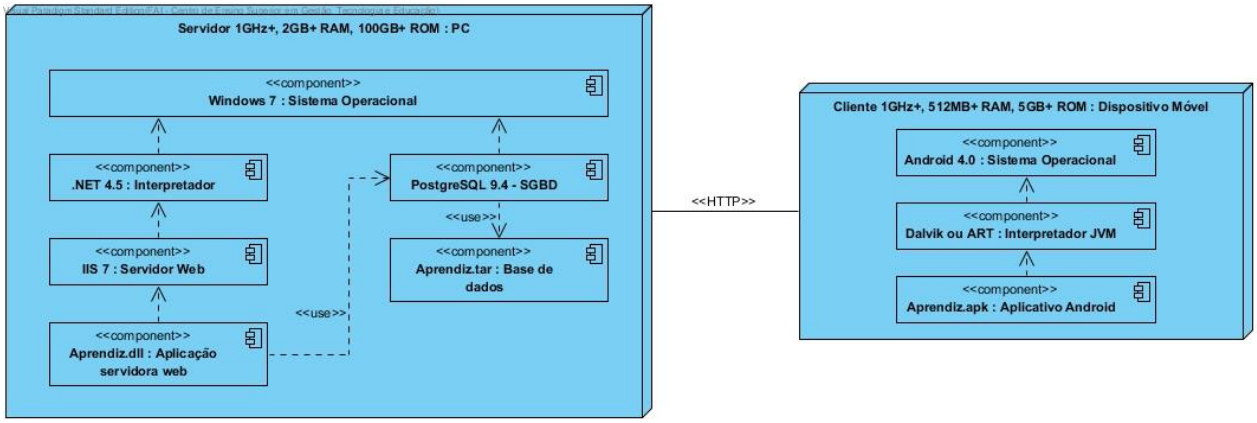

FIGURA 21 - Diagrama de distribuição do projeto

6.5 PADRÕES DE PROJETO

Nesta seção são apresentados os *design patterns* utilizados e as convenções para codificação.

Padrões de projeto são técnicas que viabilizam o reuso de código resolvendo um problema comum de diversos tipos de aplicativos. Essa técnica de modelagem de classes e objetos aperfeiçoa a comunicação entre desenvolvedores através de uma padronização.

Um padrão de projeto pode ser utilizado em diversas linguagens orientadas a objetos diferentes, de maneira simples *design patterns* é uma forma de organizar classe(s) diante determinada circunstância.

Objetivos comuns de um *design pattern*:

a) código menos extenso possível;
b) código mais legível possível;
c) código mais flexível possível;
d) divisão de responsabilidades entre os objetos;
e) reutilização de código.

6.5.1 Design patterns

O padrão arquitetural escolhido para as aplicações cliente foi o MVC (*Model, View and Controller*), para a aplicação servidora também o MVC, porém sem a camada View.
O padrão MVC organiza o sistema em três componentes:

a) *Model*: responsável por fornecer as principais funcionalidades da aplicação.
b) *View*: responsável pelos componentes que serão exibidos ao usuário.
c) *Controller*: responsável por receber as entradas do usuário e controlar o fluxo de execução da aplicação.

Propósito	Padrões utilizados	Classes e/ou métodos utilizados
Arquitetural	MVC	Model – UsuarioEntity, UsuarioDAO, UsuarioService View – UsuarioActivity Controller – UsuarioController

QUADRO 4 - Apresentação dos design patterns utilizados

6.5.2 Convenções para codificação

Convenções são usadas para criar uma interface consistente para o código, e ajudam na manutenabilidade e legibilidade do código. Algumas convenções para codificação estão sendo utilizadas no desenvolvimento do projeto:

a) O recuo de endentação do código deve ser de quatro espaços.
b) As variáveis devem ser declaradas uma por linha.
c) Nomes de pacotes, classes, interfaces, métodos, variáveis e constantes devem ser de fácil entendimento.
d) Nomes de pacotes, classes e interfaces devem ser substantivos, com o padrão de escrita c*amelcase* com a primeira letra maiúscula.
e) Nomes de métodos e variáveis devem seguir o padrão de escrita *camelcase* com a primeira letra minúscula.
f) Nomes de métodos devem ser sempre verbos.

g) Nomes de constantes devem seguir o padrão *uppercase*, com todas as letras maiúsculas com o separador "_" entre as palavras.

6.6 ANÁLISE DE COMPLEXIDADE

Um bom desenvolvedor deve prever o comportamento do algoritmo para obter eficiência, pois um algoritmo pode ser eficiente com poucos dados de entrada e ineficiente com muitos dados, por exemplo, algoritmos de ordem de complexidade exponencial, em que o aumento de N aumenta exponencialmente a quantidade de instruções realizadas.

As avaliações dos algoritmos podem ser empíricas – dependência do computador, compilador, linguagem, etc. – ou podem ser teóricas – fórmula matemática em função do tamanho da entrada.

No QUADRO 5 é apresentada a ordem e o fator de complexidade em que são classificados os algoritmos.

DESCRIÇÃO	**ORDEM**	**FUNÇÃO**
Constante	1	$g(n) = 10$
Logarítmica	log n	$g(n) = 2 \log n$
Log quadrático	Log^2 n	$g(n) = 2 \log^2 n$
Linear	n	$f(n) = 10 . n$
n Log n	n log n	$f(n) = n \log n + \log n$
Quadrática	n^2	$f(n) = n^2 + 2$
Cúbica	n^3	$f(n) = n^3 + 5$
Exponencial	2^n	$g(n) = 10 . 2^n$

QUADRO 5 - Classes e ordens de complexidade
FONTE: VILLAS et al. (1993)

Baseado nos conceitos apresentados foi realizada a análise de complexidade de 3 algorítmos da aplicação cliente, que são demonstradas a seguir.

O primeiro algoritmo analisado é o método que atualiza o tempo de revisão de um cartão, como pode ser visto na FIGURA 22 a seguir.

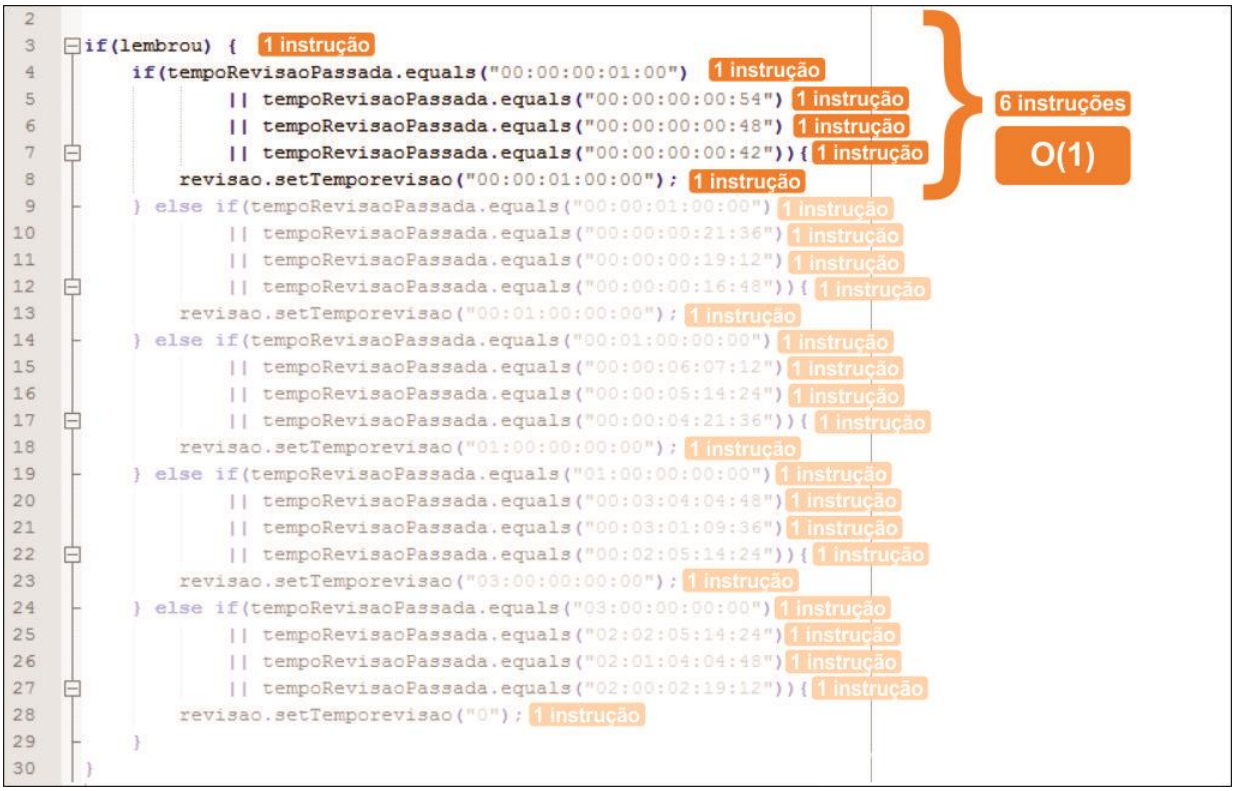

FIGURA 22 - Análise de complexidade de algoritmo, parte 1

O método possui as condições “Lembrou” ou “Não Lembrou” dos cartões estudados e ambas possuem ordem de complexidade constante (O(1)), significando que o algoritmo é de baixa complexidade.

Na FIGURA 23 a seguir é apresentada a segunda parte do algoritmo, que é a condição “Senão”, caso o usuário não lembrou a resposta do cartão revisado. Em qualquer condição da segunda parte a complexidade será menor do que na primeira parte.

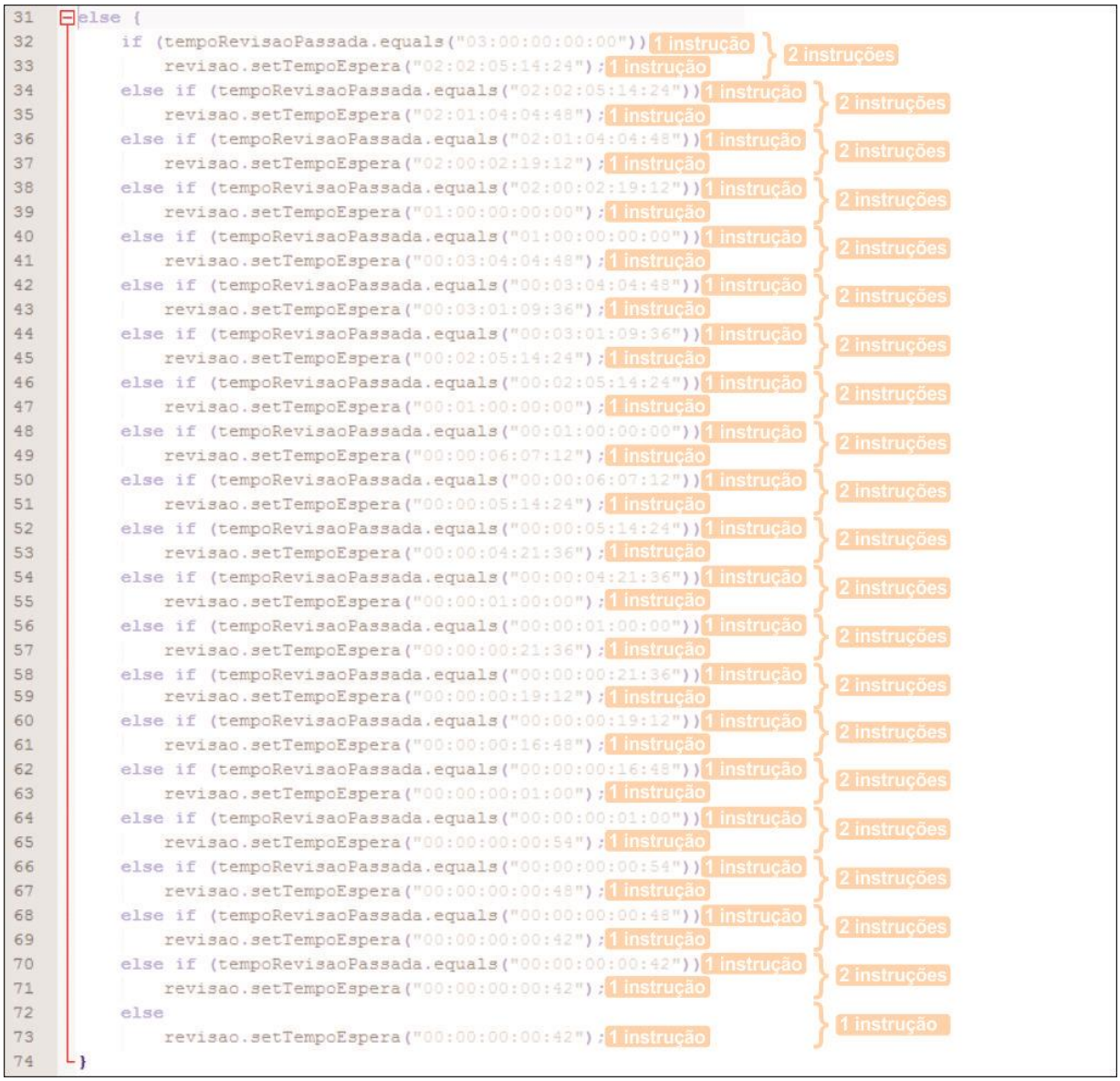

FIGURA 23 - Análise de complexidade de algoritmo, parte 2

O segundo algoritmo analisado é um método responsável por fazer a comunicação com a aplicação servidora, chamado “doInBackgorund”. A FIGURA 24 apresenta a primeira parte de três do método “doInBackgorund”.

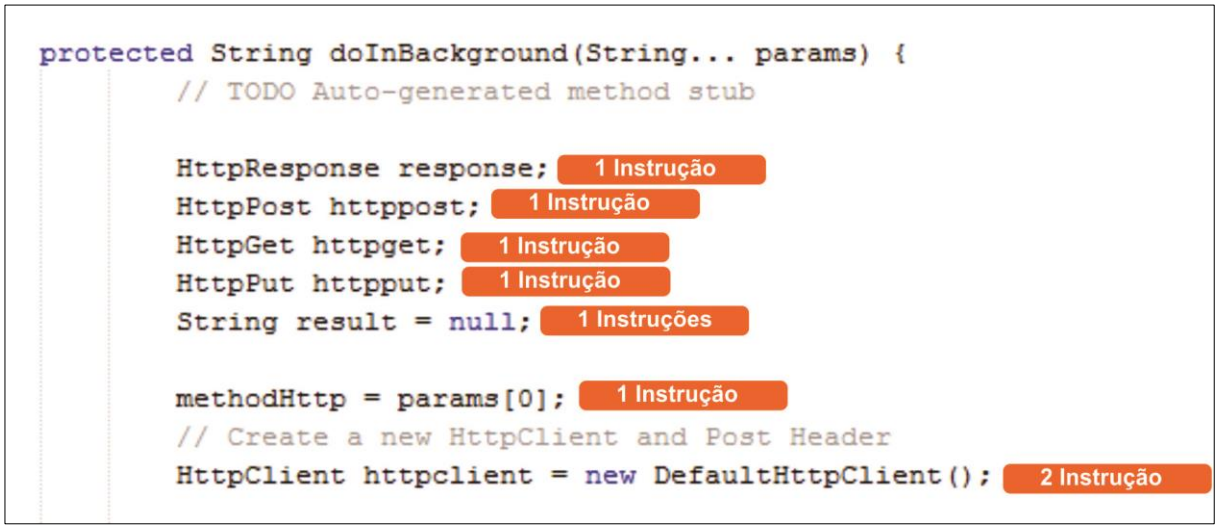

FIGURA 24 - Algoritmo responsável pela comunicação com a aplicação servidora, parte 1

A FIGURA 25 apresenta a segunda parte do método “doInBackground”.

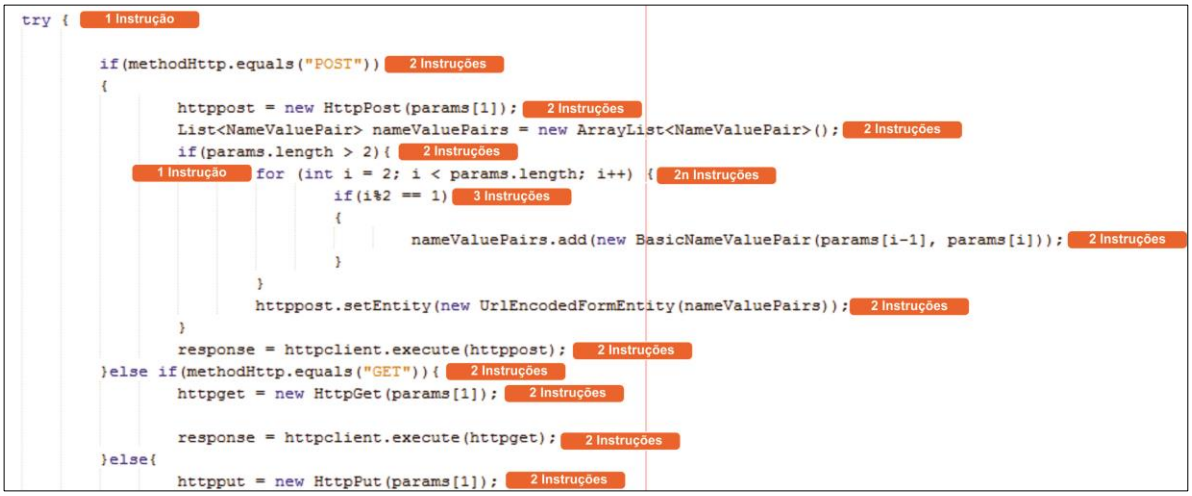

FIGURA 25 - Algoritmo responsável pela comunicação com a aplicação servidora, parte 2

A FIGURA 26 apresenta a terceira parte do método “doInBackground”.

```
            List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();  2 Instruções
            if(params.length > 2){  2 Instruções
        1 Instrução  for (int i = 2; i < params.length; i++) {  2n Instruções
                        if(i%2 == 1)  3 Instruções
                        {
                            nameValuePairs.add(new BasicNameValuePair(params[i-1], params[i]));  2 Instruções
                        }
                    }
                    httpput.setEntity(new UrlEncodedFormEntity(nameValuePairs));  2 Instruções
            }
            response = httpclient.execute(httpput);  2 Instruções
        }
        HttpEntity entity = response.getEntity();  2 Instruções

        if (entity != null) {  2 Instruções
            if (response.getStatusLine().getStatusCode() == 200) {  4 Instruções
                InputStream instream = entity.getContent();  2 Instruções
                result = convertStreamToString(instream);  2 Instruções
                instream.close();  1 Instrução
            }
        }
} catch (ClientProtocolException e) {  1 Instrução
        // TODO Auto-generated catch block
        e.printStackTrace();  1 Instrução
} catch (IOException e) {  1 Instrução
        // TODO Auto-generated catch block
        e.printStackTrace();  1 Instrução
}
return result;  1 Instrução
```

FIGURA 26 - Algoritmo responsável pela comunicação com a aplicação servidora, parte 3

O fator de complexidade do método “doInBackground” é de ordem linear (N), representado pela função “O(n) = 7n + 37” que é o caminho de pior caso (mais complexo).

O terceiro algoritmo analisado é um método responsável por fazer o controle da revisão de cartões, chamado “Registrar”. A FIGURA 27 apresenta a primeira parte de três do método “Registrar”.

```
private void Registrar(boolean lembrou)
        {
            int idCartao = listaCarta.get(IndiceCartao).getId();  3 Instruções

            Historico historico = new Historico();  2 Instrução
            historico.setLembrou(lembrou);  1 Instrução
            historico.setUsuarioFk(usuarioID);  1 Instrução
            historico.setCartaoFk(idCartao);  1 Instrução
            historico.setDataHora(new Date());  2 Instruçao
            listaHistorico.add(historico);  1 Instrução

            Date d = Calendar.getInstance().getTime();  3 Instruções
            SimpleDateFormat dataAtual = new SimpleDateFormat("yyyy-MM-dd HH:mm");  2 Instrução
            String dataCorrente = dataAtual.format(d);  2 Instruções
            String lembranca = lembrou ? "S" : "N";  2 Instruções

            Lembranca revisao = new Lembranca(0, "", "", "", "", "");  2 Instrução

            revisao.setBaralho(String.valueOf(idQuestionario));  2 Instrução
            revisao.setCartao(String.valueOf(idCartao));  2 Instrução
            revisao.setLembrou(lembranca);  1 Instrução
            revisao.setTempoEspera("00:00:00:01:00");  1 Instrução
            revisao.setDataRevisado(dataCorrente);  1 Instrução
```

FIGURA 27 - Algoritmo de controle de revisão de cartões, parte 1

A FIGURA 28 apresenta a segunda parte do método “Registrar”.

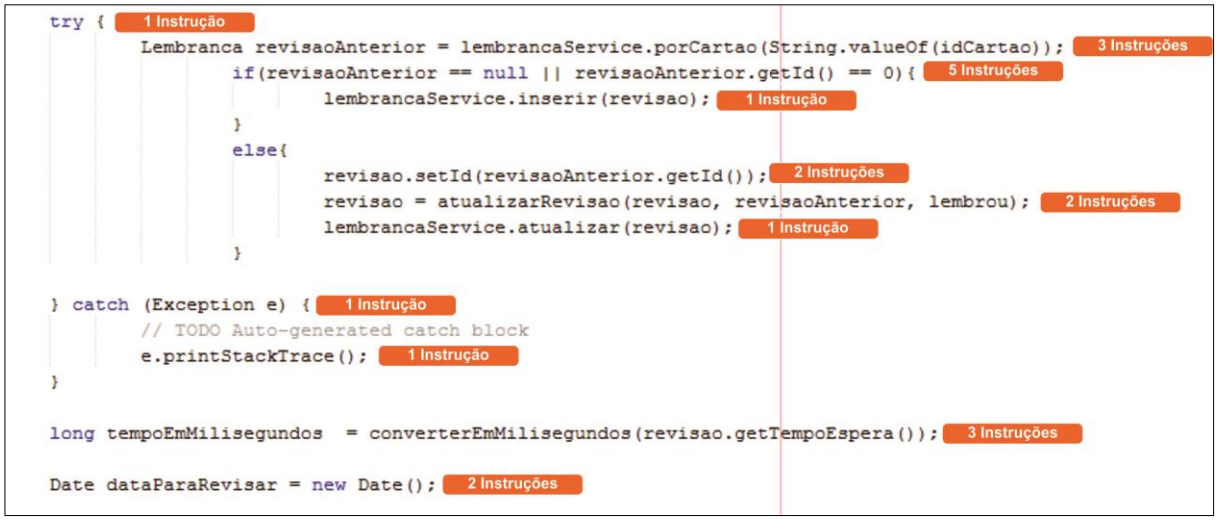

FIGURA 28 - Algoritmo de controle de revisão de cartões, parte 2

A FIGURA 29 apresenta a terceira parte do método “Registrar”.

FIGURA 29 - Algoritmo de controle de revisão de cartões, parte 3

O fator de complexidade do método “Registrar” é o de ordem Constante (1), representado pela função “O(n) = 64” que possui baixa complexidade.

6.7 PROJETO DE SISTEMAS DISTRIBUÍDOS

Nesta seção são apresentados os procedimentos para tratamento dos desafios do projeto e as tecnologias e arquiteturas de distribuição.

6.7.1 Procedimentos para tratamento dos desafios

Nesta seção são apresentados os desafios de sistemas distribuídos a serem enfrentados pelo projeto Aprendiz: heterogeneidade, abertura, segurança, manuseio de falhas, escalabilidade, concorrência e transparência. Também são descritos os procedimentos a serem adotados para tratar cada desafio.

6.7.1.1 Heterogeneidade

Este desafio se refere à capacidade de integração entre diferentes sistemas operacionais, redes, linguagens de programação, implementações de diferentes desenvolvedores, dentre outros aspectos do sistema.

> A *internet* permite aos usuários acessarem serviços e executarem aplicativos por meio de um conjunto heterogêneo de computadores e redes. A heterogeneidade (isto é, variedade e diferença) se aplica aos seguintes aspectos: redes, *hardware* de computador, sistemas operacionais, linguagens de programação, implementações de diferentes desenvolvedores (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 28).

O sistema está projetado para ser executado nos sistemas operacionais *Android* (aplicação cliente) e *Windows* (aplicação sevidora), e será codificado nas linguagens Java e C *Sharp* (C#), sendo Java a linguagem padrão utilizada para aplicações *Android*, e C# a mais utilizada e conhecida pelos integrantes do grupo para o propósito da aplicação servidora. Com isso surge a necessidade de integração entre estas diferentes linguagens e sistemas operacionais. Para vencer este desafio, será feito o uso do protocolo HTTP para realizar a comunicação entre as partes.

O sistema também será executado em diferentes arquiteturas de processadores, como *Acorn RISC Machine* (ARM), *SnapDragon, Intel, Motorola*, etc. Como os sistemas *Android* e *Windows* abstraem as camadas de acesso ao hardware, nada precisará ser feito no sistema pra vencer este desafio.

6.7.1.2 Escalabilidade

O desafio de escalabilidade é a capacidade do sistema continuar funcionando de forma eficiente com o grande aumento de quantidade de solicitações aos serviços do sistema. É também a facilidade de ser configurado ou expandido no futuro, para suprir novas demandas de solicitações.

“Um sistema é descrito como escalável se permanece eficiente quando há um aumento significativo no número de recursos e no número de usuários.” (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 31).

Como o sistema é experimental, o número de usuários esperado é de poucas dezenas, e com isso a sobrecarga de processamento no sistema não é prevista.

Foi feita análise de complexidade de alguns algoritmos, e acredita-se que os mesmos são de complexidade baixa. Quanto às sentenças SQL, serão otimizadas, de forma que as transações sejam executadas com uma quantidade mínima de solicitações SQL possível, para que seja amenizada a carga de processamento na base de dados.

6.7.1.3 Abertura

O desafio de abertura se refere à padronização, documentação adequada e distribuição dos artefatos necessários para fazer a utilização dos serviços do sistema, com a finalidade de que desenvolvedores de software terceiros possam acessar e entender a interface de comunicação e utilizar os serviços.

> A característica de sistema aberto é obtida a partir do momento em que a especificação e a documentação das principais interfaces de software dos componentes de um sistema estão disponíveis para os desenvolvedores de software. Em uma palavra, as principais interfaces são publicadas (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 29).

Neste projeto não existirá abertura de serviços à aplicações terceiras.

6.7.1.4 Segurança

O desafio de segurança visa manter o bom funcionamento do aplicativo, e é dividido em três categorias: confidencialidade, integridade e disponibilidade.

> Muitos recursos de informação que se tornam disponíveis e são mantidos em sistemas distribuídos têm um alto valor intrínseco para seus usuários. Portanto, sua segurança é de considerável importância. A segurança de recursos de informação tem três componentes: confidencialidade (proteção contra exposição para pessoas não autorizadas), integridade (proteção contra alteração ou dano) e disponibilidade (proteção contra interferência com os meios de acesso aos recursos) (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 30).

São necessários mecanismos que aumentem a confidencialidade das informações na aplicação servidora, para isso existirá o processo de cadastro de usuário e *login/logout*.

Para manter a integridade da base de dados, serão utilizadas funções de "*commit*" e "*rollback*" - utilizadas para concretizar transações na base de dados apenas quando todas as etapas são executadas com sucesso.

Considerando que a maior parte das funcionalidades do sistema dependem do correto funcionamento da aplicação servidora, existe a necessidade de alta disponibilidade da mesma. Portanto é esperado que a aplicação servidora permaneça disponível por 23 horas e 30 minutos por dia (~98% do tempo). Porém o impacto de os servidores ficarem indisponíveis por pequenos períodos de tempo será pequeno, considerando o propósito experimental da aplicação.

6.7.1.5 Manuseio de falhas

O desafio de manuseio de falhas é a capacidade das partes do sistema reconhecer as falhas e trata-las de forma que não comprometa o correto funcionamento do sistema.

"Às vezes, os sistemas de computador falham. Quando ocorrem falhas no hardware ou no software, os programas podem produzir resultados incorretos ou podem parar antes de terem concluído a computação pretendida." (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 32).

Falhas são previstas na troca de mensagens entre as aplicações cliente e servidor. Para tratar esta falha, serão utilizadas as técnicas de *timeout* e repetição de solicitação, e caso a falha persista na terceira tentativa de conexão, a tarefa deve ser abandonada e uma mensagem deve ser apresentada para o usuário, indicando a falta de conexão com a aplicação servidora.

6.7.1.6 Concorrência

Concorrência é o desafio de compartilhar simultaneamente os recursos de hardware, software e dados por diferentes partes do sistema, que possuem diferentes velocidades de processamento, localização topológica na rede, relógios, etc.

> [...] qualquer objeto que represente um recurso compartilhado em um sistema distribuído deve ser responsável por garantir que ele opere corretamente em um ambiente concorrente. Isso se aplica não apenas aos servidores, mas também aos objetos nos aplicativo. Portanto todo programador que implemente um objeto que não foi destinado para uso em um sistema distribuído, deve fazer o que for necessário para garantir que em um ambiente concorrente ele não assuma resultados inconsistentes (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 34).

É previsto a implementação de *threads* para comunicação com a base de dados, para que a comunicação seja assíncrona, utilizando de forma mais eficiente os recursos de hardware do servidor. O desafio de utilização "simultânea" dos recursos de hardware e software será vencido pelo controle de processos e *threads* dos sistemas operacionais Android e Windows, e pelo servidor web IIS.

O número de acessos "simultâneos" suportados pela base de dados será limitado em 50 para o propósito experimental da aplicação.

6.7.1.7 Transparência

O desafio de transparência é a capacidade das partes do sistema serem separadas em camadas com responsabilidades específicas, fazendo com que o funcionamento das camadas mais baixas (Exemplos: acesso aos servidores em redundância, tratamento de falhas na rede, controle de concorrência de solicitações, *etc*.) sejam transparentes para o desenvolvimento das camadas mais altas (Exemplo: solicitação de serviços da aplicação servidora).

> A transparência é definida como sendo a ocultação, para um usuário final ou para um programador de aplicativos, da separação dos componentes em um sistema distribuído de modo que o sistema seja percebido como um todo, em vez de uma coleção de componentes independentes. As implicações da transparência têm grande influência sobre o projeto do software de sistema (COULOURIS; DOLLIMORE; KINDBERG, 2001, p. 34).

Está projetado para que o sistema possua camadas separadas para os diferentes níveis de responsabilidades (o diagrama de pacotes podem ser consultados na seção 6.1.1), tornando transparente para os desenvolvedores o funcionamento das camadas mais baixas.

Como será utilizado o protocolo HTTP para a comunicação entre as aplicações do sistema, não será necessário definir/conhecer o endereço IP e porta da máquina em que estará sendo executada a aplicação servidora, ou seja, o endereço IP e porta em que a aplicação servidora estiver sendo executada é transparente para a aplicação cliente. E caso a aplicação servidora seja alterada para outra máquina na rede *internet*, não será necessário realizar alterações no endereço IP ou porta na aplicação cliente, já que o registro de um domínio os tornam transparentes.

6.7.2 Tecnologias e arquiteturas de distribuição

Para realizar a comunicação entre as partes do sistema, é previsto a utilização do protocolo HTTP, onde será possível a aplicação cliente solicitar os serviços da aplicação servidora através do domínio da máquina na rede *internet*.

No esquema do sistema distribuído, mostrado na FIGURA 30 a seguir, pode ser visualizado o *host* servidor, onde será executada a aplicação servidora, representado por H1, e o *host* cliente, onde será executada a aplicação cliente, representado por H2. No *host* H1 são executados os processos: “Aprendiz.dll” sendo a aplicação servidora do sistema, “.NET” sendo o interpretador da aplicação *web*, “IIS” sendo o servidor web, “PostgreSQL” sendo o sistema gerenciador de banco de dados. Há o arquivo “Aprendiz.tar” que é a base de dados. No *host* H2 são executados os processos: “Aprendiz.apk” sendo o aplicativo Aprendiz e “Dalvik” sendo o interpretador *Java Virtual Machine* (JVM).

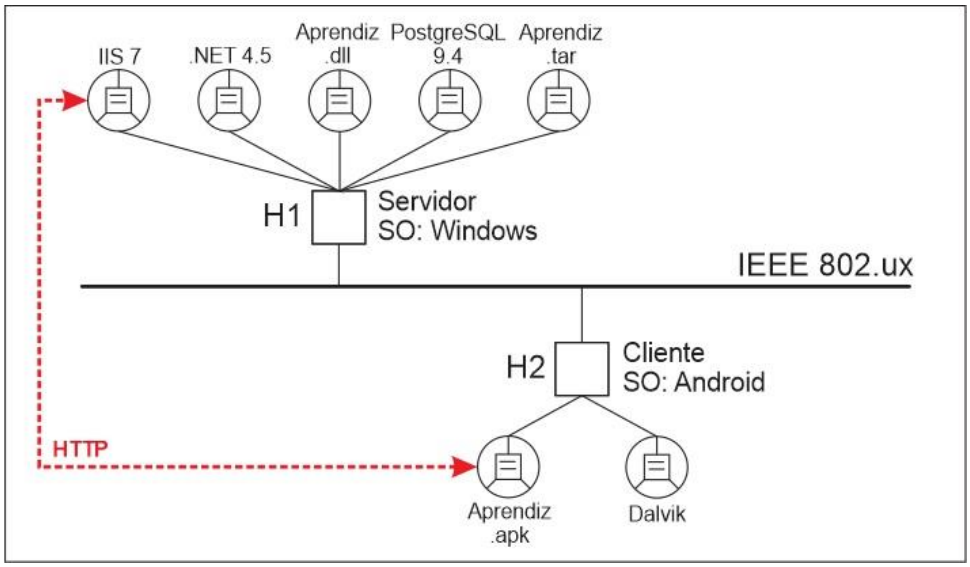

FIGURA 30 - Diagrama de sistema distribuído do projeto

6.8 PROJETO DA INTERAÇÃO HUMANO-COMPUTADOR

6.8.1 Perfis de usuários

Para a verificação do que os perfis dos futuros usuários pensam sobre o sistema, foram feitas três entrevista, questionando os entrevistados sobre a experiência com sistemas similares ao projeto Aprendiz, se usariam o sistema no futuro, com que frequência usariam, e se possuem alguma sugestão de modificação do sistema. As entrevistas podem ser visualizadas nos quadros a seguir.

Nome: Mauricio Ramalho de Morais
Idade: 30 anos
Sexo: Masculino
Profissão/Estudante: Bacharelando em Engenharia de Produção
Já utilizou algum sistema de *flashcards*? R: “Nunca utilizei e não conheço”.
Você usaria o aplicativo Aprendiz (*sistema de flashcards*)? Se sim, com que frequência? R: “Usaria, porque ajudaria melhorar a forma de estudo e em qualquer lugar seria possível estudar sem carregar papel, livros e caneta, já que o material estaria disponível no *smartphone*”.

“A frequência seria diária, principalmente, em semana de provas”.
Baseado na explicação sobre o que é o projeto e os recursos oferecidos pela aplicação, você mudaria algo (sugestão)? R: “Utilizar sistema de voz para ouvir e responder (com fone ou sem)”.

QUADRO 6 - Perfil 1 de usuário do sistema

Nome: Michel Liberato de Sousa
Idade: 34 anos
Sexo: Masculino
Profissão: Professor/Instrutor
Já utilizou alguma vez um sistema de *flashcards*? R: “Sim, o aplicativo *Flashcards* Liderança, mas muito pouco, pois ainda não estou bem familiarizado com um aplicativo que baixei no *smartphone*”.
Você usaria o aplicativo Aprendiz (sistema de *flashcards*)? Se sim, com que frequência? R: “Sim. Com uma frequência necessária para aprender o desejado”.
Baseado na explicação sobre o que é projeto e os recursos oferecidos pela aplicação, você mudaria algo (sugestão)? R: “O *flashcard* que eu estou tentando usar aqui, eu mudaria algumas coisas, eu ainda não estou bem familiarizado como te disse, mas não sei se tem como. Se fosse possível, eu o colocaria fundido ao *Whatsapp*, como se fosse uma mensagem com opções de múltipla escolha, por exemplo. Acho que teria mais atenção do usuário, mas precisaria saber se é possível fazer essa função”.
Você como professor, testaria o aplicativo com seus alunos (as)? R: “Quando tiver concluído faremos um teste sim, podemos usar inicialmente na aula de legislação e posteriormente nas aulas de direção veicular”.

QUADRO 7 - Perfil 2 de usuário do sistema

Nome: João Pedro de Jesus e Pinto
Idade: 22 anos

Sexo: Masculino
Profissão/Estudante: Estagiário/Bacharelando em Sistemas de Informação
Já conhece/já utilizou alguma vez a técnica de repetição espaçada? R: Conheço sim, com o aplicativo Duolingo. Obs.: O entrevistado relatou que acredita que o Kumon utiliza repetição espaçada.
Já utilizou alguma vez um sistema de *flashcards*? R: "Não".
Você usaria o aplicativo Aprendiz (*sistema de flashcards*)? Se sim, com que frequência? R: "Usaria, com frequência em épocas de provas e necessidade de aprender grande quantidade de informações".
Baseado na explicação sobre o que é projeto e os recursos oferecidos pela aplicação, você mudaria algo (sugestão)? R: "Colocaria uma forma de validar as respostas, pois talvez o usuário minta nas respostas por não haver uma validação das mesmas".

QUADRO 8 - Perfil 3 de usuário do sistema

6.8.2 Aspectos visuais da interface de usuário

Nesta seção apresentamos um pouco sobre os aspectos visuais da interface de usuário.

O ideal é que essa interface seja consistente, que mostre apenas informações importantes ao contexto do usuário, o uso de nomes de fácil entendimento é fundamental, os ícones escolhidos são ícones padrões que fazem algum tipo de analogia ao mundo real.

As cores escolhidas para serem predominantes na aplicação são as cores laranja, preto e branco, como pode ser visualizado na FIGURA 31.

FIGURA 31 - Tela de cadastro de questionário

6.8.3 Heurísticas de usabilidade

Nesta seção, são exploradas as práticas de usabilidade, visando proporcionar um alto grau de usabilidade de forma que atenda e satisfaça as necessidades do usuário.

6.8.3.1 Visibilidade do estado do sistema

O usuário sempre precisa ser informado do que está acontecendo durante as operações, ou seja, o usuário deve saber onde está, o que está fazendo, para onde pode ir e o que está ocorrendo durante a operação que está realizando, para isso é necessário um *feedback*.

A visibilidade do estado do sistema, também conhecida como *feedback* instantâneo, visa permitir que o usuário saiba em que posição atual no sistema ele se encontra. O sistema manterá o usuário informado através de mensagens, como erros operacionais, informações complementares, aprovações de operações, dentre outras, como pode ser visualizado na FIGURA 32 a seguir.

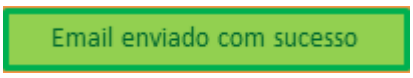

FIGURA 32 - Mensagem de uma operação realizada com sucesso

6.8.3.2 Relacionamento entre a interface do sistema e o mundo real

A heurística de relacionamento entre a interface do sistema e o mundo real visa evitar palavras no sistema que não faz sentido para o usuário, como termos técnicos comuns à equipe de desenvolvimento.

Criando uma interface intuitiva e sem termos técnicos, pretende-se que o aplicativo Aprendiz não sobrecarregue a memória do usuário, a fim de que este possa fazer um bom uso do aplicativo, podendo realizar suas tarefas e não ser induzido a erros. Alguns exemplos desta heurística no aplicativo são mensagens negativas destacadas em vermelho, e mensagens de operações concluídas com sucesso destacadas em verde, como pode ser visualizada na FIGURA 32.

6.8.3.3 Liberdade de controle fácil para o usuário

Esta heurística propõe facilitar as operações e manter o usuário no controle do sistema.

Na aplicação cliente o usuário poderá realizar operações e cancelá-las de forma que não contrariem as regras de negócio do sistema. Alguns exemplos da utilização dessa heurística no aplicativo podem ser visualizadas na FIGURA 33, que são:

a) possiblidade de criar, atualizar e remover questionário, cartão e sala de estudo; e

b) possibilidade de cancelar qualquer operações de criação, atualização e remoção, como questionário, cartão e sala de estudo.

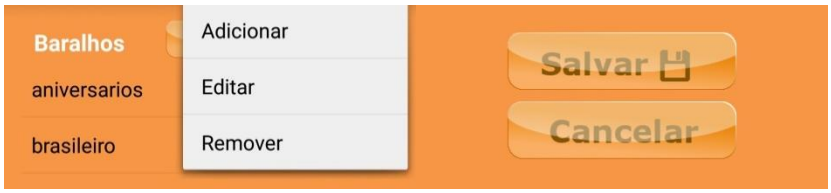

FIGURA 33 - Controles de realização e cancelamento de tarefas

6.8.3.4 Consistência e padrões

O usuário ao fazer uso de um sistema que não possui consistência e padrões bem definidos aplicados à sua interface, poderá se confundir e ter dificuldades durante suas operações. Se um usuário tem mais confusão do que informação útil, poderá ser um grande problema, e o sucesso de qualquer sistema se dá por sua eficiência em atender as expectativas do usuário.

Visando atender essa heurística foram definidos alguns padrões a serem utilizados em todas as telas do aplicativo, como fontes de textos, menus, cores, posicionamento de componentes e sequências de ações em situações similares, como pode ser visualizado na FIGURA 34 a seguir. Estes padrões são aplicados ao sistema a fim de reduzir a carga de memória do usuário, possibilitando que este possa fazer um bom uso do aplicativo, satisfazendo suas necessidades.

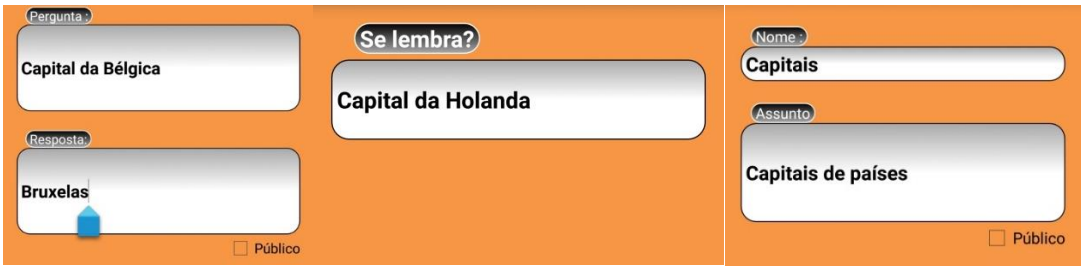

FIGURA 34 - Telas padronizadas

6.8.3.5 Prevenções de erros

Serão criados mecanismos que possam prevenir erros básicos do usuário, como a exibição de janelas de confirmação em caso de exclusão de questionários, cartões e salas de estudo, para certificar de que o usuário realmente deseja realizar as operações solicitadas.

6.8.3.6 Reconhecimento em vez de memorização

Por ser um aplicativo que auxilia na memorização de informações na memória de longo prazo, seria um problema para o usuário caso ele precisasse memorizar longos caminhos de navegação pelo aplicativo, ou algo que pudesse sobrecarregar sua memória. Sendo assim, existe a necessidade de garantir que o sistema demande pouca memorização do usuário, com uma interface de usuário consistente e de fácil navegabilidade, com uma linguagem clara e que indique ao usuário sua posição atual no sistema.

6.8.3.7 Flexibilidade e eficiência de uso

O aplicativo não possui teclas/comandos de atalho – com exceção das teclas de atalho do Android – por oferecer poucas funcionalidades e possuir uma interação simples. Pretende-se que tanto usuários leigos quanto avançados se adaptem facilmente às funcionalidades e recursos do aplicativo.

6.8.3.8 Estética e design minimalista

Apresentar informações úteis é um princípio básico de um sistema. Para tornar eficiente a comunicação do sistema para com o usuário, pretende-se que a comunicação seja simples, informando apenas o necessário para o usuário, como pode ser visualizado na FIGURA 34.

6.8.3.9 Suporte a usuários para reconhecer, diagnosticar e sanar erros

Mais do que um *design* cuidadoso, o aplicativo não permitirá que erros operacionais procedam, pois respostas imediatas serão retornadas ao usuário, diagnosticando e sugerindo alternativas para a correção dos erros. A linguagem utilizada será simples e sem códigos, pela diversidade existente de usuários que pode utilizar o aplicativo.

6.8.3.10 Ajuda e documentação

Uma das ideias do projeto de interface é que o aplicativo possua uma interface tão simples que suas operações sejam intuitivas, não sendo necessário o apoio de materiais de ajuda. Mas caso o usuário necessitar de ajuda para utilizar o aplicativo, poderá consultar o manual do usuário – acessível no Apêndice I.

7 PLANO DE TESTES

Neste capítulo é apresentado o plano de testes do sistema Aprendiz, em que está contido a finalidade de realização dos testes, o escopo, documentos relevantes a serem consultados, ambiente para a realização dos testes, descrição dos casos de teste, itens que serão ou não testados e a realização dos mesmos. Por fim é apresentado o histórico de realização dos testes e os resultados obtidos.

7.1 FINALIDADE

A finalidade deste plano de testes é garantir que os requisitos funcionais essenciais e não funcionais estão sendo atendidos, e para isto serão realizados testes de sistema.

7.2 ESCOPO

Nesta seção são apresentados os documentos relevantes e o ambiente para a realização dos testes.

7.2.1 Referências a documentos relevantes

No QUADRO 9 a seguir são apresentados os documentos relevantes para a execução dos testes.

Tipo do Material	Referência
Documento de requisitos funcionais e não funcionais	Seção 5.1 deste documento

QUADRO 9 - Documentos relevantes para testes

7.2.2 Ambiente para a realização dos testes

No QUADRO 10 a seguir é apresentado os equipamentos a serem utilizados para a realização dos testes.

Equipamento	**Marca/Modelo/Configuração**	**Finalidade**
Smartphone	Lenovo A6020L36, 2GB RAM, 16GB ROM, 8 Cores 1.5 GHz. Android 5.1.	Executar a aplicação cliente para a realização dos testes.
Computador	2GB RAM, 500GB ROM, Itel	Executar a aplicação servidora

	Atom 1.6 GHz	durante a realização dos testes.

QUADRO 10 - Equipamentos a serem utilizados para a realização dos testes

7.3 ESPECIFICAÇÃO DOS CASOS DE TESTES

Nesta seção são apresentados os itens que serão testados e os que não serão testados, a rastreabilidade entre os casos de teste e os requisitos do sistema e a descrição dos casos de teste.

7.3.1 Itens a testar

Identificação do item	Descrição
001	Cadastro de usuário
002	Recuperação de senha
003	Cadastro de questionário
004	Cadastro de cartão
005	Revisão de questionário

QUADRO 11 - Identificação dos itens a serem testados

7.3.2 Itens que não serão testados

Serão realizados testes apenas dos requisitos essenciais do sistema. Não serão realizados testes de instalação e configuração do sistema.

7.3.3 Rastreabilidade entre requisitos e casos de testes

O QUADRO 12 a seguir apresenta o relacionamento entre os casos de teste e os requisitos.

Identificação do requisito	Caso(s) de teste(s) aplicável (eis)
RF01 – Manter usuário	CT01 e CT02
RF02 – Solicitar recuperação de senha	CT03 e CT04
RF04 – Manter questionário	CT05 e CT06
RF05 – Manter cartão	CT07 e CT08

RF06 – Realizar estudo/revisão de questionário	CT09
RF19 – Realizar comunicação com aplicação servidora	CT01, CT02, CT03, CT04, CT05, CT06, CT07, CT08 e CT09

QUADRO 12 - Requisitos e casos de testes

7.3.4 Descrição dos casos de testes

Na descrição do caso de teste é apresentado o fluxo de execução, entradas, saídas esperadas e dependências para a realização do teste. Os casos de teste a serem realizados no sistema Aprendiz podem ser consultados no Apêndice G.

7.4 RESULTADOS DOS TESTES

Nesta seção é apresentado o histórico de realização dos casos de teste e o resultado obtidos após a realização de todos os casos de teste.

7.4.1 Histórico de realização

O histórico de realização dos casos de testes do sistema Aprendiz pode ser consultado no Apêndice H.

7.4.2 Resultados

Foram realizados testes de sistema relacionados aos requisitos essenciais do sistema Aprendiz e os resultados obtidos foram satisfatórios. Algumas falhas foram encontradas e reportadas para os responsáveis pelo desenvolvimento. Após a correção das falhas os testes foram refeitos e nenhuma falha foi encontrada.

Pode-se concluir que a fase de testes cumpriu sua função de realizar os testes e encontrar falhas, possibilitando a correção e melhoria do sistema.

8 PLANO DE IMPLANTAÇÃO

A implantação é a última fase no ciclo de desenvolvimento de um software, quando ele é disponibilizado aos usuários. Após a liberação da fase de testes, o software é inserido no ambiente onde será utilizado, passa por uma fase de treinamento e por acompanhamento dos resultados obtidos para que se possam prover possíveis melhorias até se alcançar todos os objetivos do sistema.

Neste capítulo é descrita a metodologia de implantação utilizada, a matriz de responsabilidades, treinamentos previstos, cronograma de implantação, documentos de apoio à implantação e tipos de suporte técnico.

8.1 METODOLOGIA

Nesta seção é apresentada a metodologia utilizada na implantação do sistema e os responsáveis por cada tarefa.

8.1.1 Descrição da metodologia

A metodologia adotada tem como objetivo orientar os usuários sobre os requisitos necessários para utilização e forma de uso do sistema.

O objetivo do plano de implantação é assegurar a disponibilidade do aplicativo aos usuários e garantir que o aplicativo esteja atendendo aos objetivos pelo qual foi desenvolvido e disponibilizado ao usuário final.

8.1.2 Matriz de responsabilidades

No QUADRO 13 a seguir é exibida a matriz de responsabilidades que se refere as atividades e aos responsáveis por cada uma delas.

Atividades	Responsáveis
Planejamento	
Definição da equipe de implantação	Juliano Costa Silva
Levantamento de recursos necessários de hardware	Robson dos Santos
Levantamento de recursos necessários de software	Francisco de Faria

	Cardoso
Programação dos treinamentos	Marcelo Pereira Costa
Preparação dos testes de aceitação	Juliano Costa Silva
Execução	
Configuração de infraestrutura de hardware	Robson dos Santos
Configuração de infraestrutura de software	Francisco de Faria Cardoso
Instalação do produto	Próprio usuário faz o download e instala o aplicativo.
Treinamentos (tutoriais)	Marcelo Pereira Costa
Realização de testes de aceitação	Juliano Costa Silva
Avaliação	
Acompanhamento pós-implantação	Equipe
Reunião final da implantação	Equipe

QUADRO 13 - Papéis e responsabilidades na implantação

8.2 TREINAMENTOS PREVISTOS

O QUADRO 14 a seguir detalha os treinamentos utilizados para a capacitação dos usuários durante a fase de implantação do produto.

Treinamento	Conteúdo	Grupo de usuários
Como utilizar o aplicativo Aprendiz.	Tutorial mostrando como utilizar todas as funcionalidades do aplicativo.	Usuários em geral.

QUADRO 14 - Treinamentos previstos

8.3 CRONOGRAMA DE IMPLANTAÇÃO

No QUADRO 15 a seguir são apresentadas as tarefas previstas durante a implantação, a duração em horas e o período de realização.

Tarefas	Duração	Período
Preparação e configuração do ambiente da aplicação servidora.	2 horas.	A definir.
Realização de testes no aplicativo.	8 horas.	A definir.
Construção de um tutorial explicando as funcionalidades do aplicativo aos usuários.	2 horas.	A definir.
Disponibilização do aplicativo para download.	0,5 horas.	A definir.
Acompanhamento do *feedback* dos usuários.	Indeterminado.	Pós disponibilização do aplicativo para download.
Tempo estimado total:	Em torno de 13 horas.	

QUADRO 15 - Cronograma de atividades da implantação

8.4 DOCUMENTOS DE APOIO À IMPLANTAÇÃO

Os documentos oferecidos para apoiar o processo de implantação e posterior uso do produto são listados no QUADRO 16 a seguir.

Documento	Referência
Manual do usuário	Vide Apêndice I

QUADRO 16 - Documentos de apoio à implantação

8.5 TIPOS DE SUPORTE TÉCNICO

Por se tratar de um aplicativo para Android que será disponibilizado na *play store* (loja online mantida pela Google para distribuição de aplicações), o único suporte técnico oferecido será por atualizações que podem ou não serem disponibilizadas para reparar possíveis falhas que serão reportados pelos próprios usuários através do *feedback* que será acompanhado pela equipe através da própria *play store*.

9 CONCLUSÃO

A primeira fase foi de muitas pesquisas, pois era necessário adquirir conhecimento sobre o assunto, precisávamos entender como funciona o processo de memorização, e também sobre algumas metodologias de ensino, e assim fizemos. Através dessas pesquisas foi possível ampliar nosso conhecimento sobre o assunto para tentar alcançar os objetivos do projeto.

Após compreendermos melhor quais eram os problemas que iriamos solucionar com o desenvolvimento do aplicativo e também quais eram os nossos objetivos e o nosso público alvo, foi possível fazer o levantamento de todos os requisitos do aplicativo e classifica-los como essencial, importante e desejável.

A partir da segunda fase precisávamos definir quais tecnologias iriamos utilizar, então foi necessário novamente fazer algumas pesquisas para definir quais seriam as melhores opções para o projeto. Optamos por usar algumas tecnologias já utilizadas em aula e acrescentamos outras que seriam boas opções para nossa aplicação, como o uso da linguagem C# – que não faz parte do conteúdo aprendido em sala de aula, mas que já era conhecida por alguns membros da equipe, o que facilitou o desenvolvimento – e a escolha do sistema operacional Android para o desenvolvimento do aplicativo – que também seria novidade para alguns membros da equipe. Com isso houve um enriquecimento de conhecimento, já que aprendemos uns com os outros.

Com a aproximação da fase final o tempo começou a ser um problema para os membros da equipe, e precisávamos ter a aplicação funcionando, então, decidimos que iriamos implementar os requisitos essenciais e os importantes, os desejáveis seriam implementados caso houvesse tempo suficiente. Alcançamos os objetivos de contribuir com uma ferramenta de estudo que torne o estudo um pouco mais interativo e menos cansativo, tornando a memorização de conteúdos mais eficiente. A FIGURA 35 mostra uma tela que exibe uma pergunta ao usuário:

FIGURA 35 - Tela de estudo

A intenção é tornar o aplicativo ainda mais interativo futuramente, implementando novas funções que possibilitem que professores possam acompanhar o desempenho dos alunos nos estudos. O aplicativo ficará disponível para download de maneira gratuita para que os interessados possam usar e também nos dar um feedback para que com isso possamos melhorar seu funcionamento com a disponibilização de novas versões.

Com o término do projeto, ficamos satisfeitos com os resultados e, principalmente, com o entusiasmo de algumas pessoas que viram o aplicativo em funcionamento, acharam uma boa ideia e se mostraram interessadas em utilizá-lo como um método de estudo. Apesar de satisfeitos, sabemos que ainda podemos fazer muitas melhorias e ampliar a utilidade do aplicativo. Com isso tomamos como lição sempre estar atentos aos erros e às críticas para que possamos sempre melhorar, e que se quisermos algo precisamos realmente correr atrás, realizar pesquisas, buscar o conhecimento necessário, e muito provavelmente os resultados serão gratificantes no final.

REFERÊNCIAS

ANKI. **Exemplo de flashcard**. Disponível em: <http://anki.com.br/>. Acesso em: 8 de set. de 2016.

BOCK, Ana Mercês Bahia; FURTADO, Odair; TEIXEIRA, Maria de Lourdes Trassi. **Psicologias**. São Paulo: Saraiva, v. 13, 1999.

COULOURIS, George; DOLLIMORE, Jean; KIDBERG, Tim. **Sistemas Distribuídos: Conceitos e projetos,** 4 .ed. Porto Alegre: Bookman Companhia Ed, 2007. 28-34p.

DELL'ISOLA, Alberto. **Mentes brilhantes: curva do esquecimento**. 2008. Disponível em: <http://memorizacao.blogspot.com.br/2008/05/curva-do-esquecimento.html>.

DUOLINGO. **Palavras**. Disponível em: <https://www.duolingo.com/words>. Acesso em: 8 de set. de 2016.

EBBINGHAUS, Hermann. **Über das gedächtnis: untersuchungen zur experimentellen psychologie**. Books on Demand, 1885.

FREIRE, Paulo. **Pedagogia do oprimido**. 17. ed. Rio de Janeiro: Paz e Terra, 1987.

LOPES, Ana. **Dê um tempo para aprender mais rápido**. Disponível em: <https://www.youtube.com/watch?v=0_Aqp3KNgMI>. Acesso em: 2 de abr. de 2016.

PINTO, A.. **Memória, cognição e educação: Implicações mútuas.** In: B. Detry e F. Simas (Eds.), Educação, cognição e desenvolvimento: Textos de psicologia educacional para a formação de professores. Lisboa: Edinova, 2001. 17-54p. Disponível em: <http://www.fpce.up.pt/docentes/acpinto/artigos/16_memoria_e_educacao.pdf>.

PRESSLEY, M.; GOODCHILD, F.; FLEET, F.; ZAJCHOWSKI, R.; EVANS, E. D.. **The challenges of classroom strategy instruction**. The Elementary School Journal, 1989. 301–342p.

PROJECT MANAGEMENT INSTITUTE, INC. **Conhecimento em gerenciamento de projetos (Guia PMBOK)**. 2013.

PRUZAN, Todd. **As pessoas mais desajeitada da Europa: Ou, mal-humorado guia da Sra. Mortimer para o mundo vitoriana**. [S.l.]: Bloomsbury Publishing, 2008. 5p.

SILVA, Diogo Correia A.; CARNIELLO, Andreia; CARNIELLO, Adriana. **Flashcards virtuais – Técnica de repetição espaçada aplicada ao apoio na memorização do conteúdo estudado: flashcards**. 2006. Disponível em: <http://www.gestaouniversitaria.com.br/system/scientific_articles/files/000/000/067/original/Artigo_Flashcard_-_Gest%C3%A3o_Universit%C3%A1ria.pdf?1423144482>.

SILVA, Diogo Correia A.; CARNIELLO, Andreia; CARNIELLO, Adriana. **Utilização da técnica de repetição espaçada por meio de flashcards virtuais para o aumento da memorização na aprendizagem**. Sinergia - Revista Cientifica do Instituto Federal de São Paulo, v15, artigo 7, 2014. Disponível em: <http://www2.ifsp.edu.br/edu/prp/sinergia/complemento/sinergia_2014_n4/pdf_s/segmentos/artigo_07_v15_n4.pdf>.

VILLAS, Marcos Vianna et al. **Estruturas de dados: conceitos e técnicas de implementação**. Rio de Janeiro: Campus, 1993.

ZNP. **Flashcards e o Sistema Leitner**. 2015. Disponível em: <http://www.zinplez.com/2015/02/09/flashcards-e-o-sistema-leitner/>.

GLOSSÁRIO

A

Acorn RISC Machine*:* é uma arquitetura de processador de 32 bits usada principalmente em sistemas embarcados. São processadores que visam a simplificação das instruções, com o intuito de atingir a máxima eficiência por ciclo, podendo realizar tarefas menores com ciclos mais curtos, e uma maior ordenação das operações dentro do núcleo de processamento.

Ad-hoc*:* em Informática, uma rede ad-hoc é uma ligação temporária entre vários computadores e dispositivos utilizada para uma finalidade específica, são redes sem fio em que os computadores interligados comunicam diretamente entre si sem necessidade de um roteador.

Algoritmo: em informática, conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas.

Arcabouço do projeto: em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

Artefato: em desenvolvimento de software, é o produto de uma ou mais atividades dentro do contexto do desenvolvimento de um software ou sistema.

Atômico: Adj. Que é relativo ao átomo. No sentido figurado, é aquilo que tem efeitos rápidos e enérgicos.

B

Backup*:* em informática, é um termo inglês que tem o significado de cópia de segurança**.**

BlueStacks*:* é uma ferramenta que permite executar aplicativos criados para o sistema operacional Android no Windows e Mac.

C

C#: é uma linguagem de programação de alto nível e orientada a objetos criada pela Microsoft.

Código hash*:* é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo.

Commit*:* em ciência da computação e gerenciamento de dados, commit refere-se à ideia de fazer permanente um conjunto de mudanças experimentais.

Confidencialidade: é a propriedade da informação pela que não estará disponível ou divulgada a indivíduos, entidades ou processos sem autorização.

Consolidada: adj. Consistente, sólido; firme, seguro, estável.

D

Declínio: inclinação para plano inferior; declivamento.

Diagrama: representação gráfica de fatos, fenômenos etc.; gráfico, esquema.

Dispendioso: adj. que exige grande despesa; caro. Que consome muito.

dotNET: O DotNET é um Framework de desenvolvimento. Com ele você não precisa investir seu tempo para fazer determinadas tarefas, pois ele já tem um caminho para que esta tarefa seja feita, basta apenas você usar.

E

Eficácia: é a qualidade daquilo que cumpre com as metas planejadas.

Escalável: em informática, que tem condições para crescer de forma uniforme ou para suportar um aumento de carga.

Escopo: em gerenciamento de projetos, é a soma total de todos os produtos do projeto e seus requisitos ou características, e possui dois usos distintos: Escopo do Projeto e Escopo do Produto.

F

Feedback*:* Feedback é uma palavra inglesa que significa realimentar ou dar resposta a uma determinado pedido ou acontecimento.

G

Gradualmente: que se faz por graus; por etapas; Progressivo.

H

Hardware: em informática, é a parte física de um computador, é formado pelos componentes eletrônicos, como por exemplo, circuitos de fios e luz, placas, utensílios, correntes, e qualquer outro material em estado físico, que seja necessário para fazer com o que computador funcione.

Heurística*:* é um método ou processo criado com o objetivo de encontrar soluções para um problema. É um procedimento simplificador (embora não simplista) que, em face de questões difíceis, envolve a substituição destas por outras de resolução mais fácil a fim de encontrar respostas viáveis, ainda que imperfeitas.

Hierarquia: classificação, de graduação crescente ou decrescente, segundo uma escala de valor, de grandeza ou de importância.

Hipotética: adj. que se pauta em hipóteses ou suposições; suposto: argumento hipotético. Que possui hipótese; incerto.

Host*:* Em informática, host ou hospedeiro, é qualquer máquina ou computador conectado a uma rede, podendo oferecer informações, recursos, serviços e aplicações aos usuários ou outros nós na rede.

I

IDE: Ambiente de Desenvolvimento Integrado, um ambiente integrado para desenvolvimento de software.

Incremento: ato de crescer ou aumentar; desenvolvimento.

Integridade: em informática, indica que os mesmos não podem sofrer modificações não autorizadas.

Intrínseco: é um adjetivo masculino que significa íntimo, interno, inerente, constitutivo e classifica algo que está no interior**.**

J

Java: é uma linguagem de programação interpretada, orientada a objetos, desenvolvida na década de 90.

JVM Dalvik: é uma máquina virtual baseada em registradores, projetada e escrita por Dan Bornstein com contribuições de outros engenheiros do Google como parte da plataforma Android para telefones celulares.

K

Kumon: é uma instituição de ensino de disciplinas como matemática, português, japonês e inglês. Kumon significa auto-didata em japonês e é exatamente esse o objetivo do método Kumon, fazer com que o aluno aprenda por si próprio.

L

Lecionamento: ato de lecionar; ensinar algo a alguém.

Legibilidade: estado do que é legível; é uma qualidade que determina a facilidade de leitura de alguma coisa.

Leigo: que ou aquele que é estranho a ou que revela ignorância ou pouca familiaridade com determinado assunto, profissão etc.; desconhecedor, inexperiente.

M

Manutenibilidade: é uma característica inerente a um projeto de sistema ou produto, e se refere à facilidade, precisão, segurança e economia na execução de ações de manutenção nesse sistema ou produto.

Microsoft Project*:* é um software de gestão de projetos produzido pela Microsoft.

N

Nuvem: em informática, o conceito de computação em nuvem (em inglês, cloud computing) refere-se à utilização da memória e da capacidade de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet, seguindo o princípio da computação em grade.

O

Open source: é um termo em inglês que significa código aberto. Isso diz respeito ao código-fonte de um software, que pode ser adaptado para diferentes fins.

Organograma: em informática, gráfico que representa as operações de um programa e sua ordem interna no computador.

P

PgAdmin: é um software gráfico para administração do SGBD PostgreSQL disponível para Windows e UNIX.

Pontos de Casos de Uso: permitir que seja possível estimar o tamanho do sistema ainda na fase de levantamento de Casos de Uso, utilizando-se dos próprios documentos gerados nesta fase de análise como subsídio para o cálculo dimensional.

Pontos de Função: é uma técnica para a medição de projetos de desenvolvimento de software, visando a estabelecer uma medida de tamanho, em Pontos de Função (PF), considerando a funcionalidade implementada.

PostgreSQL: é um sistema de gerenciamento de banco de dados objeto-relacional.

Postman: é uma aplicação que permite realizar requisições HTTP a partir de uma interface simples e intuitiva, facilitando o teste e depuração de serviços REST.

Protocolo HTTP: É um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores, principalmente na World Wide Web (Internet).

Psicologia Experimental: é o comportamento observável, a fim de testar modelos e teorias matemáticas sobre diversos aspectos do mesmo: prestar atenção, perceber, recordar, aprender, decidir, reagir emocionalmente e interagir.

R

Redundância: em banco de dados, diz respeito à repetição não necessária dos dados nele contidos. É a melhor maneira de obter um sistema de alta disponibilidade.

Repositório: lugar onde se guarda, arquiva, coleciona alguma coisa.

Requisito: condição para se alcançar determinado fim.

RiouxSVN: é uma ferramenta de controle de versão utilizada por desenvolvedores de software.

Rollback: em informática, é uma palavra de usada para definir encerramento da transação, descartando (desfazendo) todas as alterações realizadas durante a transação.

S

Servidor: Em informática, é um software ou computador, com sistema de computação centralizada que fornece serviços a uma rede de computadores, chamada de cliente.

SGBD: é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados.

Sincronização: em informática, é o gerenciamento adequado de múltiplas linhas de execução ou processos concorrentes que acessam um mesmo recurso limitado ou uma porção de dados, situação conhecida como condição de corrida.

SnapDragon*:* Snapdragon é uma linha de processadores fabricada pela Qualcomm para smartphones, tablets e outros dispositivos móveis.

Software: conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.

Stakeholders*:* é uma pessoa ou grupo que possui participação, investimento ou ações e que possui interesse em uma determinada empresa ou negócio.

Suscetível: capaz ou passível de receber, de experimentar, de sofrer certas impressões ou modificações ou de adquirir determinadas qualidades.

T

Thread*:* Um pequeno programa que trabalha como um sub-sistema independente de um programa maior, executando alguma tarefa específica.

Timeout: em informática, é um evento que indica que um limite de tempo predeterminado esgotou-se sem que algum outro evento esperado ocorresse.

U

Usabilidade: é um termo usado para definir a facilidade com que as pessoas podem empregar uma ferramenta ou objeto a fim de realizar uma tarefa específica e importante.

V

Visual Paradigm*:* é uma ferramenta para desenvolvimento de aplicativos utilizando modelagem UML.

OBRAS CONSULTADAS

CARMO, João dos Santos. **Fundamentos psicológicos da educação**. Curitiba: InterSaeres, 2012.

DE SANTANA, Vagner Figuerêdo. **Especialização em projeto e desenvolvimento de sistemas**. 2012. Disponível em: <http://pt.slideshare.net/santanavagner/padroes-arquiteturais-de-sistemas>.

DUNLOSKY, John; RAWSON, A. Katherine; MARSH, J. Elizabeth; NATHAN, J. Mitchell; WILLINGHAM, T. Daniel. **Improving Students' Learning With Effective Learning Techniques: Promising Directions From Cognitive and Educational Psychology**. APS - Association of Psychological Science, 2013. Disponível em: <http://www.indiana.edu/~pcl/rgoldsto/courses/dunloskyimprovinglearning.pdf>.

Fases de teste de software. Disponível em: <http://www.matera.com/br/2013/07/19/fases-de-testes-de-software/>. Acesso em 11 de nov. de 2016.

FERREIRA, Ana Lúcia Duarte. **Informática educativa na educação infantil: Riscos e Benefícios**. Fortaleza: Universidade Federal do Ceará, 2002.

Implantação de software. Disponível em: <https://pt.wikipedia.org/wiki/Implanta%C3%A7%C3%A3o_de_software>. Acesso em: 7 de nov. de 2016.

JUCÁ, Sandro César Silveira. **A relevância dos softwares educativos na educação profissional,** Ciências & Cognição, Vl.8, 2006.

RiouxSVN. Disponível em: <https://riouxsvn.com/>. Acesso em: 8 de set. de 2016.

SANTIAGO, Heber. **Coisas de desenvolvimento, Android: Coding Style – Convenções oficiais do Android**. 2014. Disponível em: <http://coisasemandroid.blogspot.com.br/2014/04/coding-style-convencoes-oficiais-do.html>.

SENGER, Vinicius; XAVIER, Kleber. **33 design patterns aplicados com Java: introdução a design patterns**. 2011. Disponível em: <http://pt.slideshare.net/vsenger/33-design-patterns-com-java>.

APÊNDICE A - LISTA DE ATIVIDADES, DIAGRAMA DE REDE E CRONOGRAMA DAS ATIVIDADES

O apêndice A encontra-se no arquivo digital “APÊNDICE A - LISTA DE ATIVIDADES, DIAGRAMA DE REDE E CRONOGRAMA DAS ATIVIDADES.mpp”.

APÊNDICE B - RELATÓRIO DE DESEMPENHO

O apêndice B encontra-se no arquivo digital “APÊNDICE B - RELATÓRIO DE DESEMPENHO.xlsx”.

APÊNDICE C - CONTROLE DE MUDANÇAS

O apêndice C encontra-se no arquivo digital “APÊNDICE C - CONTROLE DE MUDANÇAS.xlsx”.

APÊNDICE D - GERÊNCIA DA QUALIDADE

O apêndice D encontra-se no arquivo digital “APÊNDICE D - GERÊNCIA DA QUALIDADE.xlsx”.

APÊNDICE E - GERENCIA DE RISCOS

O apêndice E encontra-se no arquivo digital “APÊNDICE E - GERENCIA DE RISCOS.xlsm”.

APÊNDICE F - DICIONÁRIO DE DADOS

O apêndice F encontra-se no arquivo digital “APÊNDICE F - DICIONÁRIO DE DADOS.docx”.

APÊNDICE G - DESCRIÇÃO DOS CASOS DE TESTE

O apêndice G encontra-se no arquivo digital “APÊNDICE G - DESCRIÇÃO DOS CASOS DE TESTE.docx

APÊNDICE H - HISTÓRICO DE REALIZAÇÃO DOS CASOS DE TESTE

O apêndice H encontra-se no arquivo digital “APÊNDICE H - HISTÓRICO DE REALIZAÇÃO DOS CASOS DE TESTE.docx”.

APÊNDICE I - MANUAL DO USUÁRIO

O apêndice I encontra-se no arquivo digital “APÊNDICE I - MANUAL DO USUÁRIO.docx”.

APÊNDICE J - DIAGRAMAS DE CASOS DE USO

O apêndice J encontra-se no diretório “APÊNDICE J - DIAGRAMAS DE CASOS DE USO”.

APÊNDICE K - CENÁRIOS DOS CASOS DE USO

O apêndice K encontra-se no diretório “APÊNDICE K - CENÁRIOS DOS CASOS DE USO”.

APÊNDICE L - MODELOS ENTIDADE E RELACIONAMENTO

O apêndice L encontra-se no diretório “APÊNDICE L - MODELOS ENTIDADE E RELACIONAMENTO”.

APÊNDICE M - DIAGRAMAS DE CLASSES - APLICAÇÃO CLIENTE

O apêndice M encontra-se no diretório “APÊNDICE M - DIAGRAMAS DE CLASSES - APLICAÇÃO CLIENTE”.

APÊNDICE N - DIAGRAMAS DE CLASSES - APLICAÇÃO SERVIDORA

O apêndice N encontra-se no diretório “APÊNDICE N - DIAGRAMAS DE CLASSES - APLICAÇÃO SERVIDORA”.

APÊNDICE O - DIAGRAMAS DE OBJETOS

O apêndice O encontra-se no diretório “APÊNDICE O - DIAGRAMAS DE OBJETOS”.

APÊNDICE P - ESTIMATIVAS POR PONTOS DE CASOS DE USO E FUNÇÃO

O apêndice P encontra-se no diretório “APÊNDICE P - ESTIMATIVAS POR PONTOS DE CASOS DE USO E FUNÇÃO”.

APÊNDICE Q - DIAGRAMAS DE SEQUÊNCIA - APLICAÇÃO CLIENTE

O apêndice Q encontra-se no diretório “APÊNDICE Q - DIAGRAMAS DE SEQUÊNCIA - APLICAÇÃO CLIENTE”.

APÊNDICE R - DIAGRAMAS DE SEQUÊNCIA - APLICAÇÃO SERVIDORA

O apêndice R encontra-se no diretório “APÊNDICE R - DIAGRAMAS DE SEQUÊNCIA - APLICAÇÃO SERVIDORA”.