Universidade Federal Fluminense
Centro Tecnológico
Instituto de Computação
Curso de Sistemas de Informação
Curso de Ciência da Computação

IGOR BARBOSA PINTO
TIAGO CÂNDIDO ALMEIDA SANTOS

INTEGRA UFF - SISTEMA DE GESTÃO DE APRENDIZADO

Niterói-RJ
2016

IGOR BARBOSA PINTO
TIAGO CÂNDIDO ALMEIDA SANTOS

INTEGRA UFF - SISTEMA DE GESTÃO DE APRENDIZADO

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação e Bacharel de Sistemas de Informação.

Orientador: Prof. Dr. Raphael Pereira de Oliveira Guerra

Niterói-RJ
2016

IGOR BARBOSA PINTO
TIAGO CÂNDIDO ALMEIDA SANTOS

INTEGRA UFF - SISTEMA DE GESTÃO DE APRENDIZADO

Trabalho submetido aos Cursos de Bacharelado em Ciência da Computação e Sistemas de Informação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação e Bacharel em Sistemas de Informação.

Aprovado por:

BANCA EXAMINADORA

Prof. Dr. RAPHAEL PEREIRA DE OLIVEIRA GUERRA - Orientador
UFF

Prof. NOME DO PROFESSOR
INSTITUIÇÃO

Prof. NOME DO PROFESSOR
INSTITUIÇÃO

Niterói-RJ
2016

”Existem apenas duas coisas difíceis em Ciência da Computação: invalidar cache e nomear as coisas.”
(KARLTON, Phil)

Agradecimentos

Espaço reservado para os agradecimentos.

Agradecimento 1.

Agradecimento 2.

...

Agradecimento N.

Os agradecimentos devem ser sucintos e específicos a cada tipo de ajuda, a cada idéia relevante, a cada empréstimo significativo, pois um agradecimento é, de certa forma, um crédito dado a alguém [**?**].

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API: *Application Programming Interface*

SGA: Sistema de Gestão de Aprendizagem

AVA: Ambiente Virtual de Aprendizagem

CVDS: Ciclo de Vida de Desenvolvimento de Sistemas

TDD: *Test-driven Development*

XP: *Extreme Programming*

LMS: *Learning Management System*

UFF: Universidade Federal Fluminense

COC: *Convention Over Configuration*

URL: *Uniform Resource Locator*

REST: *Representational State Transfer*

TFD: *Test-First Development*

MVC: Modelo-Visão-Controlador

HTTP: *Hypertext Transfer Protocol*

HTML: *Hypertext Markup Language*

JSON: *JavaScript Object Notation*

CSS: *Cascading Style Sheets*

GPA: *Grade Point Average*

Sumário

Resumo

As disciplinas dos cursos da Universidade Federal Fluminense possuem informações alocadas em diversos Sistemas de Gestão de Aprendizado. Desta forma a tarefa de gerenciar os conteúdos disponibilizados pelos professores pode se tornar bastante custosa.

Com o objetivo de diminuir o trabalho manual realizado pelos alunos, propomos a criação de uma aplicação que possibilite agregar as informações das várias disciplinas em uma única interface de usuário, independente de onde ela foi disponibilizada pelo professor.

Para alcançar este objetivo, desenvolvemos um aplicativo móvel multi-plataforma, que tem como propósito reunir, em uma única ferramenta, múltiplos ambientes de ensino, apresentando de maneira transparente ao usuário os serviços oferecidos pelos diferentes ambientes de ensino, como eventos, tópicos, arquivos, mensagens, todos em uma única interface.

O uso do aplicativo tem como objetivo aproximar alunos e professores das diversas disciplinas ministradas na universidade, intensificar a interação das turmas, centralizar as informações e materiais da disciplina e prover uma comunicacção rápida e eficiente.

Palavras-chave: Sistema de Gestão de Aprendizado, Aplicação Mobile, Phonegap, Ionic, Padrões de Projeto, Ruby on Rails.

Abstract

The subjects of Universidade Federal Fluminense courses have information in several distinct Learning Management Systems. Thus the task of managing the content provided by the professors can become quite cumbersome.

In order to reduce the manual work done by the students, we proposed to create an application that allows aggregating information from various subjects and systems in a single user interface, regardless of where it was made available by the professors.

To accomplish this , we developed a multi -platform mobile application that aims to bring together in a single tool , multiple learning environments , presenting transparently to the user the services offered by different educational environments, such as events , topics, files , messages , all in a single interface.

The use of the application aims to bring together students and professors from various subjects taught at the university, intensify the interaction between classes, centralize the information and materials of the course and provide a fast and efficient communication.

Keywords: Learning Management System, Mobile Application, Ionic, Design Patterns, Phonegap, Ruby on Rails.

Capítulo 1

Introdução

O uso de tecnologias digitais educacionais se torna cada vez mais presente nas universidades do Brasil e do mundo. Esta transformação das vias tradicionais de transferência do conhecimento gera grande impacto na restruturação e modernizaçao do aprendizado [15], incorporando um vasto ferramental às incumbências do ensino. Cria-se novos canais de comunicação que permitem desde a disseminação do conteúdo lecionado até o gerenciamento do calendário de atividades do curso.

A combinação do uso destas tecnologias com o ensino tradicional em sala de aula cria um ambiente de *blended learning*, que provou ser muito bem sucedido em muitos casos [36].

A escolha da Tecnologia empregada em determinada Instituição de Ensino, em sua totalidade ou em suas menores partes organizacionais e cursos específicos, terá grande influência no grau de aprimoramento das interações de ensino pretendidas em relação ao potencial absoluto que essas tecnologias oferecem.

O foco deste trabalho apresenta uma solução computacional, para uma possível segmentação na adoção de tecnologias educacionais, visando reduzir os impactos negativos dessa pluralidade de tecnologias no cotidiano dos seus usuários.

1.1 Cenário Atual

A redução do custo dos computadores e dispositivos portáteis, como smartphones e tablets, a ampliação do acesso à Internet em conexões de alta velocidade e a difusão de sistemas e tecnologias de informação, impulsionaram a adoção de tecnologias educacionais e uma reformulação das metodologias de aprendizado tradicionais.

Na última década, presenciamos o crescimento espressivo da oferta em Educação a distância e semi presencial [17], consolidadas através do uso de Sistemas de Gestão de Aprendizado (SGA) como facilitadores do processo de aprendizado à distância. Em países como Australia, Reino Unido, Canadá e Estados Unidos, a adesão a algum SGA supera supera 90% das IES [18][16], na Espanha a adesão é de 100% das instituições [19], .

A mesma tendência também pode ser observada nos ambientes de ensino presencial formal das

Instituições de Ensino Superior brasileiras [21] com o crescimento anual de 21,5% na adesão de plataformas digitais de ensino [22], ocupando atualmente a 3^a posição mundial em número de registros da ferramenta líder mundial deste segmento, com 4,383 instalações ativas [23] .

Conforme Oliveira [13], o uso dos recursos tecnológicos no ambiente acadêmico proporciona ao aluno uma aprendizagem mais envolvente, como também faz com que os discentes passem a aprender precocemente os conteúdos e aprimorar suas habilidades que são estimuladas de maneira tardia ou que talvez não sejam exploradas. O uso pedagógico destes recursos tem potencializado os processos de interação entre estudantes e professores [14].

1.2 Problema

Na instituição em que é desenvolvido este trabalho, no âmbito do instituto e cursos nos quais os autores estão inseridos, as tecnologias computacionais utilizadas no ensino estão distribuídas em uma série diversa de plataformas e tipos de mídia, com seus critérios de adoção muitas vezes exclusivamente delineados pela preferência dos professores ou alunos de cada disciplina em particular, ocasionando uma grande descentralização do conteúdo necessário para o curso do semestre letivo entre as diversas disciplinas de cada aluno em particular. As possibilidades se dividem entre sites de professores e disciplinas, redes sociais de uso geral, pastas compartilhadas em sistemas de arquivos na nuvem, Sistemas de Gestão de Aprendizado variados, listas de e-mails, grupos de mensageiros instantâneos, entre outros.

1.3 Proposta

Este trabalho propõe um projeto de aplicativo, implementado pelos autores para a disciplina de Projeto de Aplicação I, com a especificação de seus requisitos e a apresentação das tecnologias utilizadas para a implementação da sua versão inicial.

1.4 Público destinado

Este documento destina-se a toda a comunidade acadêmica da Universidade Federal Fluminense, bem como aos próprios autores, como forma de concretizar a solução de um problema no qual estiveram inseridos durante o decorrer das suas formações, ao orientador do projeto, como material de apoio ao desenvolvimento e gerenciamento das atividades planejadas e a possíveis futuros desenvolvedores quem venham a estender o escopo de funcionalidades do projeto.

1.5 Escopo do Produto

O produto se trata de um aplicativo móvel multi-plataforma, que tem como propósito reunir, em uma única ferramenta, múltiplos ambientes de ensino, apresentando de maneira transparente ao usuário os serviços oferecidos pelos diferentes ambientes de ensino, como eventos, tópicos, arquivos, mensagens, todos em uma única interface. O uso do aplicativo tem como objetivo aproximar alunos e professores das diversas

disciplinas ministradas na universidade, intensificar a interação das turmas, centralizar as informações e materiais da disciplina e prover uma comunicação rápida e eficiente. Além disso, a ferramenta propõe a ludificação das atividades no grupo, através de desafios, marcos, medalhas e outros items que possam ser utilizados pelo professor da disciplina para gerar indicadores.

Capítulo 2

Sistemas de Gestão de Aprendizado

Um sistema de gestão de aprendizagem (SGA/LMS), também chamado de um ambiente virtual de aprendizagem (AVA), é um software que permite a criação de cursos na forma de sistemas on-line [24]. Um LMS é geralmente um sistema protegido mediante autenticação e autorização de recursos que permite que a instituição de ensino disponibilize vários ambientes de cursos com relativa facilidade. O ambiente do curso é normalmente gerido pelo instrutor (educador). O educador tem a autorização para fazer *upload* de conteúdo para o *site*, organizar os materiais no continuum educativo que reflete o curso, grupos de discussão abertos e gerenciar as informações enviadas para os grupos de notícias, incluindo a moderação de conteúdos impróprios. O educador pode visualizar relatórios dos usuários, receber atividades e trabalhos dos alunos, a fim de avaliá-lo. Em muitos LMSs o sistema está ligado a outros sistemas administrativos na organização, tais como o sistema de matrícula, o sistema de lançamento de notas, e assim sucessivamente. As permissões dos alunos são geralmente mais limitadas do que as do educador. Alunos inscritos em um determinado curso podem visualizar o conteúdo e baixá-lo. Eles podem participar de atividades interativas que acontecem em fóruns e em alguns casos também podem contribuir com conteúdo em locais específicos, tais como ambientes wiki ou repositórios especiais de colaboração definidos pelo gestor do curso. Diferentes sistemas de gestão de aprendizagem têm diferentes interfaces de usuário e características diferentes. No entanto, todos eles compartilham três funções-chave [20] [25].

- Sistema de gerenciamento de conteúdo: Permite a criação ou o *upload* de uma variedade de itens de conteúdo, como textos, apresentações, artigos digitalizados e materiais audio-visual. O sistema de gerenciamento de conteúdo também permite que o material a ser organizado em uma estrutura planejada pelo administrador do curso, criando pastas para temas e conteúdos.

- Ferramentas para gerenciamento de interações: Diferentes sistemas de gestão de aprendizagem permitem que o instrutor para abrir diferentes fóruns. Alguns sistemas permitem a abertura de espaços assíncronos de colaboração, tais como *wikis* e *blogs*, e alguns podem fornecer comunicação síncrona usando bate-papo e outras ferramentas de conferência *on-line*.

- Ferramentas para gerenciar e avaliar alunos: Alguns sistemas fornecem ferramentas administrativas para tarefas de gravação, notas e *feedback*. Eles também fornecem relatórios de usuários que

suportam o instrutor na medição do nível da participação dos alunos e na avaliação das realizações dos alunos.

2.1 Parte contemplada no escopo da proposta

São contempladas no escopo do projeto proposto apenas as funcionalidades expositivas mais comuns e consideradas mais importantes para sistemas deste tipo. O enfoque da aplicação se dará em notificar e apresentar para os alunos inscritos nos cursos os conteúdos disponibilizados e comunicações disseminadas através das plataformas registradas no aplicativo, de modo que estas informações possam ser agregadas de forma homogênea e transparente no que se refere a origem daquela informação.

2.2 Sistemas Referenciais Escolhidos e Motivação da Escolha

Os fatores determinantes para a escolha dos SGA nos quais este projeto se baseia, na forma como a arquitetura do software foi projetada e de que forma se permitirá sua extensibilidade, foram os seguintes:

- Tipo e custo de licensa de uso do software
- Capacidade e facilidade de extensão do software
- Adesão e familiaridade no uso da ferramenta já existente na comunidade acadêmica da instituição onde se encontra o público alvo
- A existência de uma camada de serviço que possibilite o acesso dos recursos do software através da rede móvel do usuário

Visto que as plataformas de código aberto são adequadas para as universidades e outras instituições de ensino [26], pelos seguintes motivos:

1. Elas permitem que instituições de ensino tenham controle do software
2. O custo de usar a licença é muito baixo ou inexistente
3. Licença de código aberto permite qualquer alteração, modificação e melhoria no LMS.

Desta forma, selecionamos a rede social de fins acadêmicos disponibilizada pela Universidade Federal Fluminense, o Conexão UFF, e a ferramenta *open source* líder de mercado e já utilizada em alguns cursos da Universidade, o Moodle.

2.3 Conexão UFF

A plataforma foi inicialmente concebida como projeto pessoal de um membro da Superintendência de Tecnologia da Informação da UFF e aluno do curso de Sistemas de Informação sob o nome ”Orkuff”, que fazia alusão a rede social *Orkut*, ainda muito popular na época.

Em 2011 a plataforma foi lançada oficialmente pela universidade sob o nome de Conexão UFF, já integrada ao sistema único de autenticação da UFF, o Iduff. Dentre as funcionalidades apresentadas na versão inicial estavam os grupos, formados por participantes das respectivas turmas das disciplinas dos cursos, com recursos de fórums de discussão, upload de arquivos e troca de mensagens diretas. O professor automaticamente recebia permissões de moderação do referido grupo.

2.3.1 Camada de Serviços

Para o escopo desse trabalho foi criada uma camada de serviços na aplicação do Conexão UFF de modo que os recursos disponíveis no sistema pudessem ser expostos através de uma arquiterura de serviços *RESTful*.

Para cada um dos recursos disponíveis foi criado um ponto de acesso, que exibe as informações permitidas para o cliente autenticado.

A autenticação é feita pela camada de serviços do Sistema Único de Autenticação da UFF, o IDUFF. O processo de autenticação se dá através dos seguintes passos:

1. O usuário cadastra suas credenciais, IDUFF e Senha, no aplicativo.
2. O aplicativo faz uma requisição ao serviço do IDUFF enviando as credenciais salvas no aplicativo
3. O serviço de autenticação do IDUFF valida as credenciais e caso se confirme a validade dos dados, um token de acesso é enviado ao Cliente, com validade de 20 minutos.
4. A cada requisição às APIs do Conexão UFF, o token é enviado junto ao pedido
5. O Conexão UFF verifica através do serviço do IDUFF se o token enviado é válido e caso se confirme a validade o IDUFF estende a validade do token por mais 20 minutos.
6. Com o token validado o Conexão continua com o processamento do pedido e responde à requisição com o conteúdo requisitado.

A arquitetura da camada de serviços do conexão UFF é apresentada na 2.1

O formato escolhido para a transferência dos dados oriundos do Conexão UFF foi o JSON, por conta da fácil manipulação nativa dos objetos pelo Javascript, linguagem escolhida para o desenvolvimento da aplicação cliente, definido em 5.2.

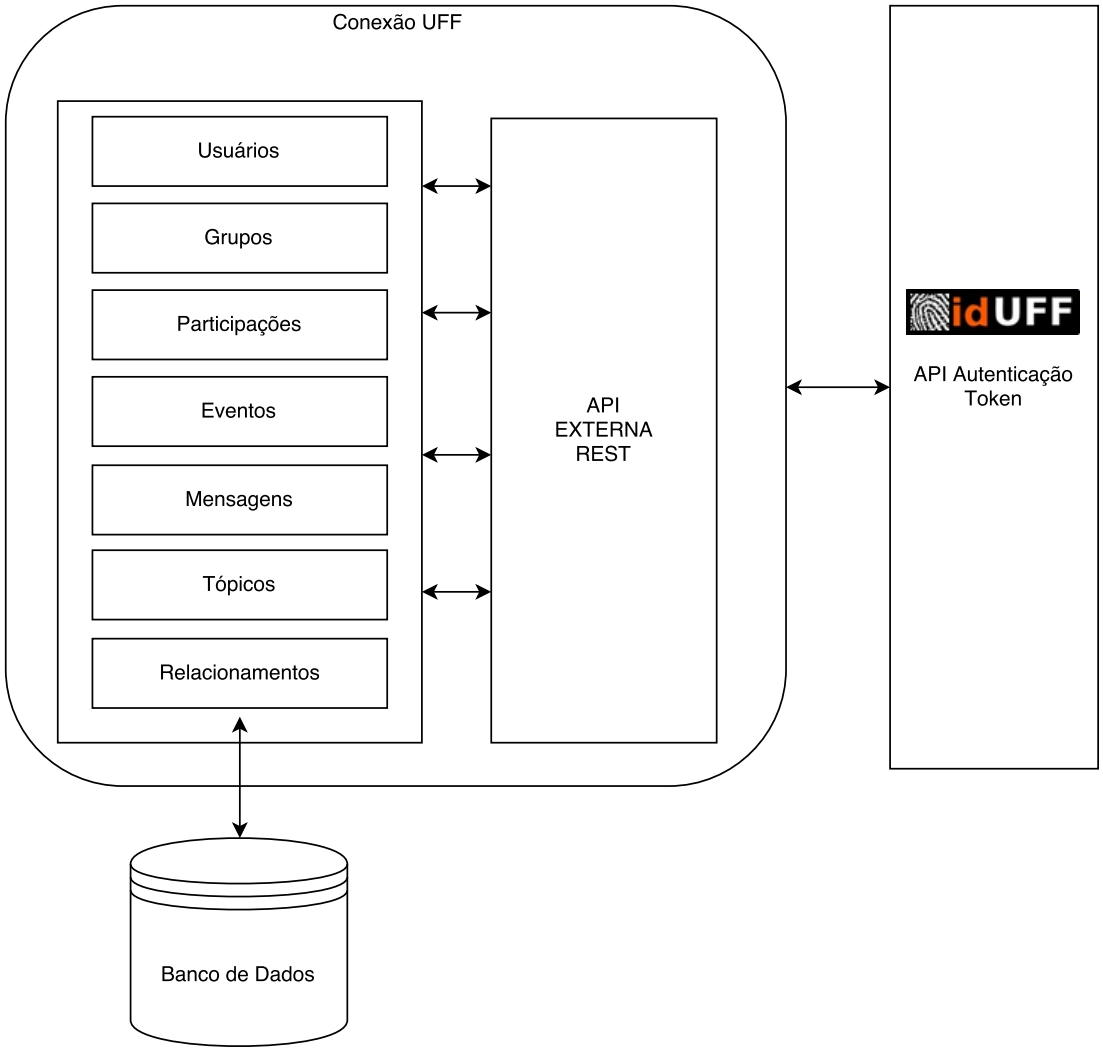

Figura 2.1: Arquitetura da camada de Serviços Conexão UFF

2.4 Moodle

A plataforma mais utilizada para cursos acadêmicos on-line é o Moodle. O Moodle é um pacote de software especialmente concebido para ajudar professores a criar cursos online e oferece as funcionalidades de um LMS completo [35]

O Moodle é um software de código aberto, que pode ser baixado gratuitamente a partir da Internet, usado, modificado, e até mesmo distribuído (sob uma licença GNU). O Moodle é facilmente executado em UNIX, Linux, Windows, Mac OS X, ou qualquer outro sistema que suporte PHP. Todos os seus dados são gravados em um único banco de dados, MySQL ou PostgreSQL são os mais suportados; no entanto, Oracle, Access, Interbase e ODBC também podem ser utilizados com algumas configurações. Dentre as funcionalidades do Moodle estão incluídos os fóruns, recursos de publicação de conteúdo, questionários, chats, atribuições, e outros recursos que geralmente são suficientes para a criação de cursos padrão.

A unidade organizacional básica do Moodle é o curso, que é acessado através de uma página web.

Um curso é organizado em seções que podem corresponder aos tópicos ou semanas, aparecendo na coluna do meio da página. É possível incluir diferentes recursos e atividades em todas as seções. O último está a ser atribuído como casa ou classe trabalho a ser desenvolvido pelos alunos. Os usuários são outro objeto Moodle essencial: eles podem se inscrever em cursos diferentes, como administradores, professores ou estudantes. Cada função é definida por suas capacidades em um determinado contexto, o que significa que eles têm um conjunto de privilégios ao executar determinadas ações [37].

De acordo com o estudo realizado nos Estados Unidos [32], Blackboard e Moodle são os LMS com maior quota de mercado, sendo que o Moodle tem maior satisfação entre os usuários. Em nosso projeto, Blackboard foi descartado e escolhemos Moodle, pois além de ser *Open Source*, ele já é utilizado em diversos cursos da *UFF*.

2.4.1 Camada de Serviços

Moodle é composto por três elementos principais: o *Core*, os módulos de atividades e os *Plugins*. O *Core* inclui as funcionalidades básicas da plataforma de aprendizagem, tais como o fórum, wiki e atividades.

Os *Plugins* são blocos de software que adicionam extensões funcionais ao sistema. Dentro do Moodle, diferentes interfaces de plug-ins podem ser encontradas. Uma destas interfaces é a interface de Serviços Web, que é extensível e garante a escalabilidade do sistema em função dos diversos protocolos de comunicação.

Em 2008 foi projetada uma solução para fornecer algumas das funcionalidades através de Serviços Web, esta solução foi a Arquitetura de Serviços do Moodle, que foi implementada para Moodle 2.0 e lançada no final de 2009 [38].

A arquitetura de Serviços do Moodle acrescenta duas camadas lógicas à arquitetura do Moodle (mostrado na Figura 2.2). A primeira é API externa, um conjunto de arquivos php que incluem a lógica de cada serviço. A segunda é a camada de conectores. A arquitetura de Serviços do Moodle não está vinculada a um protocolo de webservices específica; ele é projetada para ser independente de protocolo. Para cada protocolo suportado (SOAP, REST, XML-RPC, etc.) há um módulo conector específico nesta camada. Cada conector implementa a tradução dos métodos implementados na API externa com o protocolo e sintaxe específica. Além disso, o conector também oferece outros serviços necessários como autenticação, autorização e outros serviços de infra-estrutura. A camada de conectores é uma camada expansível que permite que o adição de novos protocolos de comunicação.

Um elemento-chave na concepção da arquitetura de Serviços do Moodle é sua capacidade de ampliação baseada em plugins. A API externa pode ser estendida de forma segura, dando total controle e segurança para o administrador da plataforma. Se for necessário um novo tipo de protocolo de serviços web ou método de autenticação, um desenvolvedor pode criar um novo conector de webservices para implementá-lo.

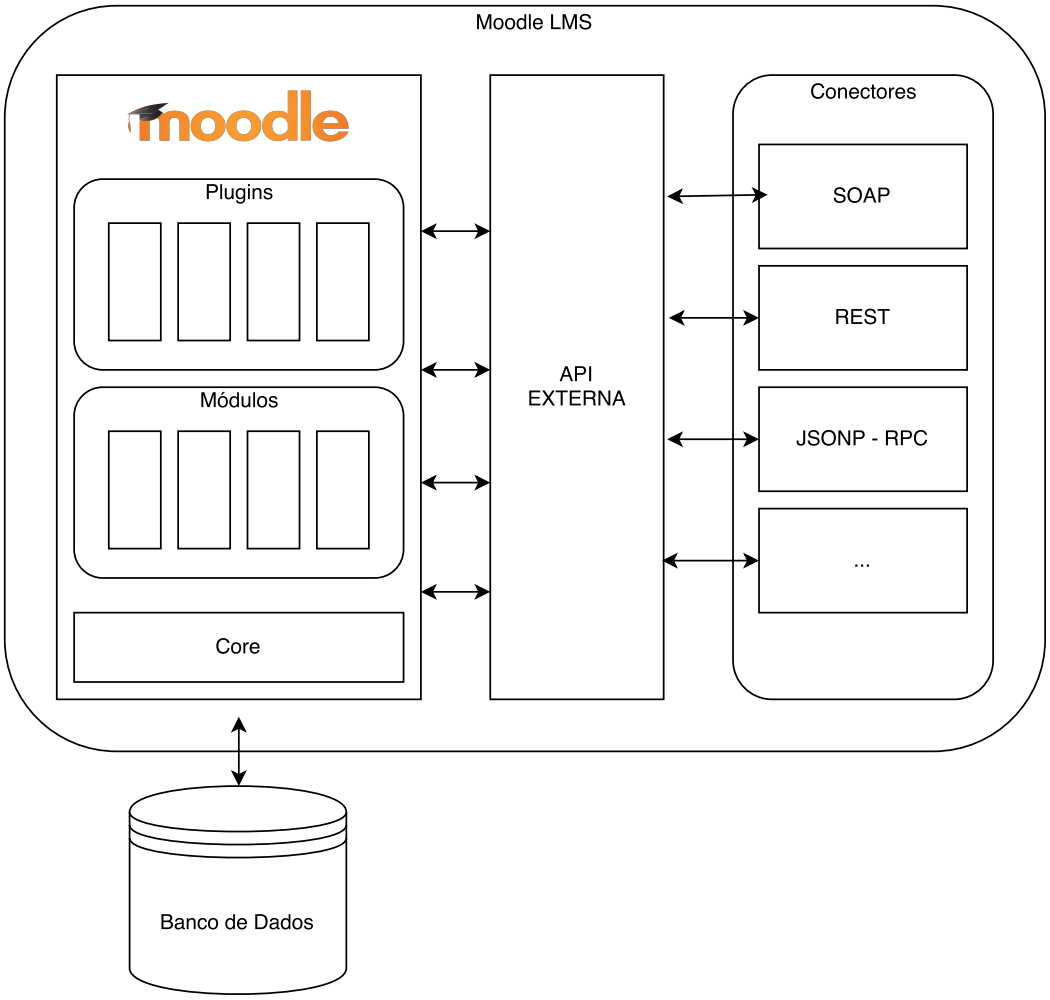

Figura 2.2: Arquitetura da camada de Serviços Moodle [38]

Capítulo 3

Desenvolvimento de Software

Neste capítulo serão descritas algumas das metodologias de desenvolvimento de sistemas que se demonstraram adequadas ao contexto e escopo do projeto e foram utilizadas no desenvolvimento da aplicação.

O desenvolvimento tradicional, também conhecido como cascata, acontece de forma linear e em várias fases onde cada uma utiliza o resultado da fase anterior. Ou seja, primeiro a equipe realiza a análise, seguido do design, implementação, teste e assim por diante. Em contrapartida o caráter incremental e evolutivo das Metodologias Ágeis permite que o planejamento do desenvolvimento subsequente a cada reunião com o orientador do projeto, bem como a avaliação das tarefas que foram planejadas na reunião anterior tenham um ritmo bem definido e ocorra de forma iterativa com ciclos de desenvolvimento se repetindo, onde todas as fases necessárias podem e devem se repetir a cada ciclo. Ao final de cada um destes, é entregue uma versão utilizável do *software*. Começando por uma versão bem simples na primeira iteração até que ele alcance o resultado desejado ao final do desenvolvimento.

A programação em par, o desenvolvimento orientado a testes, as refatorações, medições e a integração continua foram meios de manter a qualidade das entregas e a garantia de funcionamento integral do produto a medida que novas funcionalidades são adicionadas.

A seguir apresentaremos as metodologias utilizadas e como foram utilizadas no processo de desenvolvimento deste projeto.

3.1 Metodologia Ágil de Desenvolvimento de Software

O Desenvolvimento Ágil é baseado na idéia de desenvolvimento incremental e iterativo, em que as fases que compõem o Ciclo de Vida de Desenvolvimento de Sistemas (CVDS) são revisitadas repetidamente. O *software* é melhorado iterativamente usando *feedback* do cliente para convergir em soluções [27].

No desenvolvimento ágil, ao invés de um modelo de processo único e grande como os implementados em CVDS convencionais, o CVDS é dividido em partes menores, chamadas ”incrementos”ou ”iterações”, em que cada um destes incrementos aborda uma das fases do CVDS convencional. De acordo com o Manifesto Ágil, os principais fatores incluem quatro seguintes:

1. O envolvimento do cliente desde cedo

2. O desenvolvimento iterativo

3. Equipes auto-organizadas

4. Adaptação à mudanças

Atualmente seis métodos são identificados como métodos ágeis de desenvolvimento: Metodologia Crystal, o Método Dinâmico de Desenvolvimento de Sistemas, *Feature Driven Development*, *Lean Software*, SCRUM e *Extreme Programming* [3]. Os dois últimos foram escolhidos como metodologia de desenvolvimento para este projeto e serão apresentados nas seções seguintes.

3.2 SCRUM

Scrum pertence à família de metodologias ágeis de desenvolvimento de *software* que têm atraído significativa atenção entre os profissionais de *software* durante os últimos cinco anos. Enquanto o método *Extreme Programming*, que tem sido largamente aceito como uma das mais importantes abordagens ágeis, tem seus conceitos bem definidos acerca da programação (programação em par, padrões de codificação, desenvolvimento orientado à testes, refatoração, integração contínua), o Scrum se concentra na gestão de projetos de software. Por essa divisão clara das aplicabilidades de ambas metodologias, foi possível adotá-las de forma complementar para o desenvolvimento deste projeto.

Scrum parte da premissa de que o desenvolvimento de software é complexo e imprevisível demais para ser planejado antecipadamente com exatidão. Em vez disso, o controle de processos empíricos deve ser aplicado para assegurar a visibilidade, inspeção e adaptação. As diferentes variáveis ambientais e técnicas (tais como período de tempo, qualidade, requisitos, recursos, tecnologias de implementação e ferramentas, e até mesmo métodos de desenvolvimento) devem ser controladas constantemente a fim de poder se adaptar às alterações de forma flexível. Isto é obtido através de um processo de desenvolvimento iterativo e incremental.

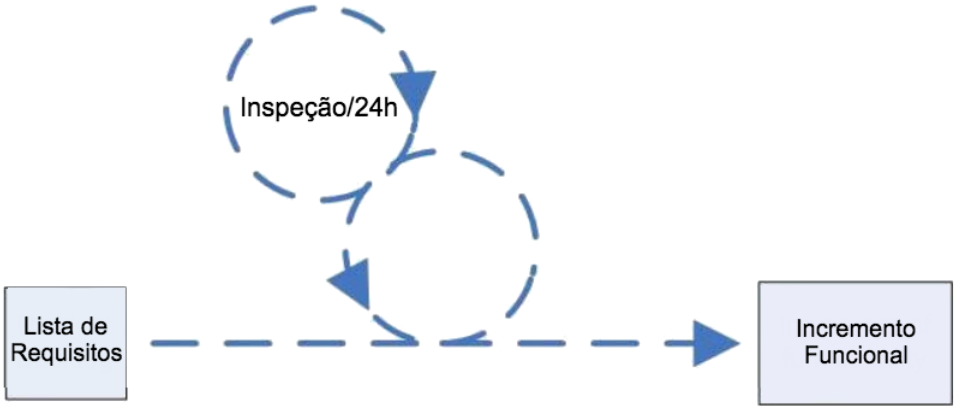

Figura 3.1: Esqueleto SCRUM

O esqueleto de scrum é mostrado na Figura 3.1 [29]

O círculo inferior representa uma iteração das atividades de desenvolvimento que ocorrem sequencialmente. O resultado de cada iteração é um incremento do produto. O círculo superior representa a inspeção diária que ocorre durante a iteração, em que os membros da equipe se reúnem para inspecionar as atividades uns dos outros e fazer as adaptações necessárias. Uma lista de requisitos é o que conduz a iteração. Este ciclo se repete até que o projeto seja concluído.

O Scrum implementa este ciclo iterativo e incremental através de três papéis: o *Product Owner*, a equipe, e o *Scrum Master*.

- O *Product Owner* é responsável por representar os interesses de todos os interessados no projeto do sistema final. Ele mantém o *Product Backlog*, isto é, uma lista priorizada de requisitos do projeto com tempos estimados para transformá-los em funcionalidade do produto concluído.

- A Equipe é responsável pelo desenvolvimento das funcionalidades. As equipes são auto-geridas, auto-organizadas, interfuncionais, e eles são responsáveis por descobrir como transformar o *Backlog* do Produto em um incremento de funcionalidade dentro de uma iteração e gerir o seu próprio trabalho para fazê-lo. Os membros da equipe são coletivamente responsáveis pelo o sucesso de cada iteração e do projeto como um todo.

- O *Scrum Master* preenche a posição normalmente ocupada pelo gerente de projeto, mas seu papel é ligeiramente diferente. Enquanto o gerente de projeto tradicional é responsável pela definição e gestão do trabalho, o *Scrum Master* é responsável por gerenciar o processo Scrum, ensinando o Scrum para todos os envolvidos no projeto, implementando o Scrum que caiba dentro da cultura da organização em questão, entregando os benefícios esperados, e assegurando que todos sigam as regras e práticas do Scrum.

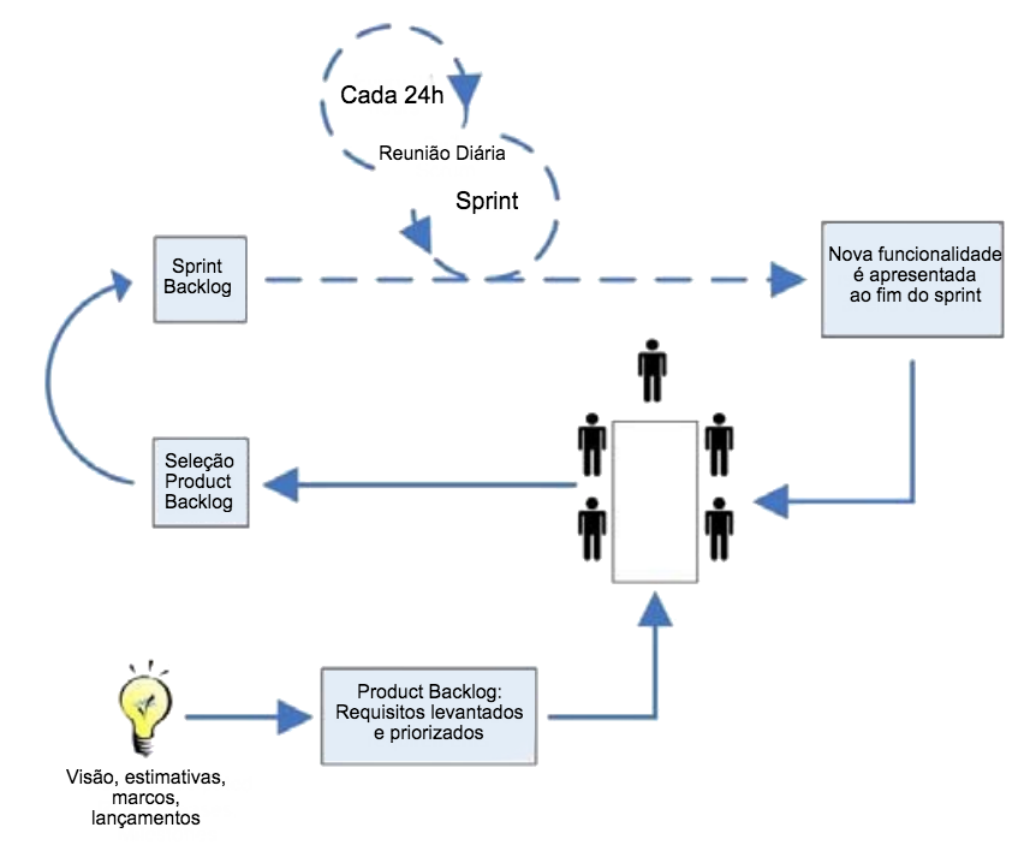

Figura 3.2: Esqueleto SCRUM detalhado

O fluxo Scrum detalhado é mostrado na Figura 5.2 [29]..

O projeto de um Scrum começa com o vislumbre do sistema a ser desenvolvido. Em seguida o *Product Backlog*, uma lista é criado contendo todos os requisitos já conhecidos. O *Product Backlog* é priorizado e dividido em lançamentos propostos. Todo o trabalho é feito em divisões temporais chamadas de *Sprints*. Cada *Sprint* é uma iteração 30 dias consecutivos ou outro intervalo que melhor se adeque a organização.

Cada *Sprint* é iniciado com uma reunião de planejamento, onde o *Product Owner* e a equipe se juntam para planejar o que será feito para o próxima iteração. O *Product Owner* diz à equipe o que é desejado e para ele tem o mais alto nível de prioridade do *Product Backlog*, a equipe diz ao *Product Owner* o quanto daquilo acredita que pode se transformar em funcionalidade ao longo do próximo *Sprint*. Depois de decidir o que deve ser feito, a equipe desenvolve o *Sprint Backlog*, isto é, uma lista de tarefas que devem ser executadas para entregar um incremento completo de funcionalidade do produto potencialmente utilizável até o final da iteração. As tarefas na lista emergem a medida que o *Sprint* evolui e devem ser divididas de modo que cada uma leve entre 4 e 16 horas para ser concluída. A cada dia, a equipe se reúne para uma reunião de 15 minutos chamada Reunião Diária. No Scrum diário, cada membro da equipe responde a três perguntas: O que você fez neste projeto desde a última reunião diária? O que você vai fazer antes da próxima reunião? Você tem obstáculos? O objetivo da reunião é sincronizar o trabalho de todos os membros da equipe e para agendar quaisquer reuniões que a equipe precise para transmitir o seu progresso.

No desenvolvimento deste projeto, foram atribuídos os papéis de *Product Owner* para o professor orientador do projeto, e os autores dividiam as atribuições de *Scrum Master* e da equipe. Os *Sprints* tinham duração semanal delimitados pelas nossas reuniões de orientação, onde era feito a revisão das entregas e o planejamento do *Sprint* seguinte, com os próximos requisitos selecionados e priorizados pelo orientador e acordados com a equipe.

3.3 *Extreme Programming*

Extreme Programming, ou XP, de acordo com Teles [3], é um processo de desenvolvimento de *software* voltado para:

- Projetos cujos requisitos são vagos e mudam com frequência.
- Desenvolvimento de sistemas orientados a objeto.
- Equipes pequenas, preferencialmente até 12 desenvolvedores.
- Desenvolvimento incremental (ou iterativo), onde o sistema começa a ser implementado logo no início do projeto e vai ganhando novas funcionalidades ao longo do tempo.

Baseado nisso o XP se tornou um processo determinante para o bom andamento do desenvolvimento do *software*. Entretanto, devido ao ambiente em que o desenvolvimento estava inserido, por exemplo não existia um cliente, papel que a maioria das vezes foi representado pelo orientador, foram adotadas apenas algumas da práticas do XP.

Algumas dessas práticas foram, o jogo do planejamento, onde a equipe se reunia semanalmente para obter *feedbacks* e baseado nisso realizar o planejamento da semana, sempre priorizando as tarefas julgadas mais importantes para o sistema. Programação em par, embora não executada o tempo todo, como prega o XP, esta prática foi utilizada diversas vezes pela equipe quando foi necessário o desenvolvimento de de partes mais complexas do sistema. Dessa forma os dois desenvolvedores contribuíam diretamente com revisões e opiniões enquanto a funcionalidade era construída, gerando uma solução mais eficaz ao final. Código Coletivo, todos os desenvolvedores eram responsáveis e possuíam acesso a todo o código. Código padronizado e *design* simples. Além disso a Integração Contínua, na qual as novas funcionalidades do sistema eram incorporadas ao código o mais rápido possível. Integração coberta por testes automatizados que garantem que o sistema ainda esteja funcionando após cada integração. Tais testes foram criados utilizando a prática conhecida como TDD descrita a baixo.

3.4 *Test Driven Development*

Test Driven Development (TDD) [30], é uma técnica de desenvolvimento de software que combina as técnicas *test-first development* (TFD), no qual você escreve seus testes antes do código, e refatoração. Ambas muito relacionadas com o desenvolvimento ágil. Segundo Beck (2003), criador do TDD, além de um dos idealizadores do Manifesto Ágil, Desenvolvimento Orientado a Testes deve encorajar *design*

simples e inspirar confiança. É uma prática de programação e não apenas técnicas para teste, apesar de gerar um conjunto de testes ao final, nem sempre tais testes por si só são satisfatórios para o projeto como um todo.

3.4.1 O ciclo do TDD

TDD é definido como um simples ciclo de tarefas a serem executadas e repetidas até que o resultado obtido seja satisfatório. Apesar de existirem variações, a forma mais comum de se realizar isso é através do uso de testes unitários automatizados.

O primeiro passo é escrever o teste. A cada nova funcionalidade os testes devem ser escritos primeiro, antes de qualquer linha de código da mesma. Este teste deve falhar, caso isto não aconteça tal funcionalidade pode já estar implementada ou o teste está errado. O *test-first development* permite que o programador, neste momento, esteja completamente focado nos requisitos para se implementar essa funcionalidade, não sendo influenciado por como ele pretende implementa-la, eliminando o risco de testes que só funcionam para determinados códigos.

O segundo passo é rodar todos os testes para verificar se o novo falha. O objetivo é validar se o teste que acabou de ser escrito não passa sem que o código requerido esteja implementado, o que o caracterizaria com um teste inútil, aumentando a confiança de que este esteja testando corretamente. O terceiro passo é escrever o código que faça o novo teste passar. Nesta etapa não é necessário que o código esteja perfeito, e sim o mais simples possível. O objetivo é garantir uma versão inicial que atenda aos requisitos, mesmo que de forma pouco eficiente. Eficiência, ou qualquer outro problema que este venha a ter, serão tratados nos próximos passos. O quarto passo é novamente rodar o testes. Desta vez é esperado que eles todos passem, garantindo assim que eles realmente cumprem o objetivo.

O quinto passo é a refatoração. Nesta etapa o código precisa ser melhorado o quanto for necessário. Como nesta etapa podem surgir diversas modificações os testes escritos anteriormente se fazem fundamentais, e devem ser executados a todo tempo para auxiliar a refatoração. Testes passando indicam que suas modificações não mudaram o comportamento do código.

Os cinco passos então são repetidos para cada novo teste necessário a funcionalidade. O tamanho dos passos devem sempre ser pequenos, e os testes devem ser executados continuamente.

Os testes gerados por tal prática são muito importantes para este projeto para que possamos garantir que novas integrações não introduzam problemas ao código já existentes, principalmente porque este é aberto para contribuições da comunidade.

3.4.2 Vantagens

TDD é um processo que necessita de adaptação e foco para ser executado, portanto é preciso conhecer suas vantagens para motivar a equipe.

Primeira e talvez a principal vantagem que a prática tem a oferecer é o aumento da qualidade do seu código, e consequentemente do produto. Com o constante suporte dos testes, o Desenvolvimento Orientado a Teste oferece uma grande validação dos requisitos, pois demanda um bom foco nas especi-

ficações do produto para que seja possível escrever os testes primeiro. Além de facilitar a depuração de erros no futuro, já que a prática encoraja testes simples, permitindo a detecção de erros mais facilmente. O encorajamento em produção de testes simples, geralmente levam a produção de códigos simples e mais claros. No futuro do projeto, códigos mais legíveis e bem construídos contribuirão para manutenção, mudanças e desenvolvimento de novas funcionalidades.

Apesar do TDD precisar de mais código do que o normal, por causa dos seus testes, o tempo total da implementação de um projeto é normalmente reduzido. Com a melhora da qualidade já no momento da implementação, o número de defeitos no futuro é consideravelmente reduzido.

Toda mudança ou nova funcionalidade exige que você execute todos os testes já criados tanto referentes a tal mudança quanto os outros. Isso aumenta a confiabilidade no sistema, já que se eles estiverem passando isso garante que o sistema continua funcionando do modo esperado. Um conjunto de testes bem feito provê localização de defeitos, detecta mudança indesejadas, aumenta a confiaça do desenvolvedor, diminui os riscos. Além de permitir rápida reação a mudanças de escopo, o que confirma sua ótima utilidade em processos ágeis. Uma mudança grande no sistema pode ser feita com segurança de que se algum erro inesperado acontecer os testes irão detectá-los.

3.4.3 Desvantagens

Apesar de tantas vantagens TDD não é a solução para todos os problemas. Existem também desvantagens as quais devem ser severamente consideradas antes de mudar seu processo para utilizá-lo.

Desenvolvimento Orientado a Testes toma tempo e exige certo esforço para treinar seus desenvolvedores e preparar um bom ambiente de testes. Não é simples de ser aprendido, pode até se tornar um desafio torná-lo parte do seu dia a dia. Mesmo após se acostumar com o ciclo de tarefas, o desenvolvedor ainda precisa sempre se esforçar e se aprimorar para escrever testes de qualidade. Afinal, conjuntos de testes ruins, incompletos, lentos ou até mesmo muito grandes podem comprometer o sucesso da prática. Deve-se sempre pensar em desenvolver códigos e testes de qualidade da maneira mais simples possível.

Um dos grandes erros ao se praticar TDD é acreditar que ele força ou garante desenvolvimento de código de qualidade. Isto não acontece, praticar TDD apenas aumenta o foco em locais que necessitam de melhorias. Os desenvolvedores precisam ficar sempre atentos a qualidade do código do seu projeto e considerá-la o fator mais importante durante a fase de desenvolvimento.

Equipes novas com a prática de TDD acabam percebendo uma diminuição na velocidade de progresso do seu projeto. Isso ocorre devido ao esforço feito para escrever testes antes da funcionalidade ser implementada. O que é completamente normal, já que as recompensas de tal prática não aparecem imediatamente, e sim mais tarde no desenvolvimento. É importante ficar atento nesse quesito para que a equipe não fique desmotivada com essa falsa queda de rendimento, pois provavelmente isso será recompensado com menos *bugs* ou requisitos falhos mais a frente.

3.5 Gerência de Configuração de *Software*

"Gerência de Configuração de *Software*, também chamado gerência de mudanças, é um conjunto de atividades elaboradas para gerenciar mudanças por identificação de produtos de trabalho que são suscetíveis a mudanças, estabelecendo relações entre eles, definindo mecanismos para gerenciar diferentes versões destes produtos de trabalho, controlando mudanças impostas, e auditando e relatando a cada mudança feita."(PRESSMAN, 2010)

O *Git* [5] foi a ferramenta escolhida para a realizar a gerência de configuração do projeto. Ele é um sistema de controle de versionamento gratuito e possui código aberto. O *Git* possui um avançado sistema de ramos, permitindo a criação, junção e remoção de ramos localmente. Além de ser extremamente rápido comparado a outros sistemas, devido ao fato da maioria de suas ações serem realizadas localmente, evitando diversas comunicações com um servidor remoto. Isto é possível pois ele é distribuído, o que significa que cada usuário possui uma cópia completa e funcional do repositório, podendo até mesmo servir de *backup*.

```
[igor@MacBook-Pro-de-Igor\:$ git log
commit 1f70044862164f88067932781dd1aea296d1e5f9
commit 1f70044862164f88067932781dd1aea296d1e5f9
Author: Igor Barbosa <igorbarbosa.p@gmail.com>
Date:   Wed Sep 2 18:13:05 2015 -0300

    Fix course_id on events_strategy#format_events

commit da5e1a81324fbc5a7b3a5fa94a45e60a6b0e600a
Author: Tiago Cândido <tiago.candido@bcc.ic.uff.br>
Date:   Wed Sep 2 11:09:01 2015 -0300

    Allow Cross Origin Requests to the API

commit c8324713920b0c71cb6b34e0966821efca4a24ff
Author: Tiago Cândido <tiago.candido@bcc.ic.uff.br>
Date:   Wed Sep 2 09:19:27 2015 -0300

    Sets Conexao API to the staging URL

commit 0f1988563f990e644f1be19ef73cd9b0d661efe8
Author: Tiago Cândido <tiago.candido@bcc.ic.uff.br>
Date:   Wed Sep 2 09:06:51 2015 -0300
```

Figura 3.3: Exemplo do histórico de versões do *Git*

O repositório *Git* foi compartilhado remotamente na plataforma *Github* [9], um serviço *web* que oferece, além do repositório, uma excelente interface gráfica facilitando assim a visualização do código, novas modificações, histórico, mudanças entre revisões, métricas e etc. E também um robusto sistema de colaboração, permitindo que qualquer pessoa possa visualizar e contribuir com o código hospedado caso

este seja aberto, como é o caso deste projeto.

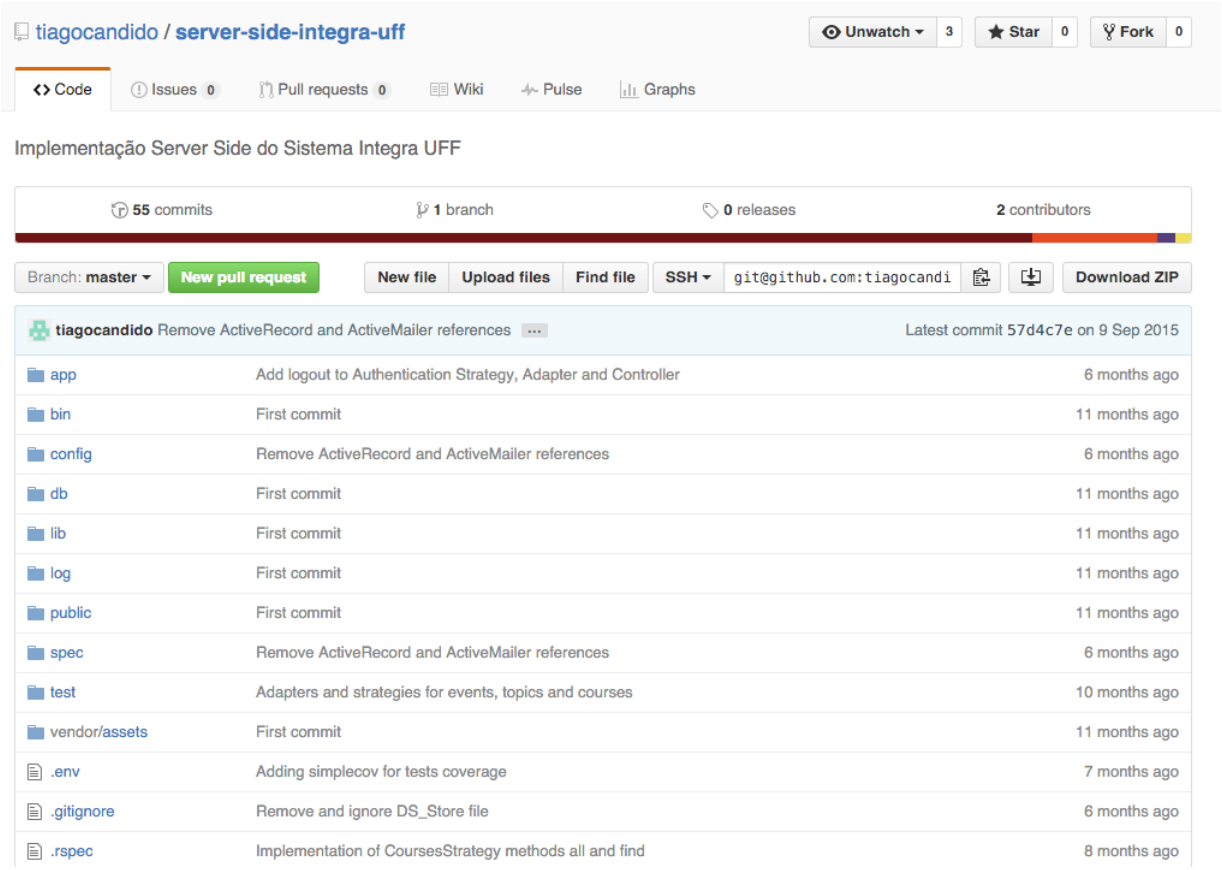

Figura 3.4: Interface do *Github*

3.6 Qualidade de Software

Garantir a qualidade do código, ou até mesmo definir o termo não é uma tarefa fácil. Existem diversas maneiras de se implementar determinadas soluções e a forma como um desenvolvedor interpreta um código como de qualidade pode diferir muito da opinião de outro desenvolvedor.

Neste projeto foram empregadas várias práticas para que a qualidade do código e do produto final em si fosse mantida ao longo do desenvolvimento. Algumas delas, as quais se inserem no contexto das metodologias ágeis, já foram citadas. Os testes automatizados garantem que os requisitos do sistema sejam atendidos e que continuem funcionando bem a cada nova adição ao código. Ademais foi identificada a necessidade de realizar algumas análises mais quantitativas em cima do código, devido a intenção de receber contribuições ao código aberto. Em vista disto foi decido rodar algumas métricas sobre ele.

3.6.1 Linhas de Código

A quantificação das linhas de código é uma das formas mais simples e antigas para se analisar um determinado *software*. Esta se dá pelo cálculo de todas as linhas não nulas e não comentadas da fonte do código, e exibida de forma geral, arquivo por arquivos, organizadas por módulos, camadas da arquitetura, ou por código de produção versus código de teste.

As linhas de código sozinhas não oferecem muita utilidade para uma análise de qualidade. Porém esta pode ser combinada com os testes automatizados para se obter uma razão entre linhas de código e linhas de teste, que podem indicar o tipo de abordagem utilizada para os testes da aplicação e se os testes são suficientes. E dependendo do tipo de forma que a quantidade de linhas forem exibidas, elas podem ser úteis para perceber módulos do *software* que estão muito grandes e possivelmente acumulando muitas responsabilidades, necessitando de melhorias.

O *framework Ruby on Rails* utilizado para o desenvolvimento deste projeto possui o comando *rake stats* que permite obter tais informações rapidamente como mostrado na figura 3.5.

```
igor@MacBook-Pro-de-Igor\:$ rake stats
+----------------------+-------+-------+---------+---------+-----+-------+
| Name                 | Lines |   LOC | Classes | Methods | M/C | LOC/M |
+----------------------+-------+-------+---------+---------+-----+-------+
| Controllers          |    65 |    50 |       7 |       9 |   1 |     3 |
| Helpers              |     2 |     2 |       0 |       0 |   0 |     0 |
| Models               |   364 |   305 |      12 |      45 |   3 |     4 |
| Mailers              |     0 |     0 |       0 |       0 |   0 |     0 |
| Javascripts          |    16 |     0 |       0 |       0 |   0 |     0 |
| Libraries            |     0 |     0 |       0 |       0 |   0 |     0 |
| Controller tests     |     0 |     0 |       0 |       0 |   0 |     0 |
| Helper tests         |     0 |     0 |       0 |       0 |   0 |     0 |
| Model tests          |    26 |    20 |       2 |       0 |   0 |     0 |
| Mailer tests         |     0 |     0 |       0 |       0 |   0 |     0 |
| Integration tests    |     0 |     0 |       0 |       0 |   0 |     0 |
| Controller specs     |   150 |   120 |       0 |       0 |   0 |     0 |
| Model specs          |    99 |    81 |       0 |       0 |   0 |     0 |
+----------------------+-------+-------+---------+---------+-----+-------+
| Total                |   722 |   578 |      21 |      54 |   2 |     8 |
+----------------------+-------+-------+---------+---------+-----+-------+
```

Figura 3.5: Informações de linhas de código do projeto

3.6.2 Complexidade

Para obter números acerca da complexidade do código foram utilizadas duas métricas em conjunto:

- Complexidade ciclomática - Definida por Thomas McCabe, por isso também chamada de complexidade de McCabe, ela é uma conta que mede a quantidade de caminhos de execução independentes a partir de um código fonte. De acordo com McCabe (1976), ela foi concebida para se adequar a nossa noção intuitiva de complexidade.

- Métrica ABC - Esta métrica agrega o número de atribuições, ramos e condições (*assignments, branches* e *conditionals*) de uma unidade do código. Sendo a parte da avaliação dos ramos muito similar a complexidade ciclomática. Ela foi pensada para ser aplicada em diversos tipos de linguagem e estilos, o que permite comparar as pontuações geradas por ela entre diferentes bases de código. Embora seja provavelmente ineficaz em linguagem não imperativas.

3.6.3 *Churn*

A métrica *Churn* mensura o número de vezes que um arquivo do seu código fonte mudou ao longo do tempo, calculado pelo número de versões que aquele arquivo possui no histórico do seu controle de versionamento. Como a métrica de linhas de código o número absoluto de *Churn* de um arquivo sozinho não é muito útil. Este se torna valioso quando combinado com a métrica de complexidade como mostrado no gráfico da figura 3.6, auxiliando a equipe a perceber quais os arquivos causam maiores impactos ao desenvolvimento.

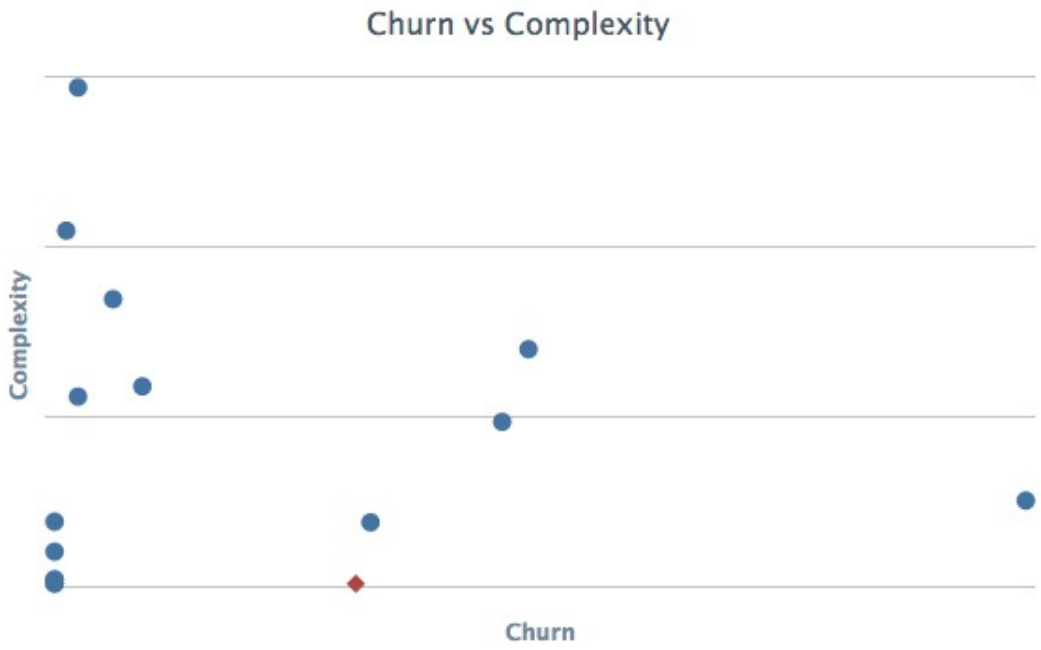

Figura 3.6: Gráfico *Churn* vs Complexidade

O gráfico apresenta quatro quadrantes que permitem as seguintes análises:

1. Superior direito — Arquivos com alta complexidade e muitas modificações. Arquivos neste quadrante são bons candidatos a serem refatorados, já que a sua manutenção é frequente e a complexidade alta impacta os desenvolvedores negativamente e regularmente.

2. Superior esquerdo — Alta complexidade mas não muita mudança. Lidar com arquivos deste quadrante é um pouco mais complexo já que o custo da sua manutenção é alto devido a sua complexidade, contudo ele é pouco modificado indicando que seu impacto no desenvolvimento não é frequente.

3. Inferior esquerdo — Baixa complexidade e baixo *Churn*. O quadrante que deve abrigar a maior parte de seus arquivos, os quais dificilmente causarão problemas durante o desenvolvimento.

4. Inferior direito — Baixa complexidade e alto *Churn*. Arquivos que precisam ser mudados frequentemente, como por exemplo arquivos de configuração, mas não há muito o que melhorar neles.

3.6.4 Duplicidade

A análise de duplicação busca identificar códigos idênticos ou similares. Ele é executado buscando árvores com sintaxes idênticas e também usa a técnica de *fuzzy matching* para detectar código que se diferenciam somente por identificadores ou constantes específicas. Trechos de código duplicados geralmente podem ser extraídos para um único local, propiciando uma manutenção mais simples deles no futuro.

3.6.5 *Code Climate*

Após a definição das métricas que seriam utilizadas, foi fundamental a escolha de uma ferramenta que amparasse a execução das mesmas sobre um código fonte em constante evolução. Tal tarefa foi atribuida ao serviço *Code Climate* [6].

Code Climate se define como uma plataforma para análises estáticas que ajuda a garantir seus padrões de qualidade a todo novo *commit* feito. Ele consolida o resultados de uma game de análises em um único relatório, provendo informações essenciais que time de desenvolvimento precisa para identificar os pontos mais críticos que precisam de melhorias, avaliar novas abordagens, e melhorar a qualidade do produto.

O serviço também dá uma nota para cada um dos seus arquivos de A, a melhor nota, até F. Tais notas são calculadas pela soma de todos os problemas encontrados durante a análise daquele arquivo e os transformando em uma escala absoluta. Desta forma se torna fácil identificar quais são os pontos que precisam de maior atenção. A figura 3.7 demonstra a interface que informa o histórico deste projeto, no qual se pode observar informações de arquivos que melhoraram ou pioraram ao longo do tempo.

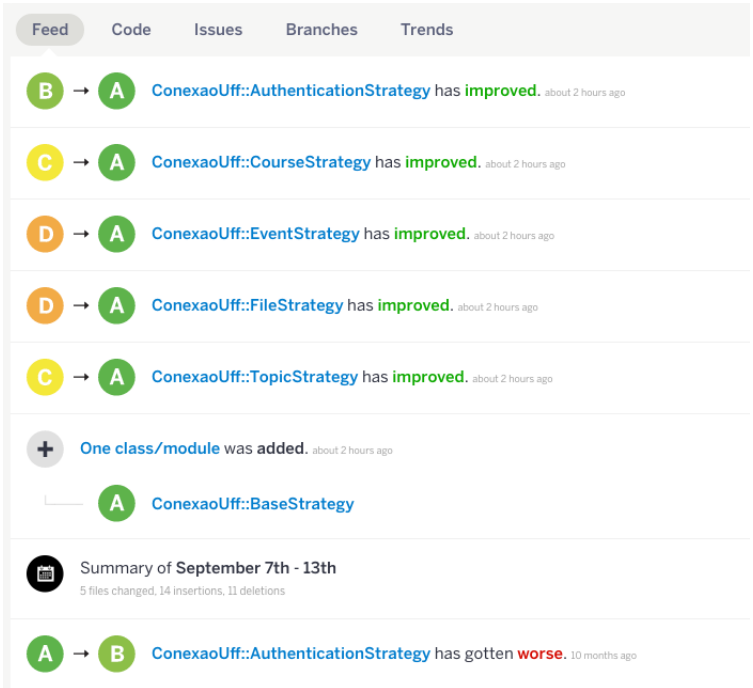

Figura 3.7: *Feed* do projeto no *Code Climate*

A plataforma ainda conta com uma avaliação geral, chamada de *grade point average* (GPA). Esta

avaliação é gerada pela média das notas de todos os arquivos analisados, atribuindo um peso diferente a cada um deles dependendo da quantidade de linhas de código. A GPA varia de zero a quatro, sendo quatro a melhor avaliação possível.

A interface do serviço oferece alguns gráficos interessantes para tornar mais simples o gerenciamento do repositório. Como mostrado na figura 3.8, contém dados do IntegraUFF, ele informa o GPA ao longo do tempo, o *Breakdown over time*, o qual utiliza um código de cores para identificar as notas individuais de cada arquivo e o *Churn vs Quality*, já explicado anteriormente.

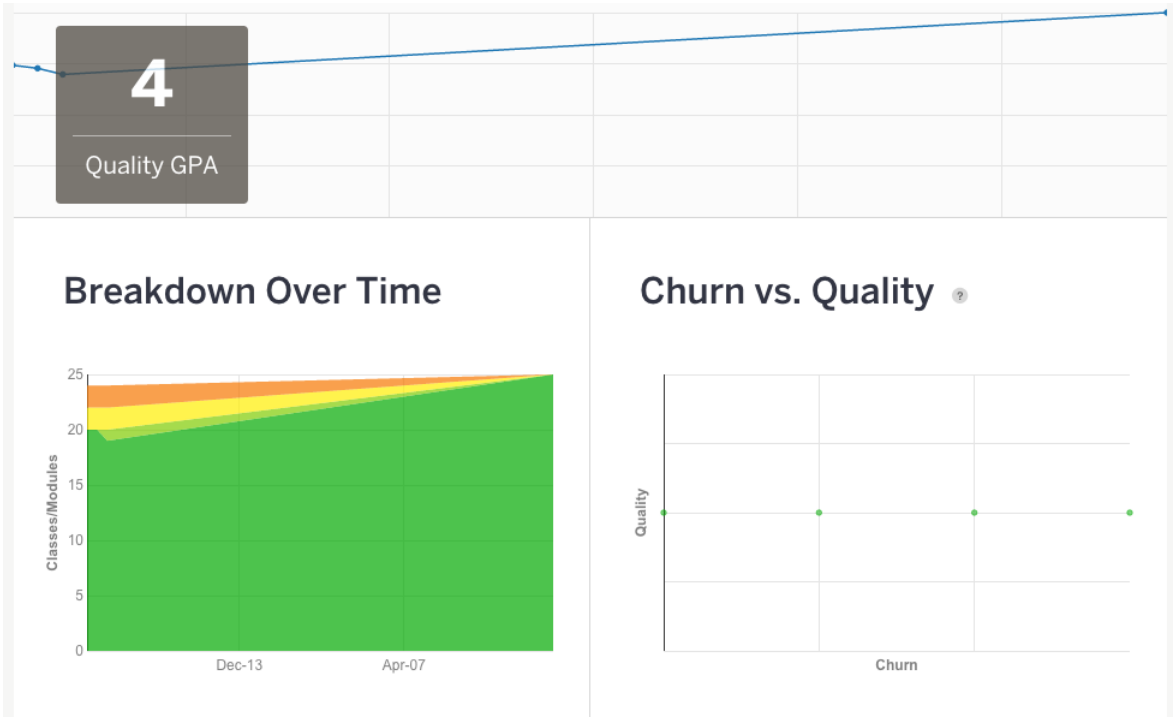

Figura 3.8: Gráficos gerados pelo *Code Climate*

Capítulo 4

Integra UFF - Especificação de Software

Este capítulo apresenta a especificação de software do Integra UFF. Aqui serão descritos os objetivos, ambientes, suposições, requisitos e interfaces pensadas durante o desenvolvimento da mesma. Assim como as funcionalidades e telas implementadas no produto final.

4.1 Descrição Geral

4.1.1 Perspectiva do produto

O produto implementa uma interface que agrega conteúdos obtidos de outros sistemas LMSs já existentes estendendo suas funcionalidades. Ele consiste de uma aplicação móvel cliente e uma aplicação web remota que acessa as interfaces públicas dos sistemas terceiros. A aplicação web oferece uma *Application programming interface* (API) que atende os pedidos da aplicação móvel, intermediando o acesso aos sistemas externos. A aplicação web contará com uma base de dados auxiliar, para controle dos usuários e comunicação instantânea.

4.1.2 Funções do Produto

A aplicação deve ser capaz permitir a autenticação do usuário nos sistemas integrados, obtendo o conteúdo necessário para fornecer as principais funcionalidades contempladas para o Integra UFF, que são:

- Disponibilização de uma listagem das disciplinas do usuário.
- Tópicos (Grupo de Discussão).
- Listagem e *download* de Arquivos.
- Calendário de Eventos das Disciplinas.
- Envio de Mensagens.

- Lista de Amigos.

4.1.3 Classes de Usuário e Caracterísiticas

As classes de usuários previstas serão, o aluno participante da disciplina, o professor da disciplina no papel de moderador do grupo, e os administradores do sistema que exercerão funções de manutenção do sistema.

4.1.4 Ambiente Operacional

A interface do usuário será executada em dispositivos móveis como *smartphones* e *tablets*, nos três sistemas operacionais mais utilizados, sendo eles o iOS da fabricante *Apple*, o *Android* da *Google* e o *Windows Phone* 8 da *Microsoft*. A aplicação remota será executada em uma máquina servidora com sistema operacional Linux, distribuição Ubuntu, com servidor *web* Apache ou Nginx com o módulo *passenger-phusion* devidamente configurado para servir aplicações *Ruby on Rails*, todos em suas versões mais recentes.

4.1.5 Restrições de *Design* e Implementação

Por se tratar de um trabalho acadêmico, não haverão restrições de *design* e implementação rígidas, deixando os desenvolvedores juntamente ao orientador livres para escolher as tecnologias verificadas e tidas como adequadas para a solução. No entanto toda a integração feita com os sistemas LMSs são limitadas a disponibilidade de uma API acessível pela *web* e ao conteúdo fornecido por elas.

4.1.6 Suposições e Dependências

O sistema tem como dependências a plataforma de desenvolvimento móvel *Apache Cordova*, o *framework* de desenvolvimento ágil *Rails*, o *framework* AngularJS, o *framework* de CSS *Twitter Bootstrap* e outros *plugins* indentificados no decorrer do desenvolvimento.

4.2 Requisitos de Interface Externa

4.2.1 Interfaces de Usuário

As interfaces de usuário serão desenvolvidas com as tecnologias e padrões abertos *web*, *Javascript*, CSS3 e HTML5 e serão responsivas, se ajustando a diversos tipos de dispositivos e tamanhos de tela.

4.2.2 Interfaces de Hardware

O *software* será executado em dispositivos móveis, *smartphones* e *tablets*, do tipo *Android*, *Windows* e *iPhone* que se comunicará com os serviços de uma aplicação intermediária, executada em um computador servidor *web*, que por sua vez se comunica com as APIs das diversas plataformas de ensino integradas.

4.2.3 Interfaces de Software

O *software* executado no dispositivo móvel, utiliza o empasulamento de uma aplicação *web* para as diversas plataformas móveis através do *framework phonegap*, de forma totalmente transparente ao usuário que utiliza o *software* como um aplicativo nativo. Esta *webapp* se comunica com a aplicação servidora intermediária através de recursos *web* executados no *software* Apache ou *Ngnx* com o *framework Rails*. A aplicação intermediária por sua vez utiliza as intefaces disponibilizadas pelas plataformas de ensino, que podem ser diversas e implementadas sob a forma de *plugins*.

4.2.4 Interfaces de Comunicação

A comunicação é totalmente via *web*, utilizando o protocolo HTTP entre a aplicação cliente, a aplicação servidor e as APIs das plataformas de ensino, seguindo o estilo arqutetural REST entre a a aplicação cliente e a camada intermediária.

4.3 Funcionalidades do Sistema

Nesta seção descreveremos as funcionalidades implementadas na versão final do aplicativo Integra UFF, assim como suas telas.

4.3.1 Sincronizar com uma plataforma

O primeiro passo para começar a utilizar o Integra UFF é realizar a sincronização com uma das plataformas integradas. Atualmente apenas o Conexão UFF está disponível.

Ao acessar o aplicativo o usuário verá uma tela com um botão de sincronização, que ao apertado exibirá um formulário de autenticação, figura 4.1. Feito a autenticação, o usuário será redirecionado para o menu principal do aplicativo, figura 4.2, com o acesso a Disciplinas, Eventos e Arquivos.

O caso de uso da sincronização está explicitado na tabela 4.1.

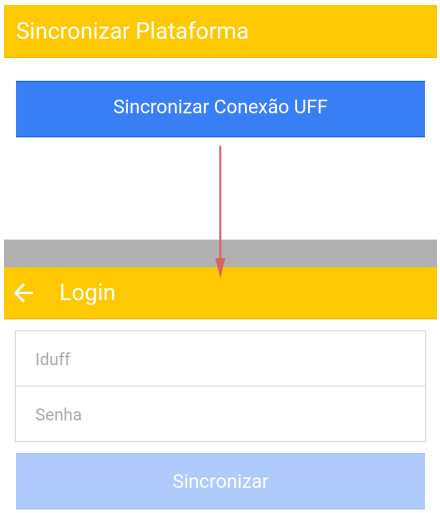

Figura 4.1: Autenticação na plataforma

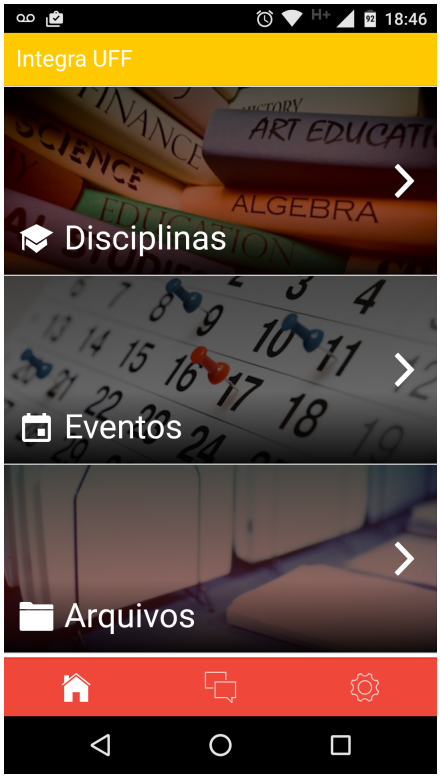

Figura 4.2: Menu do Integra UFF

Trigger	O usuário acessa a aplicação
Pré condição	O usuário não sincronizou com nenhuma das aplicações anteriormente. O aparelho deve estar conectado à internet.
Caminho Básico	1. O usuário escolhe a qual rede deseja se conectar selecionando com o toque na tela em um item dentre uma lista de opções. 2. O sistema exibe um formulário de login com um campo para o e-mail e outro para a senha. 3. O usuário preenche seus dados e seleciona sincronizar. 4. O sistema sincroniza com a aplicação escolhida e exibe a tela principal com as opções Disciplinas, Eventos, Arquivos e Configurações.
Caminho alternativo	Se o usuário já sincronizou com alguma aplicação anteriormente ele deve seguir os seguintes passos: 1. O usuário seleciona Configurações na tela principal. 2. O usuário seleciona Contas Sincronizadas na tela de configurações. 3. O sistema exibe a tela com as opções de aplicações disponíveis para sincronização. 4. A partir daqui o usuário segue o caminho básico.
Pós condição	Informações da aplicação sincronizada devem ser baixadas para o aparelho.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento. Caso o usuário erre o login o sistema deve exibir a tela de login informando o erro.

Tabela 4.1: Sincronizar com uma aplicação

4.3.2 Acessar listagem de disciplinas

As disciplinas representam o conteúdo principal da aplicação. Todos os outros conteúdos estão associados a ela. Portanto se fez necessário exibir uma listagem delas para o usuário, a qual é acessada pelo menu principal demonstrado anteriormente.

A listagem possui uma interface simples que informa o nome das disciplinas associadas e o semestre a qual ela pertence como mostra a figura 4.3.

Figura 4.3: Listagem de Disciplinas

Ao clicar no nome de uma das disciplinas da lista, o usuário será redirecionado para uma seção contendo os detalhes da mesma.

O caso de uso desta funcionalidade é mostrado na tabela 4.2.

Trigger	O usuário acessa a aplicação.
Pré condição	O usuário está na tela do menu principal do sistema.
Caminho Básico	1. O usuário seleciona a opção disciplinas no menu do sistema. 2. O sistema exibe uma tela com uma lista de todas as disciplinas sincronizadas com o aparelho. A listagem contém o nome da disciplina.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.2: Acessar listagem de disciplinas

4.3.3 Acessar detalhes de disciplinas

Os detalhes de cada disciplina podem ser acessados através do clique em seu nome exibido na listagem. Nesta tela serão exibidas uma lista de tópicos, eventos e arquivos associados àquela disciplina, conforme a figura 4.4, viabilizando uma forma simples de visualizar todos os conteúdos de uma disciplina específica.

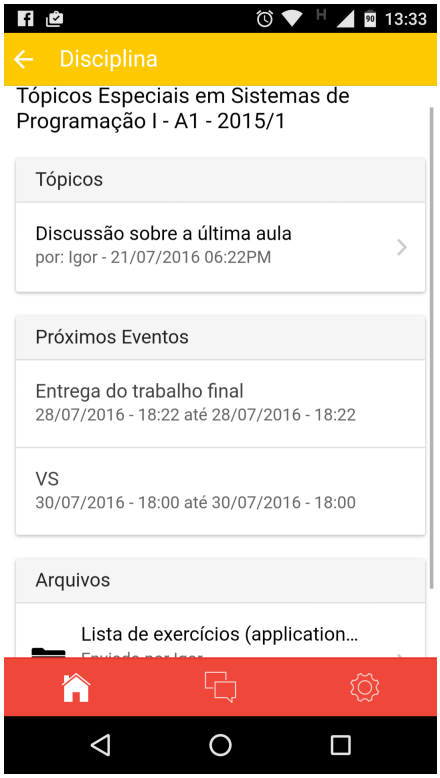

Figura 4.4: Detalhes de uma Disciplina

O clique no nome de um arquivo ou tópico redireciona o usuário para a página de detalhes dos mesmos.

O caso de uso desta funcionalidade é mostrado na tabela 4.3.

Trigger	O usuário acessou a tela de listagem de disciplinas.
Pré condição	O usuário possui alguma disciplina sincronizada com o aparelho.
Caminho Básico	1. O usuário seleciona uma das disciplinas da listagem. 2. O sitema exibe uma tela contendo as seguintes informações sobre a disciplina: nome, últimos tópicos, próximos eventos e últimos arquivos.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.3: Acessar detalhes de disciplinas

4.3.4 Acessar listagem de arquivos

Apesar de ser possível acessar a lista de arquivos de uma disciplina na tela de listagem da última, também está disponível uma listagem de arquivos sincronizados através do clique na opção Arquivos no menu principal da aplicação. Esta listagem demonstra todos os arquivos sicronizados, agrupados pelas disciplinas a qual pertencem, como mostra a figura 4.5

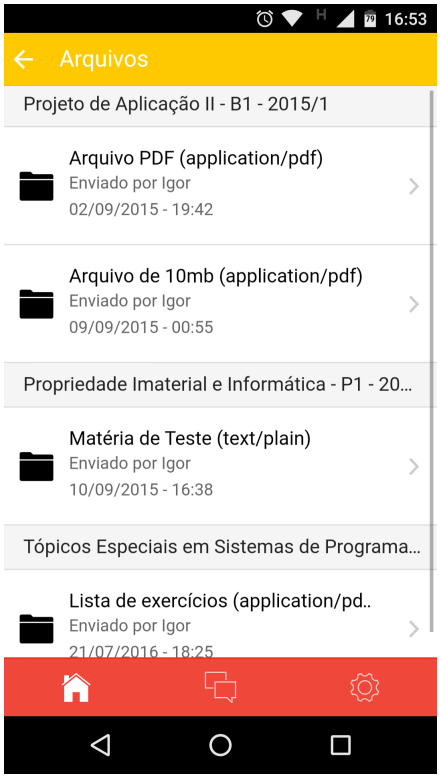

Figura 4.5: Listagem de Arquivos

Clicando em qualquer um dos arquivos o usuário será redirecionado para a página de detalhes deste. A tabela 4.4 apresenta o caso de uso desta funcionalidade.

Trigger	O usuário acessa a aplicação.
Pré condição	O usuário possui alguma disciplina sincronizada com o aparelho.
Caminho Básico	1. O usuário seleciona a opção arquivos no menu do sistema. 2. O sistema exibe uma tela com uma lista de todos os arquivos sincronizados com o aparelho. A listagem contém as seguintes informações dos arquivos: nome, formato, quem o enviou, data e hora de envio. Os arquivos exibidos são organizados por disciplinas.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.4: Acessar listagem de arquivos

4.3.5 Acessar detalhes de arquivos

Conforme as citações anteriores a página de detalhes de arquivos pode ser acessada através da listagem geral de arquivos, ou da listagem de arquivos na tela de detalhes de uma disciplina. Esta tela contém o nome do arquivo e informações do formato, tamanho, data e hora em que foi enviado e o nome de quem o enviou, visto na figura 4.6. Além de possuir um botão para realizar o *download* para o dispositivo.

Figura 4.6: Detalhes de um Arquivo

Caso o arquivo já tenha sido baixado, a tela irá exibir um botão para abri-lo e um *link* para apagá-lo.

A tabela 4.5 evidencia o caso de uso.

Trigger	O usuário acessou a tela de listagem de arquivos.
Pré condição	O usuário possui algum arquivo sincronizado com o aparelho.
Caminho Básico	1. O usuário seleciona um dos arquivos da listagem. 2. O sitema exibe uma tela contendo as seguintes informações sobre o arquivo: nome, formato, tamanho, quem o enviou, data e hora do envio, um botão para baixá-lo.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.5: Acessar detalhes de arquivos

4.3.6 Baixar arquivos

Baixar os arquivos é possível através de um botão na tela de detalhes de um arquivo. O procedimento é demonstrado pela figura 4.7 e o caso de uso pela tabela 4.6.

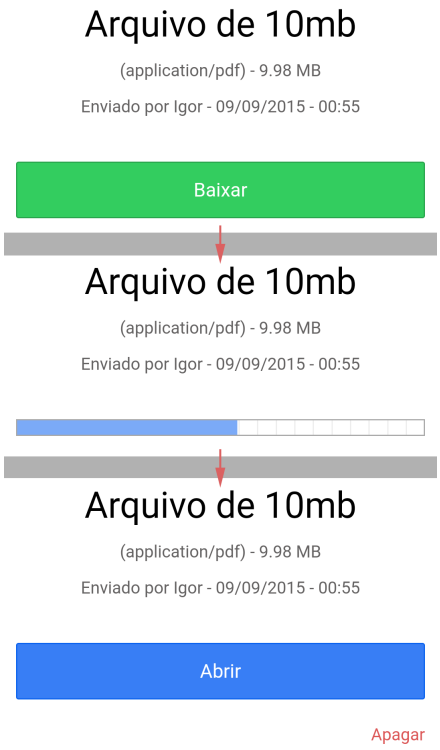

Figura 4.7: *Download* de um Arquivo

Trigger	O usuário acessa a página de detalhes de um arquivo.
Pré condição	O usuário possui algum arquivo sincronizado com o aparelho. O aparelho deve estar conectado à internet.
Caminho Básico	1. O usuário aperta o botão download. 2. O sistema inicia o download do arquivo selecionado.
Pós condição	O arquivo selecionado deve ser baixado para o aparelho do usuário e o sistema deve mostrar os botões para abrir e apagar o arquivo.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.6: Baixar arquivos

4.3.7 Apagar arquivos

Apagar os arquivos, assim como o seu *download*, é realizado através de um *link* na tela de detalhes de um arquivo. O procedimento é demonstrado pela figura 4.8 e o caso de uso pela tabela 4.7.

Figura 4.8: Apagar um Arquivo

Trigger	O usuário acessa a página de detalhes de um arquivo.
Pré condição	O usuário possui algum arquivo baixado no aparelho.
Caminho Básico	1. O usuário aperta o botão apagar.
Pós condição	O arquivo selecionado deve ser apagado do aparelho do usuário e o sistema deve exibir o botão para baixar o arquivo.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.7: Apagar arquivos

4.3.8 Acessar listagem de eventos

A listagem de eventos reúne todos os eventos sincronizados no dispositivo. Os eventos são agrupados por próximos, eventos prestes a acontecerem, e encerrados como mostra a figura 4.9. Esta listagem também informa a qual disciplina cada evento pertence.

Figura 4.9: Listagem de Eventos

O caso de uso pode ser observado na tabela 4.8.

Trigger	O usuário acessa a aplicação.
Pré condição	O usuário está na tela do menu principal do sistema.
Caminho Básico	1. O usuário seleciona a opção eventos no menu do sistema. 2. O sistema exibe uma tela com uma lista de todos os eventos sincronizados com o aparelho. A listagem contém as seguintes informações dos eventos: nome, data, hora e disciplina. Os eventos são exibidos separadamente entre próximos eventos e eventos passados.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.8: Acessar listagem de eventos

4.3.9 Adicionar evento ao calendário

Os eventos podem ser adicionados no calendário padrão do dispositivo utilizado pelo usuário. Para isso é preciso acessar a listagem dos eventos e clicar no ícone ao lado direito do evento escolhido. Uma caixa de diálogo será exibida confirmando a operação. A figura 4.10 ilustra o procedimento, e o caso de uso é descrito pela tabela 4.9.

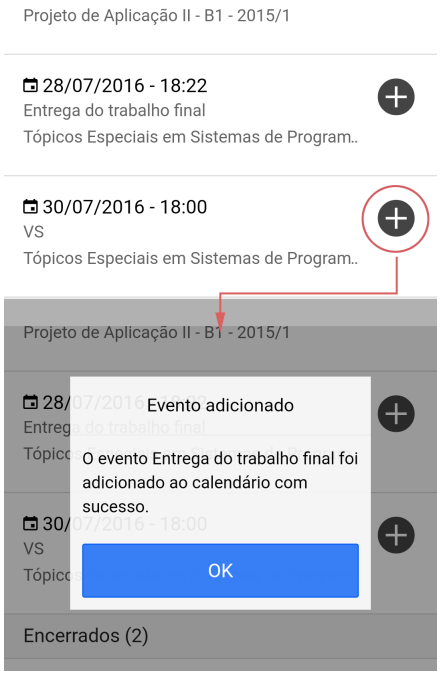

Figura 4.10: Adicionar evento ao calendário

Trigger	O usuário acessa a página de listagem de eventos.
Pré condição	O usuário possui algum evento sincronizado no aparelho.
Caminho Básico	1. O usuário seleciona o ícone de adicionar situado ao lado do evento que deseja colocar no calendário.
Pós condição	O evento selecionado deve ser criado no calendário nativo do aparelho do usuário.
Caminho de exceção	O usuário pode abandonar a operação a qualquer momento.

Tabela 4.9: Adicionar evento ao calendário

4.4 Outros requisitos não funcionais

4.4.1 Requisitos de segurança

Para a utilização do sistema o usuário precisará sincronizar a aplicação com as plataformas disponíveis. Para tal será necessário uma autenticação, esta será feita pelas próprias plataformas integradas ao sistema.

4.4.2 Atributos de Qualidade de Software

O sistema precisa possuir uma arquitetura que permita uma fácil extensão através do desenvolvimento de plugins. Também deve possuir o código aberto e contar com testes automatizados que garantem que os requisitos sejam atendidos e continuem funcionando a cada nova contribuição ao código fonte. Métricas devem ser rodadas a cada nova revisão para que a qualidade final do produto seja mantida.

Capítulo 5

Arquitetura de Software

A implementação do projeto ocorreu em um contexto cliente-servidor, ou seja, foram desenvolvidas duas aplicações. A primeira, realizada no *server side*, ou lado do servidor, consistiu no desenvolvimento de um sistema *web* que funciona como uma API. Este sistema tem como objetivo receber requisições da aplicação implementada no *client-side*, ou lado do cliente, e se comunicar com as API's das LMS's, obtendo as informações referentes ao usuário, como por exemplo as suas disciplinas. O sistema também é responsável por adaptar tais informações para o formato suportado pela aplicação cliente antes de enviá-las. Já a aplicação do lado do cliente foi desenvolvida como um aplicativo móvel, e tem como responsabilidade prover uma interface simples de usar e que reúna e apresente todas as informações dos sistemas LMSs integrados em um único local.

5.1 *Server-Side*

Para o desenvolvimento da aplicação do servidor foi utilizada a linguagem de programação *Ruby* (MATSUMOTO, 1995), orientada a objetos, interpretada e com foco na produtividade e simplicidade além de ser totalmente livre.

Em conjunto foi o utilizado o *Ruby On Rails*, um *framework web*. Criado com *Ruby*, seu objetivo é tornar o desenvolvimento *web* o mais simples possível. Trazendo consigo todo o necessário para construir uma aplicação moderna. Ele possui as seguintes características:

- COC (*Convention Over Configuration*, ou Convencao Sobre Configuracao): um paradigma que busca determinar um padrao de regras e organizacao em busca de diminuir o número de decisões e configurações que o desenvolvedor precisa realizar para iniciar o desenvolvimento.
- RESTFul (*Representational State Transfer*, ou Transferencia do Estado Representacional):de acordo com Fielding (2000), consiste no uso de identificadores de recurso (URL) e a mudança de estados de um objeto utilizando metodos do protocolo HTTP, como os metodos *GET*, *POST*, *PUT* e *DELETE*.

A estrutura de uma aplicação Rails é o padrão MVC (Modelo-Visão-Controlador) que organiza a lógica de programação em três camadas principais. O modelo, no qual colocamos a lógica de negócio. O

controlador que recebe as interações ou requisições, direciona comandos ao modelo para enfim construir uma resposta HTML ou JSON como a visão.

Considerando que novos sistemas de gerenciamento de ensino poderiam ser adicionados futuramente, tais integrações foram organizadas em uma estrutura de módulos. Na qual todo o código e lógica envolvida na integração de uma LMS específica foi encapsulado em um módulo isolado.

Contudo, ainda foi necessário a implementação de dois padrões de projeto adicionais na aplicação. Os padrões *Adapter* (GAMMA et al., 1994) e *Strategy* (GAMMA et al., 1994) se mostraram muito úteis para que fosse possível alcançar os objetivos do sistema.

5.1.1 Padrão *Adapter*

A intenção do padrão *Adapter*, também conhecido como *Wrapper*, é converter uma interface para outra interface diferente a qual é esperada pelo cliente. Ou seja, ele permite a comunicação de interfaces incompatíveis.

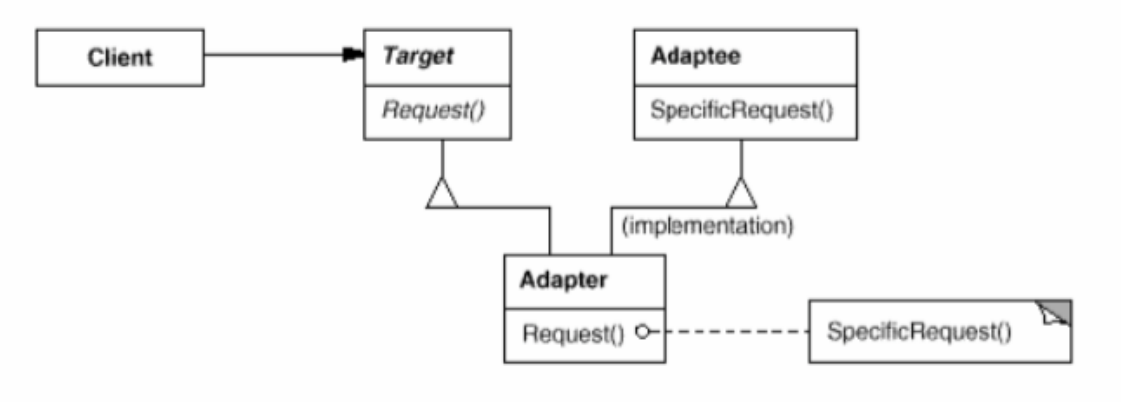

Figura 5.1: Padrão *Adapter*

O padrão é importante para o projeto pois é necessário adaptar as diversas respostas obtidas das diferentes interfaces de cada uma das LMSs integradas ao sistema, para que seja possível que a interface cliente os utilize. Sendo assim, foram implementados um *adapter* para cada recurso diferente previsto na aplicação.

- *CourseAdapter*: para disciplinas e cursos;
- *EventAdapter*: para eventos;
- *FileAdapter*: para arquivos;
- *TopicAdapter*: para tópicos;
- *AuthenticationAdapter*: responsável pela autenticação em cada um dos serviços integrados;

Sozinho este padrão não solucionava todo o problema. Como é preciso lidar com diversas interfaces diferentes, uma para cada LMS, tais classes precisariam ser modificadas toda vez que um sistema

LMS novo fosse integrado. Pensando nisso, os *adapters* foram combinados com um outro padrão, o *Strategy*, que passa a cuidar da lógica de adaptação. Os *adapters* recebem como parâmetro a LMS e define a *strategy* correta para adaptação.

5.1.2 Padrão *Strategy*

A intenção do padrão *Strategy*, também conhecido como *Policy*, é definir um grupo de algorítimos, encapsular cada um deles, e permitir que eles sejam permutáveis. Ou seja, permite que a lógica varie dependendo do cliente que precisa utilizá-la.

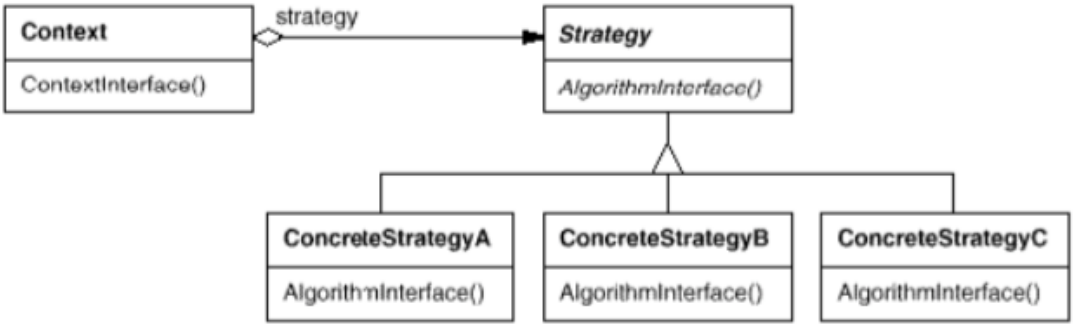

Figura 5.2: Padrão *Strategy*

As *strategies* do projeto foram idealizadas para serem implementadas isoladamente em cada módulo dos serviços integrados. E assim as classes *adapters* poderíam trocar as estratégias de adaptação das interfaces baseado na chamada do cliente. Por isso, para cada um dos módulos foi preciso implementar uma *policy* para cada recurso, da mesma forma que os adaptadores.

- *CourseStrategy*: para disciplinas e cursos;
- *EventStrategy*: para eventos;
- *FileStrategy*: para arquivos;
- *TopicStrategy*: para tópicos;
- *AuthenticationStrategy*: para a autenticação;

5.2 Client-side

A aplicação Client-Side, isto é, a parte que será executada no dispositivo móvel do usuário, tem a função de apresentar de forma clara e objetiva as informações trazidas do *Server-Side* através de uma interface de usuário que seja responsiva aos estímulos particulares utilizados em dispovitivos móveis, como o toque na tela, o arraste de objetos e a rolagem do conteúdo com gestos de ”puxar”e ”empurrar”dos dedos. Além

disso, a disposição do conteúdo deve levar em consideração o tamanho reduzido das telas dos dispositivos, os botões e teclas físicas disponíveis nos diversos tipos de aparelhos, bem como a disponibilidade e o meio de conectividade com a Internet.

Quando se fala de desenvolvimento de aplicativos para plataformas móveis, certamente um dos maiores desafios é lidar com a heterogeneidade em diversos aspectos dos dispositivos alvo. A começar pelo Hardware, temos telas que variam de 3 a 11 polegadas, com resoluções de imagem e densidades de pixels diversas, há também uma vasta gama de tecnologias de rede em padrões sortidos, com grandes variações nas taxas de transferência de dados, partindo dos 14kbps de conexões GPRS até taxas superiores a 20Mbps em tecnologias como LTE e WiMax. Existem ainda muitos outros fatores importantes, como a capacidade e o tipo de mídia de armazenamento disponível, a quantidade e disposição dos botões físicos e funções correspondentes dentre os distintos fabricantes, a infindável variedade de hardwares de geolocalizadores, acelerômetros, bússulas, câmeras, entre outros.

A julgar pela diversidade apresentada no hardware já se pode constatar a necessidade de um ferramental próprio para lidar com este desafio. No entanto, divergências mais impactantes são apresentadas quando olhamos a fundo as possibilidades existentes na camada de Software dos dispositivos. Analisando apenas as possibilidades de sistemas operacionais, constatamos a existência de uma grande quantidade de opções, sendo as principais o Android, o iOS, o Windows Phone [42], além de outras distribuições menos representativas, ou que já perderam grande parte do seu mercado ao longo da última década, como Symbian, Blackberry e WebOS.

Cada um dos Sistemas Operacionais traz junto consigo uma série de particularidades que devem ser consideradas ao ser escolhida como alvo de um projeto de aplicativo. De todas essas particularidades, o tamanho da comunidade de usuários que será contemplada certamente exerce grande influência na escolha do Sistema Operacional que executará o aplicativo. Outro grande fator a ser considerado é a plataforma de desenvolvimento associada, cada qual com suas respectivas linguagens e ambientes de programação de diferentes custos de propriedade, suas respectivas lojas, custos e regras de distribuição impostos pelos seus fabricantes.

Foi necessário uma análise entre os tipos de aplicativo móvel possíveis de ser desenvolvidos para contornar o problema da alta heterogeneidade dos dispositivos de forma que fosse possível oferecer, com custo de desenvolvimento e qualidade satisfatórios, o aplicativo para a maior parcela possível de usuários dentro da comunidade alvo do projeto. Abordaremos estes diferentes tipos de aplicativo e exploraremos suas vantagens e desvantagens na sessão seguinte e apresentaremos a solução encontrada para o problema relatado.

5.2.1 Abordagens de desenvolvimento de Aplicativo Móvel

- Aplicativos Nativos - São específicos para uma determinada plataforma móvel, como iOS ou Android, são construídos diretamente sobre os serviços disponibilizados pela plataforma móvel, expostos através de um conjunto de *Interfaces de Programação de Aplicação (API)* [34]. Utilizam as ferramentas de desenvolvimento e linguagem específicas que as respectivas plataformas suportam [39]. Por exemplo, Xcode e Objective-C são suportadas pelo fabricante para desenvolver para iOS,

e o Eclipse e a linguagem de programação Java são recomendados para o desenvolvimento para Android. Aplicativos nativos acessam geralmente executam de forma mais fluída por ter melhor desempenho no acesso às funções nativas do dispositivo [33]. Podem trazer uma sensação de usabilidade mais prazerosa, especialmente para requisitos de alto desempenho, como aceleração gráfica por hardware e leitura de sensores em sistemas de tempo real.

- Aplicativos Web ou Webapps: Usam as tecnologias web padrão, HTML5, JavaScript e CSS [43]. Nesta abordagem escreve-se o código uma única vez e o aplicativo pode ser executado em qualquer plataforma. Apesar dos desenvolvedores poderem criar aplicativos sofisticados com apenas HTML5 e JavaScript, existem algumas limitações vitais [43][44], especificamente o armazenamento offline seguro e o acesso às funcionalidades nativas dos dispositivos, como câmera, calendário, contatos, sensores, hardware de geolocalização, etc.

- Aplicativos Híbridos: Esta proposta torna possível incorporar Aplicativos Web dentro de uma fina camada recipiente nativa, combinando os elementos dos Aplicativos nativos e Webapps. A maior parte do aplicativo é construído usando tecnologias web compatíveis com todas plataformas, tais como HTML5, CSS e Javascript - as mesmas línguas usadas para escrever aplicativos web. No entanto, algum código nativo é utilizado para permitir que o aplicativo acesse funcionalidades mais específicas do dispositivo e produza uma experiência de usuário mais refinada. A vantagem desta abordagem é clara: apenas uma porção do código nativo tem de ser re-escrito para o aplicativo funcionar adequadamente sobre os diferentes tipos de dispositivos disponíveis [41], se tornando mais rápido e mais fácil de desenvolver que aplicativos nativos [31].

Tabela 5.1: Vantagens das abordagens de desenvolvimento de aplicativos móveis [46]

Web	• Capacidade de construir uma única vez e implantar todas plataformas • Distribuição Centralizada
Híbrido	• Capacidade de construir uma única vez e implantar todas plataformas • Custos de desenvolvimento e manutenção reduzidos • Tempo de lançamento reduzido Segurança e gestão do aplicativo aprimorados Desenvolvimento por pessoas com a mesma experiência (HTML5, CSS3, JavaScript) • Acesso as APIs Nativas Distribuição por Loja de Aplicativos • Processamento impulsionado pelo hardware do dispositivo
Nativo	• Melhor desempenho • Rápida adaptação para mudanças no sistema operacional • Controle total do dispositivoSegurança e Gestão de Aplicativos Aprimorados • Acesso as APIs Nativas • Distribuição por Loja de Aplicativos • Processamento impulsionado pelo hardware do dispositivo

Tabela 5.2: Desvantagens das abordagens de desenvolvimento de aplicativos móveis [46]

Web	• Sem acesso as APIs nativas • Sem modo Offline • Processamento não pode ser impulsionado pelo hardware do dispositivo
Híbrido	• Funções nativas específicas podem estar ausentes em algumas plataformas e exigir codificação nativa • Adaptação lenta a mudanças do sistema operacional • Inadequado para requisitos de alta performance
Nativo	• Cada plataforma requer o desenvolvimento independente por times de diferentes especialidades técnicas • Requer teste de plataforma específico • Suporte reduzido e menor oferta de desenvolvedores dependendo da plataforma

Considerando o escopo das funcionalidades do aplicativo proposto por este projeto, foi constatado que a maior parte das funcionalidades são relativas a apresentação de conteúdo e que não demandam recursos de alto desempenho para seu bom funcionamento [40]. Soma-se a esta constatação, que promover o acesso a maior parcela possível da comunidade alvo é de suma importância para a proposta deste projeto, portanto todas as plataformas majoritárias devem poder ser atendidas. Por último, é desejavel que alguns recursos nativos estejam disponíveis, como sincronização de informações de acordo com o tipo de conexão disponível, notificações do dispositivo para alertar sobre novos conteúdos e acesso ao sistema de arquivos para armazenamento automático de materiais disponibilizados no aplicativo. Desta forma, a solução Híbrida é a opção mais adequada para a implementação da proposta.

Dentre as tecnologias disponíveis para desenvolvimento móvel híbrido, a ferramenta escolhida para este trabalho, pela sua facilidade de uso, menor custo de desenvolvimento [31] e que contempla os requisitos desejados é o framework de desenvolvimento móvel, Ionic.

5.2.2 Framework de desenvolvimento móvel Ionic

Para a implementação do aplicativo foi utilizada as largamente difundidas tecnologias WEB tradicionais, o HTML5, o CSS e a linguagem de programação Javascript.

Para que o aplicativo possa ser disponibilizado através de pacotes compilados para dispositivos móveis de diferentes plataformas, como iOS, Android ou Windows, foi utilizado o framework de desenvolvimento de aplicativos móveis híbridos Ionic.

O Ionic incorpora e integra outras tecnologias bastante populares no desenvolvimento web e

móvel. Destacaremos dentre elas as seguintes:

- O Apache Cordoba, responsável por intermediar o acesso da aplicação escrita em Javascript aos recursos nativos do aparelho, como sistema de arquivos, wi-fi, rede móvel, geolocalização, entre outros.
- O AngularJS, um framework Javascript para execução de Single Page Applications, construídas sobre o padrao Model-View-ViewModel

A seguir apresentaremos os conceitos chaves destas partes e como elas interagem entre si.

5.2.3 *Apache Cordova*

O *Apache Cordoba* é o *framework* de desenvolvimento de aplicativos móveis híbridos mais popular da atualidade [45], foi criado em 2008 pela Nitobi sob o nome de Phonegap, posteriormente adquirido pela Adobe. A ferramenta cria uma camada recipiente nativa, como uma "casca", para aplicativos escritos em tecnologias Web tradicionais com a finalidade de construir um aplicativo multi-plataforma. Com o Apache Cordova é possível gerar um único *baseline*, ou seja escrever um único código-fonte e compilar para todas as principais plataformas, como Android, iOS, Windows Phone , Blackberry, dentre outras. O código nativo gerado pelo Apache Cordova, permite chamadas aos recursos nativos dos dispositivos nas diversas plataformas de forma otimizada [Apache, Cordova], sem a necessidade de conhecer as interfaces nativas específicas de cada plataforma ou codificar em sua linguagem nativa, como Objective-C ou Java, por exemplo, podendo assim reduzir os custos do projeto [48].

A arquitetura por trás do Apache Cordova é apresentada na figura 5.3.

Na tabela 5.3 podemos ver todos os recursos suportados atualmente pelo Apache Cordoba nas diversas plataformas

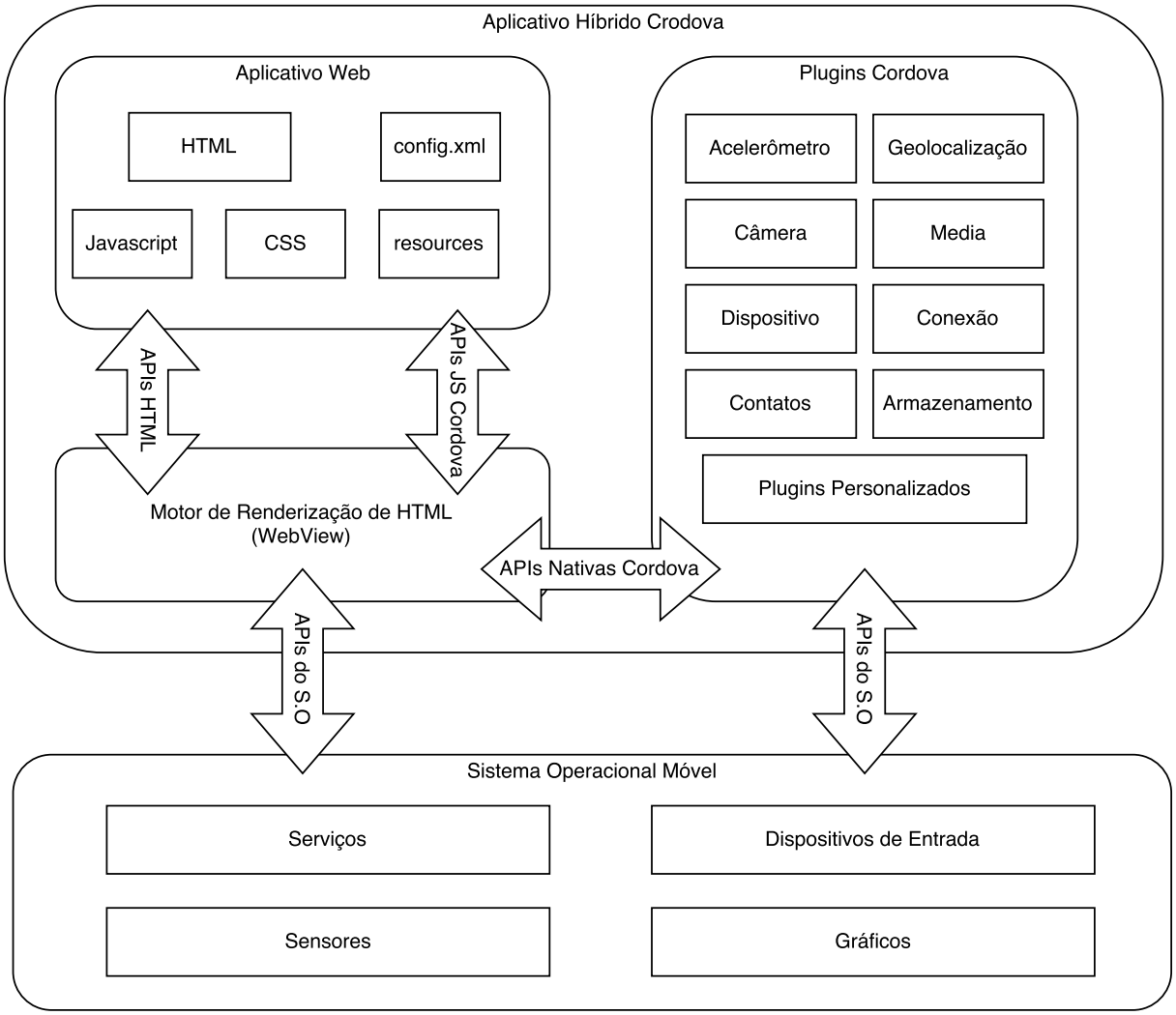

Figura 5.3: Arquitetura de uma aplicação híbrida com o Apache Cordova. [49]

Tabela 5.3: Recursos suportados pela camada nativa do Apache Cordova [47]

	Android	Blackberry10	iOS	Ubuntu	WP8	Windows(8.1, 10, Phone 8.1)
Acelerômetro	✓	✓	✓	✓	✓	✓
Situação da Bateria	✓	✓	✓	✗	✓	✓**
Câmera	✓	✓	✓	✓	✓	✓
Captura	✓	✓	✓	✓	✓	✓
Bússola	✓	✓	✓**	✓	✓	✓
Conexão	✓	✓	✓	✓	✓	✓
Contatos	✓	✓	✓	✓	✓	✓*
Dispositivo	✓	✓	✓	✓	✓	✓
Eventos	✓	✓	✓	✓	✓	✓
Arquivos	✓	✓	✓	✓	✓	✓
Transferência de Arquivos	✓	✓*	✓	✗	✓*	✓*
Geolocalização	✓	✓	✓	✓	✓	✓
Globalização	✓	✓	✓	✓	✓	✓
Navegador Interno	✓	✓	✓	✓	✓	✓*
Media	✓	✓	✓	✓	✓	✓
	✓	✓	✓	✓	✓	✓
Splashscreen	✓	✓	✓	✓	✓	✓
Barra de *Status*	✓	✗	✓	✗	✓	✓**
Armazenamento	✓	✓	✓	✓	✓*	✓*
Vibração	✓	✓	✓	✗	✓	✓**

✓ suportado

✓* suportado com limitações

✓** suportado em algumas versões do S.O.

✗ não suportado

5.2.4 AngularJS

AngularJS [50] é um framework Javascript de código aberto patrocinado e mantido pelo Google. "O objetivo do AngularJS é trazer as ferramentas e recursos que estão disponíveis apenas para o desenvolvimento do lado do servidor para o cliente web e, ao fazê-lo, torná-lo mais fácil de desenvolver, testar e manter aplicativos web ricos e complexos" [51].

AngularJS estende o HTML por elementos personalizados, atributos, classes ou comentários, através das chamadas diretivas. Diretivas são marcações em um elemento DOM que dizem ao compilador HTML do AngularJS para anexar um comportamento especial a esse elemento DOM ou até mesmo transformar o elemento DOM e seus filhos. [52]

A seguir estão os recursos oferecidos pelo AngularJS: [51]

- Ligação bidirecional de dados entre as visões e modelos, o que elimina a manipulação do DOM;

- Motor de templates incorporado;
- Versão reduzida do jQuery chamado jqLite;
- Suporte padrão MVC que ajuda a dividir a aplicação em três áreas distintas: os dados (modelo), a lógica que opera sobre esses dados (controlador) e a lógica que exibe os dados (visão);
- Gerenciamento de dependência;
- Roteamento de links aninhados que permite que o estado do aplicativo seja codificado na URL, podendo então ser restaurado para o mesmo estado posteriormente;
- Serviços integrados para comunicação com servidor RESTful.

AngularJS é uma estrutura que pode ser usada para construir aplicações complexas e de alto desempenho no lado do cliente, de uma forma limpa organizada no padrão MVC. A sua maior vantagem é a construção de aplicações que são de fácil manutenção, testáveis, e facilmente extensíveis.

A figura 5.4 mostra a arquitetura implementada em uma aplicação AngularJS

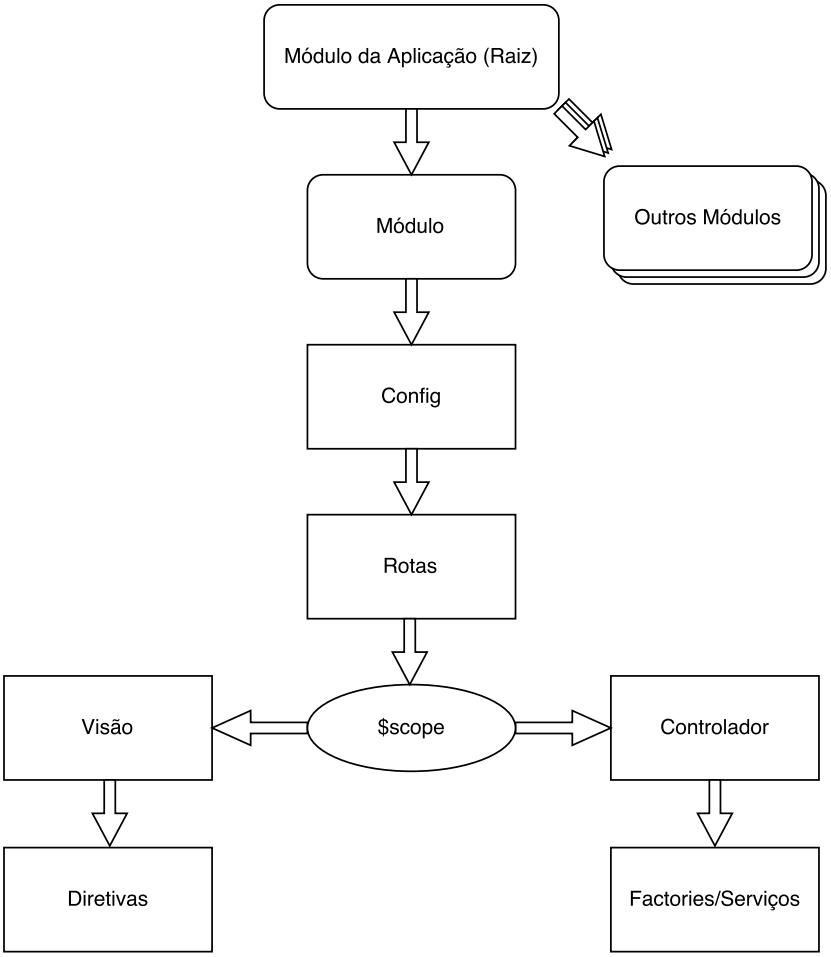

Figura 5.4: Arquitetura de uma aplicação AngularJS [52]

Capítulo 6

Conclusão

O trabalho final tem como resultado uma aplicação híbrida móvel e uma API integrada com o Conexão UFF. Tais aplicações permitem que os alunos da universidade se conectem ao sistema do Conexão UFF e recebam atualizações de disciplinas, tópicos, eventos e arquivos através da aplicação de uma forma simples e prática.

Considerando os sistemas atuais, é possível identificar melhorias e novas funcionalidades que podem ser implementadas em trabalhos futuros. A principal delas sendo a integração com outras plataformas utilizadas por alunos e professores dentro da UFF. Já que o maior objetivo deste trabalho é facilitar a integração de diversas plataformas diferentes.

Outro exemplo de uma nova funcionalidade seria a implementação de um sistema de mensagens entre alunos através do IntegraUFF. Dessa forma os usuários não precisariam se conectar diretamente através das interfaces de cada plataforma integrada para se comunicar com os outros usuários, e as mensagens ficariam centralizadas em um único local.

Atualmente o IntegraUFF só permite a sincronização e leitura das informações obtidas das plataformas integradas. Seria desejável que também fosse possível a escrita de informações em tais plataformas através da aplicação. Para a implementação desta funcionalidade é importante considerar que é necessário verificar se as interfaces das plataformas permitem tal tipo de integração.

Em relação as melhorias de funcionalidades já existentes, o IntegraUFF hoje já possui uma listagem de eventos na qual é possível marcá-los na agenda padrão do celular. Seria interessante extender esta integração para que o calendário fosse exibido de dentro do aplicativo e/ou os eventos pudessem ser marcados na agenda automaticamente. E para os arquivos, uma integração com o serviço do *Dropbox* [12], permitindo que os alunos enviem os arquivos para a nuvem, e se possível que isso fosse feito de forma automática.

O IntegraUFF é um projeto de código aberto, tanto a sua parte mobile quanto o servidor. Os códigos podem ser obtidos através dos links https://github.com/tiagocandido/client-side-integra-uff, para a aplicação móvel, e https://github.com/tiagocandido/server-side-integra-uff, para o servidor.

Referências Bibliográficas

[1] BECK, Kent , *Test Driven Development: By Example*, Addison-Wesley, 2003

[2] BECK, Kent et al. *Agile Manifesto*, Disponível em: <http://agilemanifesto.org/>. Acesso em : 17 mar. 2016

[3] TELES, Vinícius , *Extreme Programming*, Novatec Editora Ltda., 2004

[4] PRESSMAN, Roger , *Software Engineering: A Practitioner's Approach*, 7. ed., McGraw-Hill Education, 2010

[5] TORVALD, Linus , *Git*, Disponível em: <https://git-scm.com/>. Acesso em : 17 mar. 2016

[6] *Code Climate*, Disponível em: <https://codeclimate.com/>. Acesso em : 18 jul. 2016

[7] MATSUMOTO, Yukihiro , *Ruby*, Disponível em: <https://www.ruby-lang.org/>. Acesso em : 17 mar. 2016

[8] HANSSON, David , *Rails*, Disponível em: <http://rubyonrails.org/>. Acesso em : 17 mar. 2016

[9] *Github*, Disponível em: <https://github.com/>. Acesso em : 20 jun. 2016

[10] GAMMA, Erich et al., *Design Patters: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994

[11] FIELDING, Roy, *Architectural Styles and the Design of Network-based Software Architectures*. 2000. 162 f. University of California, California, Irvine, 2000.

[12] *Dropbox*, Disponível em: <https://www.dropbox.com/>. Acesso em : 18 jul. 2016

[13] OLIVEIRA, J. N. A. *Repensando a educação brasileira*. São Paulo: Atlas, p. 137, 2015.

[14] ZANETTE, Elisa Netto. O processo de construção colaborativa da disciplina de Cálculo I, na modalidade de Educação a Distância na graduação. (monografia). Curitiba: UFPR/UNIVALI. 2003.

[15] PETERS, Otto. *Didática do Ensino a Distância*. São Leopoldo,RS : UNISINOS, 2001

[16] BROWNE, T., JENKINS, M., & WALKER, R. (2006). *A longitudinal perspective regarding the use of VLEs by higher education institutions in the United Kingdom. Interactive Learning Environments*, 14(2), 177–192.

[17] *The Worldwide Market for Self-paced eLearning Products and Services: 2011-2016 Forecast and Analysis*, Disponível em: <http://www.ambientinsight.com/Resources/Documents/AmbientInsight-2011-2016-Worldwide-Self-paced-eLearning-Market-Premium-Overview.pdf>, Acesso em: 19 jun. 2016

[18] HAWKINS, B. L., & RUDY, J. A. (2007). *Educause core data service. Fiscal year 2006 summary report.* Boulder, CO: Educause.

[19] PRENDES, M. P.: *Plataformas de campus virtuales de Software Libre: Análisis compartivo de la situación actual de las Universidades Españoles.*, Informe del proyecto EA- 2008-0257 de la Secretaría de Estado de Universidades e Investigación, 2009

[20] MORGAN, G. (2003). *Faculty use of course management systems.* Disponível em: <http://www.educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf>. Acesso em : 17 jul. 2016

[21] *CAMPUS COMPUTING REPORT BR - computação e tecnologia de informação nas instituições de ensino superior no Brasil*, São Paulo, SP, Person Educacional, 2008

[22] *E-Learning Market Trends Forecast 2014 - 2016 Report*, Disponível em: <https://www.docebo.com/landing/contactform/elearning-market-trends-and-forecast-2014-2016-docebo-report.pdf>, Acesso: 20 jul. 2016

[23] *Moodle Statistics*, Disponível em: <https://moodle.net/stats/?lang=pt_br>, Acesso em 14 jul. 2016

[24] SCLATER, N. (2008). *Web 2.0, personal learning environments, and the future of learning management systems.* EDUCAUSE Research Bulletin, 13. Boulder, CO: EDUCAUSE Center for Applied research.

[25] COATS, H.J.R., & BALDWIN, G. (2005) *A critical examination of the effects of learning management systems on university teaching and learning.* Tertiary Education and Management, 11(1), 19-36.

[26] ITMAZI, J., GEA, M., PADEREWSKI, P. e GUITIÉRREZ, F.L.: *A comparison and evaluation of open source learning management systems.* In: Proceedings of the IADIS International Conference – Applied Computing 2005, Algarve, 80–86.

[27] SZALVAY, Victor. *An Introduction to Agile Software Development.* Danube Technologies Inc. 2004.

[28] DYBA, Tore. *Empirical studies of agile software development: A systematic review.* 24 Janeiro 2008.

[29] SCHWABER, K., *Agile Project Management with Scrum*, Microsoft Press, 2004.

[30] BECK, K. *Extreme Programming Explained*, AddisonWesley, 2000.

[31] *Native, web or hybrid mobile-app development.* White paper, IBM Corporation, April 2012. Document Number: WSW14182USEN

[32] WEXLER, S., GREY, N., MILLER, D., NGUYEN, F. and BARNEVELDA, A. (2008): *Learning management systems:The good, the bad, the ugly ... and the truth]. e-learning Guild Ed.*

[33] L. CORRAL, *A. Sillitti, and G. Succi. Mobile multiplatform development: An experiment for performance analysis.* Procedia Computer Science, 10:736–743, 2012.

[34] B. Fling. *Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps.* O'Reilly Media, Inc., 2009.

[35] DESNICA, E., LETIC, D., NAVALUSIC, S. (2010). *Concept of distance learning model in graphic communication teaching at university level education.* Journal TTEM – Technics, technologies, education, management, 5(2), 378-388.

[36] KUDUMOVIC, M., KUDUMOVIC, D., MESANOVIC, N. HUREMOVIC, E. (2010). *Modern information communication technologies and educational technologies applied to education of medicine.* HealthMED Journal, 4(4), 158–162.

[37] ISLJAMOVIC, S., PETROVIC, N. JEREMIC, V. (2011). *Technology enhanced learning as a key component of increased environmental awareness amongst students from the University of Belgrade.* Journal TTEM – Technics, technologies, education, management, 6(4), 1175-1181.

[38] ALIER, M., CASANY, M.J. and PIGUILLEM, J. (2009): *Towards mobile learning applications integration with learning management systems. Multiplatform e-learning systems and technologies: mobile devices for ubiquitous ICT-based education*, Information Science Reference, Hershey, New York, 182–194.

[39] MALAVOLTA, I., RUBERTO, S., TERRAGNI, V., SORU, T., *Hybrid Mobile Apps in the Google Play Store: an Exploratory Investigation.* International Conference on Mobile Software Engineering and Systems (MOBILESoft), ACM, 2015.

[40] MALAVOLTA, I., RUBERTO, S., TERRAGNI, V., SORU, T., *End Users' Perception of Hybrid Mobile Apps in the Google Play Store.* Mobile Services (MS), 2015 IEEE International Conference on. IEEE, 2015.

[41] J. M. WARGO. PhoneGap Essentials: *Building Cross-Platform Mobile Apps.* Addison-Wesley, 2012.

[42] FRAMINGHAM, *(IDC) Worldwide Quarterly Mobile Phone Tracker*, Disponível em: <https://www.idc.com/getfile.dyn?containerId=IDC_P19661attachmentId=47178213>, Acesso em: 14 Julho 2016

[43] RAHUL R.C.P TOLETY, S. B.. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach, India Conference(INDICON),2012 , pp 625-629,7-9 Dezembro, 2012

[44] XANTHOPOULOS S., XINOGALOS, S., *A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications*, BCI'13, Thessaloniki, Greece, pp 213-220, 19-21 Setembro, 2013.

[45] PALMIERI, M., SINGH, I. and CICCHETTI, A. 2012. *Comparison of cross-platform mobile development tools*. XVI International Conference on Intelligence in Next Generation Networks (ICIN„ 179-186. DOI: 10.1109/ICIN.2012.6376023

[46] *SWOT analysis: Hybrid versus native development in IBM Worklight*, Abril 28, 2014 Disponível em: <https://www.ibm.com/developerworks/community/blogs/mobileblog/entry/swot_analysis_hybrid_versus_native_development_in_ibm _worklightlang=en>, Acesso em: 16 de abr. 2016

[47] *Apache Cordova Platform Support*, <https://cordova.apache.org/docs/en/latest/guide/ support/index.html>, Acesso em: 20 jul. 2016.

[48] TIAN, L.; DU H.; TANG L.; XU Y„ *The discussion of cross-platform mobile application based on Phonegap*, Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on, vol., no., pp.652,655, 23-25 May 2013.

[49] *Apache Cordova Architecture*, Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html#architecture>, Acesso: 20 Setembro 2015.

[50] *Guidebook. Mobile Conference & Event Apps*. Disponível em: <https://guidebook.com/event-apps/>, Acesso em: 30 de março 2016.

[51] FREEMAN, A. *Pro AngularJS*. 1a ed. Expert's Voice in Web Development. Apress, 2014. ISBN: 1430264489

[52] *AngularJS - Superherois JavaScript MVW Framework*. Diponível em: <https://angularjs.org/> Acesso em: 14 de jul. 2016.

[53] PHAN, H., *Full-stack Mobile App With Ionic Framework*. Hoc Phan, 2014.