UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FERNANDO SEHNEM HECK

Sistema Móvel de Controle de Presença

Trabalho de Conclusão apresentado como
requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Prof. Dr. Leandro Krug Wives
Orientador

Porto Alegre, 21 de Outubro de 2013

"As pessoas boas devem amar seus inimigos."
— RAMON VALDEZ MADRUGA

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CPD	Centro de Processamento de Dados
DAO	Data Access Object
EAD	Educação a Distância
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
MVC	Model View Controller
SAV	Sala de Aula Virtual
SOAP	Simple Object Access Protocol
UFRGS	Universidade Federal do Rio Grande do Sul

LISTA DE FIGURAS

Mobile Application to control students attendance in class

ABSTRACT

The present work aims to describe the development of an application named Mobile System to control students attendance in class. Such system was created to help teachers account students attendances in classes. The idea is to let such task less tedious and more transparent. The application is strongly integrated to a platform known as Sala de Aula Virtual (Virtual Class Room), which was developed by the Data Center of the Federal University of Rio Grande do Sul. Such integration allows data interoperation between the application and the platform without human interaction.

Keywords: iOS, Mobility, iPhone, Call Roll.

RESUMO

O presente trabalho tem por objetivo o desenvolvimento de um sistema, Sistema Móvel de Controle de Presença, para o auxilio dos professores no dia a dia. Tal sistema visa tornar menos tediosa e mais transparente a tarefa de contabilizar a presença dos alunos. O Sistema Móvel de Controle de presenças tem uma forte integração com o SAV, plataforma criada em 2011 pelo CPD da UFRGS. Tal integração permite que após realizada a chamada por parte do professor, o resultado esteja disponível no SAV sem necessidade de mais interação humana.

Palavras-chave: IOS, Mobilidade, iPhone, Chamada, Sala de Aula Virtual.

1 INTRODUÇÃO

Os dispositivos móveis estão cada dia mais presentes na vida das pessoas. Segundo dados colhidos pelo IDC, a venda de tais equipamentos já superou a de computadores em 80%. Nota-se, portanto, que esse fenômeno não é algo passageiro, tendo em vista que as previsões são de que o número de usuários desses aparelhos aumentará consideravelmente nos próximos anos (IDC, 2013a).

Tais dispositivos são, na verdade, mini computadores com diversos sensores integrados. Com eles, podemos criar poderosas ferramentas para o auxílio de profissionais, como por exemplo, o professor, que, por sua vez, será o principal beneficiário na realização deste trabalho.

Segundo a Lei 9.394/96, a qual estabelece as diretrizes e bases da educação nacional, a universidade é obrigada a exigir 75% de frequência do aluno para que este possa ser aprovado. A responsabilidade de fiscalizar os alunos de uma certa disciplina fica a cargo do professor responsável pela mesma. Infelizmente, os professores não possuem muitas formas automatizadas de realizar esse procedimento, de modo que cada um cria uma abordagem própria para agilizar esse processo. No entanto, essa informação fica, normalmente, centralizada em algumas folhas de papel ou no próprio computador do professor, não ficando disponível para o acesso do aluno. Diante disso, a fim de tornar mais fácil e transparente o processo, além de menos tedioso, sugere-se o desenvolvimento de um sistema móvel capaz de contabilizar a frequência dos alunos nas escolas e universidades de um modo geral. É nesse contexto de modernização e busca por aperfeiçoamento que idealizou-se o presente trabalho.

1.1 Objetivo

O presente trabalho tem por objetivo trazer a tecnologia móvel para sala de aula na forma de um aplicativo para *smartphone* e *tablet* da Apple, de modo a auxiliar o professor e o aluno. O aplicativo propõe-se a facilitar a tarefa de controlar a frequência dos alunos das disciplinas lecionadas pelo professor, bem como permitir que o próprio aluno possa conferir a sua frequência na *Web*, de forma sigilosa. O desenvolvimento deste trabalho compreende o estudo das tecnologias envolvidas, a definição do problema a ser resolvido, a apresentação da solução de forma detalhada e a implementação do aplicativo que resolve o problema.

1.2 Pesquisa de Aceitação

De modo a verificar o público atingido e o nível de aceitação do sistema proposto, pediu-se aos professores do Instituto de Informática da UFRGS que respondessem a um breve questionário durante o período de 12/08/2014 a 13/08/14. No questionário, o termo "dispositivo móvel"foi apresentado como *smartphone*, *tablet* ou equipamento semelhante que possui acesso a internet. O questionário foi criado no Google Forms e contou com a participação de 28 pessoas. A seguir, são enumerados alguns resultados interessantes:

1. Praticamente todos os professores possuem um dispositivo móvel.

Figura 1.1: Quantidade de dispositivos móveis.

2. Para um novo sistema de chamadas o preferido é algo que possa ser acessado tanto via *Web* quanto por aplicativo móvel.

Figura 1.2: Preferência de plataforma

3. Os professores do Instituto estão basicamente divididos entre dispositivos iOS e Android, cada um desses representando um montante de 41 % dos pesquisados

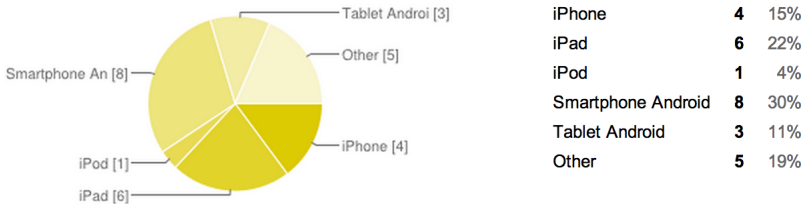

Figura 1.3: Distribuição de sistemas operacionais móveis.

Por fim, importa ressaltar que o CPD da UFRGS interessou-se bastante pelo projeto e teve um importante papel em sua realização. Já havia algum tempo que o CPD estava a procura de uma iniciativa que levasse a Universidade a usar algum tipo de sistema móvel.

1.3 Organização

Esse trabalho apresenta a seguinte organização:

No segundo capítulo – Fundamentação Teórica - será apresentado um panorama geral acerca das plataformas móveis existentes bem como serão estudados os três principais meios de desenvolvimento mobile. Em tal capítulo, serão analisados alguns aplicativos que se propõem a resolver problemas semelhantes ao proposto neste trabalho.

O terceiro capítulo – Projeto do iPresence - busca mostrar as diferentes partes do sistema, mostrando aspectos da arquitetura e também da modelagem do problema. Por fim o sistema será apresentado através de *User Stories*.

O quarto capítulo – Desenvolvimento da Aplicação – detalha tecnologias e ferramentas utilizadas no desenvolvimento deste trabalho, tal como algumas das várias decisões tomadas.

O quinto capítulo - Funcionamento do Sistema - mostra o novo fluxo criado no SAV bem como todas as telas do aplicativo junto de comentários sobre as funcionalidades.

Por fim, o sexto e último capítulo - Conclusão - trata sobre uma restrospectiva do que foi aprendido durante obtidos e planos para futuras funcionalidades.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo serão apresentados e explicados os conceitos relacionados que servem como base para o desenvolvimento do trabalho. Na seção 2.1 será explicado o que são plataformas móveis e algumas plataformas importantes serão comentadas. Na seção 2.2 será feita uma análise acerca do desenvolvimento de aplicativos na plataforma escolhida. Por fim, 2.3 serão apresentados trabalhos semelhantes que visam resolver problemas parecidos com o aqui proposto.

2.1 Plataformas Móveis

Segundo Fling, uma plataforma móvel tem o dever de prover acesso ao dispositivo. O autor também afirma que é a plataforma que define as APIs, interfaces e linguagens de programação que estão disponíveis ao programador para o desenvolvimento de aplicativos e serviços (FLING, 2009).

Assim, serão analisadas algumas plataformas móveis, tendo em vista a sua relevância histórica e/ou expressividade no mercado, quais sejam: Android, iOS, Windows Phone, Blackberry, Symbian e Firefox OS.

2.1.1 Android

Android é uma plataforma que começou a ser desenvolvida pela Android Inc em 2003. Em 2005 foi comprado pela Google e foi desenvolvido em parceria com a Open Handset Alliance. A Open Handset Alliance é um consórcio de empresas de hardware, software e telecomunicações, cujo objetivo é o desenvolvimento de tecnologias móveis abertas (STEELE; TO, 2010).

O Android é um sistema *opensource* baseado em Linux desenvolvido para dispositivos móveis. Ademais, por ser uma plataforma aberta, ele permite que fabricantes de dispositivos o customizem para as necessidades de seus aparelhos, além de permitir o acesso ao desenvolvedor a todas as funcionalidades do celular ou *tablet* (STEELE; TO, 2010).

Ainda, a principal linguagem de programação utilizada pelos desenvolvedores de aplicativos Android é o Java, também sendo possível o desenvolvimento de aplicativos em C/C++. Esta plataforma é vendida em aparelhos de diversos fabricantes, tais como: Samsung, HTC, Sony, LG e Motorola (STEELE; TO, 2010).

2.1.2 iOS

O iOS foi lançado em 9 de janeiro de 2007, pela Apple, juntamente com o primeiro iPhone que redefiniu a computação móvel mundial (VERGE, 2013). A esse respeito, Fling afirma: *"On the morning of January 9, 2007, Steve Jobs went onstage at the MacWorld conference in San Francisco to usher in the fifth and final era and to change the mobile world. He introduced the world to the iPhone."* (FLING, 2009). Nota-se, portanto, que esta plataforma marcou a transição para uma nova era dos dispositivos móveis, passando-se da era *Smart* para a *Touch*.

Esta é uma plataforma proprietária e foi baseada no Mac OS X. Hoje o iOS é tão bem aceito e popular que os usuários o preferem ao Mac OS X, o qual, inclusive, está sendo desenvolvido cada vez mais semelhante ao próprio iOS.

Salienta-se, ainda, que tal plataforma está presente apenas em dispositivos fabricados e vendidos pela própria Apple. A linguagem de programação utilizada pelos desenvolvedores de aplicativos iOS é, principalmente, o Objective C, sendo possível também a utilização de C/C++.

Essa plataforma foi usada para o desenvolvimento do aplicativo aqui proposto. Portanto, na subseção 2.1.7, serão apresentadas mais informações a seu respeito.

2.1.3 Windows Phone

O Windows Phone foi lançado em outubro de 2010 com o nome de Windows Phone 7. Segundo Petzold, "com o sua aparência limpa, fontes marcantes, e os novos paradigmas organizacionais, a plataforma Windows Phone 7 da Microsoft representa não só uma ruptura com o passado móvel do Windows, mas também se diferencia de outros smartphones atualmente no mercado" (PETZOLD, 2010).

A sua interface totalmente diferente das concorrentes foi batizada de Metro e foi muito bem recebida pela comunidade. Ademais, a referida plataforma é muito interessante do ponto de vista do desenvolvimento, tendo em vista que suporta as plataformas de programação XNA e Silverlight. Tal plataforma é distribuída em aparelhos de diversas companhias, tais como: HTC, Huawei, Nokia e Samsung.

2.1.4 Blackberry

Blackberry é a plataforma criada pela Research In Motion Limited (RIM) e revolucionou a indústria móvel em 1999. Tal plataforma perdeu muito mercado desde então e hoje corresponde somente a 2,7% do mercado. No entanto, ainda é vastamente conhecida pela sua segurança e facilidade na troca de *e-mails*, o que a torna bastante visível no mercado corporativo.

Até 14 de maio de 2013, a principal linguagem de programação utilizada no desenvolvimento de aplicativos Blackberry era o Java. Todavia, tudo mudou a partir da referida data, tendo em vista o lançamento do Blackberry 10 - maior reformulação da plataforma desde a sua invenção - permitindo, dessa maneira, a execução de aplicativos Android, além de aplicativos feitos com HTML 5 e JavaScript (BLACKBERRY, 2013).

2.1.5 Symbian OS

Symbian é uma plataforma que está há muito tempo no mercado mobile, tendo sido construída em cooperação pela Nokia, Sony Ericsson, NTT, DoCoMo e Symbian Ltd. em 1998, sendo posteriormente adquirida pela Nokia (ZUCKER; RISCHPATER, 2010). Embora tenha sido uma das mais populares no mundo da mobilidade, hoje a sua fatia de mercado não ultrapassa 0,2%, segundo dados colhidos pela IDC (IDC, 2013b).

Os desenvolvedores de aplicativos para Symbian contam com uma grande gama de plataformas disponíveis, incluindo: Qt (um ambiente multi plataforma baseado em C++), uma plataforma baseada em *Web* que utiliza HTML 5, JavaScript e CSS, APIs Java Me e também Adobe Flash.

2.1.6 Firefox OS

Segundo dados colhido no próprio *site* da Mozilla, "Firefox OS é um sistema operacional novo criado pela Mozilla que permite aos usuários instalarem e executarem aplicações *Web* utilizando HTML, CSS e JavaScript."

Essa plataforma, no momento do desenvolvimento deste trabalho, ainda está em fase beta. No entanto, embora ainda não tenha sido lançada, promete maior acesso ao hardware dos aparelhos para as aplicações *Web*, o que não é comum, bem como uma maior simplicidade de desenvolvimento e um código aberto. Seu mercado é voltado para os *low ends*.

2.1.7 Plataforma Escolhida - iOS

A plataforma escolhida para o desenvolvimento do aplicativo aqui proposto foi o iOS. A escolha dessa plataforma foi feita após a pesquisa de aceitação, citada na seção 1.2. Tal pesquisa demonstrou que muitos dos professores do instituto de informática já tem contato com tecnologias móveis e que quase metade deles possuem um dispositivo iOS. Outro motivo que foi determinante para a escolha é a baixa fragmentação das versões iOS presentes no dispositivo e também pela fragmentação de tamanhos. A baixa fragmentação torna o desenvolvimento mais simples, reduzindo assim o *time to market*. Por fim, como anteriormente mencionado, o iOS trouxe consigo uma nova era para os dispositivos móveis. Desta forma, nada melhor do que usar essa plataforma tão inovadora para desenvolver o primeiro aplicativo móvel da Universidade.

2.2 Desenvolvimento Mobile

Ao iniciar o desenvolvimento de aplicativos para dispositivos móveis, faz-se necessário escolher um dentre três caminhos diferentes, quais sejam (BUDIU, 2013):

1. **Desenvolvimento de aplicativo Nativo**, o qual será instalado no dispositivo;

2. **Desenvolvimento de aplicativo *Web***, o qual rodará no navegador do aparelho;

3. **Desenvolvimento de aplicativo Híbrido**, o qual trata-se de uma mistura dos anteriores, ou seja, será instalado no dispositivo mas será desenvolvido majoritariamente para o navegador na forma de um *site*;

Nas subseções a seguir essas três possibilidades serão analisadas e comparadas

2.2.1 Aplicativo Nativo

O aplicativo nativo, ou de plataforma, é o tipo mais antigo e comum de programa para celular. Criar um aplicativo desses implica em decidir a plataforma alvo, determinando como consequência os celulares ou *tablets* que poderão usar o aplicativo. A grande maioria dos aplicativos de plataforma é certificado, vendido e distribuído através de um portal ou uma loja *online*. Esse tipo de aplicação, por ser instalado no dispositivo, pode ser usado tanto *online*, quanto *offline* (FLING, 2009).

Ainda, por ser desenvolvido especificamente para uma plataforma, tal aplicativo apresenta um desempenho muito maior do que os aplicativos *Web* e tem acesso a maioria das funcionalidades do dispositivo, tais como: localização por GPS, câmera, sensores e sistema de arquivo. Por esse motivo, as empresas que disponibilizam plataformas móveis, devem certificar e validar os aplicativos enviados às lojas, para que o usuário final não receba um aplicativo malicioso (FLING, 2009).

Por fim, ressalta-se que devido a necessidade de conhecimentos específicos acerca de determinada plataforma, o seu desenvolvimento torna-se mais custoso, não permitindo fácil portabilidade para outras plataformas (FLING, 2009).

2.2.2 Aplicativo Web

Os aplicativos *Web* são, na verdade, *sites* especialmente preparados para dispositivos móveis. Eles rodam no navegador do aparelho e geralmente são desenvolvidos em HTML 5, ou alguma outra plataforma *Web*. Esses aplicativos podem ser salvos como favoritos, tornando-os, assim, disponíveis na tela inicial do celular ou *tablet*, junto com outros aplicativos (BUDIU, 2013).

Tais aplicativos se popularizaram com a chegada do HTML 5, que tornou possível a criação de aplicativos *Web* com funcionalidades semelhantes as nativas, tais como: uso de GPS, fazer ligações, gestos como *swipe* e até uso *offline* com o auxílio de *cache*, mesmo que de modo simplificado. Entretanto, algumas funcionalidades ainda não são acessíveis, como por exemplo, controle do sistema de arquivos, uso da câmera, execução em segundo plano e acesso a sensores (BUDIU, 2013).

Devido a tecnologia *Web* envolvida no desenvolvimento desses aplicativos, a portabilidade entre diversas plataformas é quase sempre garantida com praticamente nenhuma alteração. Ainda, por esse mesmo motivo, tais aplicativos possuem um custo mais baixo de desenvolvimento e manutenção (BUDIU, 2013).

2.2.3 Aplicativo Híbrido

Um aplicativo híbrido, como dito anteriormente, é uma mistura do nativo e *Web*. Esse aplicativo é instalado através da loja de aplicativos e o seu componente mais importante é um navegador. Ressalta-se que a maioria das plataformas móveis atuais possuem suporte para esse tipo de aplicação (BUDIU, 2013).

Os aplicativos híbridos são construídos em uma plataforma intermediária. Essa, por sua vez, é uma "ponte"entre certos *callbacks* JavaScript e o código nativo da plataforma, permitindo que tais programas acessem, praticamente, os mesmos componentes dos celu-

lares e *tablets* que os aplicativos nativos.

Atualmente, existem alguns *frameworks* que auxiliam no desenvolvimento híbrido e multiplataforma, como por exemplo, o PhoneGap, Icenium e Titanum. É necessário ressaltar que o aplicativo final necessita ser compilado para cada plataforma especificamente. Todavia, o código nativo representará, apenas, uma pequena fração do montante total da aplicação (BUDIU, 2013).

Nota-se que devido as diferentes interfaces disponibilizadas pelas plataformas móveis, nem sempre é possível acessar sensores e outros componentes do mesmo modo, o que dificulta o desenvolvimento desse tipo de aplicativo. Podemos citar como exemplo disso, os passos necessários para se obter uma localização de GPS em iOS e Android. Em Android, é possível requisitar uma localização para o sistema, que caso a possua, retornará essa informação imediatamente. No iOS, por sua vez, tal funcionalidade não existe, logo, deve-se ativar o provedor de posições de GPS e esperar por uma localização.

2.2.4 Comparação dos Diferentes Tipos de Desenvolvimento

Nesta subseção será feita uma comparação entre os três diferentes tipos de desenvolvimento para dispositivos móveis, ja aprofundados anteriormente. A comparação valer-se-á de critérios utilizados e analisados por (BUDIU, 2013).

2.2.4.1 Recursos do dispositivo

No presente contexto, entende-se por Recursos do Dispositivo a permissão para utilização de sensores e realização de tarefas especiais que a plataforma oferece. Como exemplo de sensores, pode-se citar a câmera, o GPS, o giroscópio, o acelerômetro, entre outros. Por sua vez, como exemplo de tarefas especiais, pode-se citar a realização de ligações, o envio de *e-mails* e mensagens, bem como o acesso a internet.

- **Nativo:** Tem acesso total.
- **Web:** Pouco acesso a recursos. Geralmente está restrito ao uso de GPS, a realização de chamadas e a possibilidade de enviar e-mail através do aplicativo padrão.
- **Híbrido:** Por tratar-se, em parte, de um aplicativo nativo, possui acesso total aos recursos. Ressalta-se, porém, que para valer-se desse privilégio, é necessário acessar os recursos pela parte nativa. Caso contrário, terá as mesmas restrições do aplicativo *Web*.

2.2.4.2 Modo Offline

Modo Offline refere-se a capacidade do aplicativo de continuar funcionando mesmo em caso de falta de conexão com a Internet.

- **Nativo:** A qualidade do uso sem conexão é apenas imposta pelo tipo de aplicativo. Um aplicativo, cujo objetivo é a reprodução de músicas, por exemplo, não teria a sua usabilidade afetada pela falta de Internet. Todavia um aplicativo de *streaming* de músicas ficaria com o seu funcionamento prejudicado, ou até completamente inutilizável.

- **Web:** A falta de conectividade inviabiliza o uso desses aplicativos, salvo o caso em que ele foi previamente salvo em *cache*. Porém a funcionalidade fica muito reduzida.

- **Híbrido:** Esse tipo de aplicativo é acessível offline. Todavia, é possível que tenha funcionalidades reduzidas.

2.2.4.3 Desempenho

- **Nativo:** Melhor desempenho, por ser desenvolvido especificamente para a plataforma.

- **Web:** Baixo desempenho, tendo em vista que o aplicativo necessita constantemente de informações da Internet.

- **Híbrido:** Baixo desempenho, visto que o navegador disponível para os aplicativos não é tão "poderoso"quanto o do dispositivo. Ademais, grande parte do desenvolvimento é feito em HTML 5 e não de forma nativa, reduzindo o desempenho.

2.2.4.4 Manutenabilidade

Entende-se por manutenabilidade, o tempo e o custo para consertar possíveis *bugs*, bem como o tempo que demora para que a solução chegue ao usuário.

- **Nativo:** A manutenção é cara, demorada e deve ser feita especificamente para cada plataforma. O aplicativo pode apresentar diferentes problemas em cada plataforma. Depois de resolvido o problema, o aplicativo deve ser enviado novamente para a loja, passando por um novo processo de avaliação. O tempo de avaliação, por sua vez, varia de acordo com a loja. Depois de disponível na loja, basta que os usuários façam o *update* do aplicativo, o que pode nunca acontecer.

- **Web:** Tão simples quanto manter um *Website*. A atualização de conteúdo e funcionalidade é rápida, fácil e logo torna-se disponível para todos os usuários. Para que o usuário note a atualização basta que acesse o aplicativo novamente.

- **Híbrido:** É semelhante ao *Web*. Contudo, suas funcionalidades novas ou correções de *bugs* podem exigir uma atualização do aplicativo.

2.2.4.5 Censura

- **Nativo:** Forte restrição de conteúdo imposto pelas fabricantes das plataformas. Geralmente não são aceitos conteúdos pornográficos, muito violentos ou que vão de encontro a leis.

- **Web:** Praticamente inexistente. Está sujeito as mesmas regras que regem as páginas *Web* comuns.

- **Híbrido:** Por ser em parte, aplicativo nativo, está sujeito as mesmas regras que os aplicativos nativos.

2.2.4.6 Custo de Desenvolvimento

- **Nativo:** Geralmente é mais caro que os demais, pois é necessário conhecimento específico da plataforma. Normalmente os aplicativos são lançados para mais de uma plataforma diferente, o que eleva bastante o custo final do projeto. O mais comum é que um aplicativo seja lançado para as três principais plataformas móveis do momento (Android, iOS e Windows Phone), o que implica no desenvolvimento de três projetos distintos pouco ou nenhum código compartilhado, o que acaba por elevar o custo.
- **Web:** Custo reduzido, devido ao uso de tecnologias mais conhecidas, como o HTML, CSS e JavaScript. O aplicativo normalmente é desenvolvido de forma genérica, de modo que funcione nas mais variadas plataformas.
- **Híbrido:** Custo geralmente fica entre o *Nativo* e *Web*, pois grande parte do aplicativo pode ser feita utilizando tecnologias *Web*.

2.2.4.7 Interface de Usuário

Uma das prioridades no desenvolvimento móvel é a experiência de usuário. Esta deve ser compatível com o que é esperado em cada plataforma. Isso pode incluir a aparência do aplicativo, bem como a forma de interagir com ele.

- **Nativo:** É o melhor meio de desenvolvimento quando o quesito é interface de usuário. Possibilita de forma simples que o aplicativo tenha o *look and feel* esperado.
- **Web:** A experiência de usuário pode simulada e uma diferenciação entre plataformas é bem complicada de ser atingida.
- **Híbrido:** Muito semelhante aos aplicativos *Web*.

2.2.4.8 Portabilidade

Portabilidade é a facilidade de se utilizar um aplicativo em diferentes plataformas.

- **Nativo:** Na maioria dos casos o código deve ser totalmente reescrito para diferentes plataformas, com exceção de *backends* ou alguma biblioteca compartilhada em C/C++.
- **Web:** Por não existir diferenciação entre plataformas e por ser acessado pelo *browser*, esse tipo de desenvolvimento atinge uma ampla quantidade de plataformas diferentes.
- **Híbrido:** Apenas a parte nativa do código precisa ser reescrita para as diferentes plataformas.

2.3 Trabalhos Relacionados

Nesta subseção serão apresentados alguns aplicativos que possuem objetivo semelhante ao do presente trabalho. Apesar dos aplicativos citados a seguir possuírem mais

funcionalidades, todos eles demandam um grande esforço de configuração que deve ser repetido a cada semestre. Outro ponto que vale ressaltar é que o presente trabalho é integrado com o SAV da UFRGS. De modo que o Sistema Móvel de Controle de Presença é um aplicativo que se configura sozinho sem esforço e permite aos alunos observarem os resultados através do SAV.

2.3.1 Gradebook Pro

Figura 2.1: Lista de alunos com frequência já atribuída

Gradebook Pro é um aplicativo para iOS. Esse aplicativo é uma ferramenta de gerência de turmas que permite mais do que contabilizar a frequência dos alunos. Com esse aplicativo é possível manter registros de notas, desempenho, comportamento, participação e outros critérios sobre os participantes. É possível também exportar os resultados em formato CSV para uso posterior.

Esse aplicativo não é grátis e seu custo é de U$ 10,49 na loja de aplicativos da Apple. A sua interface é simples e prática de usar porém faz-se necessário um esforço de configuração inicial que é diretamente proporcional a quantidade de turmas e participantes. A Figura 2.1 mostra a interface do aplicativo depois de devidamente configurado e já com alguns resultados. Mais informações podem ser encontradas no *website* [1] do desenvolvedor.

2.3.2 Mobile Attendance

O Mobile Attendance é um aplicativo que visa, de forma simples, permitir que o educador registre a frequência de seus alunos. Esse aplicativo possibilita a criação de inúmeras turmas, possui integração com as planilhas do Google e permite exportar e compartilhar por e-mail o resultado da lista de frequência.

O referido aplicativo é pago e seu custo é de U$ 1,99 na loja de aplicativos da Apple. Possui uma interface simples e fácil de usar. Contudo, requer um esforço considerável

[1] http://gradebookapp.com/

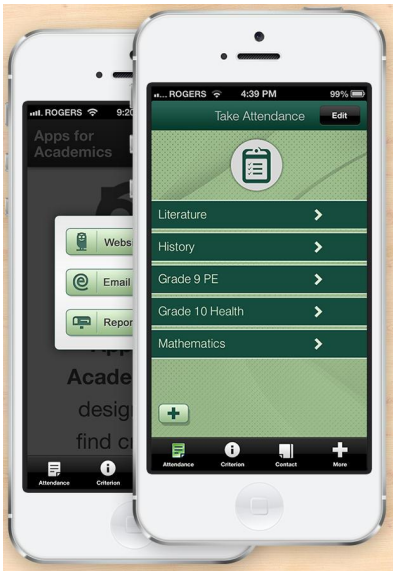

Figura 2.2: Lista de alunos com frequência do Gradebook

de configuração. A Figura 2.2 mostra a primeira tela do aplicativo já com as turmas cadastradas. Mais informações podem ser encontradas no *site* do desenvolvedor [2].

2.3.3 Sala de Aula Virtual

Figura 2.3: Sala de Aula Virtual - Tela Inicial

O SAV, ilustrado na Figura 2.3, é uma plataforma da UFRGS que tem como objetivo facilitar a comunicação e troca de conteúdo entre alunos e professores. O SAV permite ao professor enviar *e-mails* facilmente para alunos de uma certa disciplina, fazer a chamada (conforme a Figura 2.4) e até atribuir notas de trabalhos e provas. Já para o aluno, o SAV possibilita ver quem são os colegas de uma disciplina, permitindo o envio de *e-mails* para colegas e professores, bem como a visualização de suas faltas e notas. O que o diferencia das outras plataformas Educação a Distância (EAD) da Universidade é a sua total incorporação ao Portal de Serviços do Usuário.

A plataforma SAV é uma parte muito importante do sistema aqui proposto. Sua relação com o presente trabalho será melhor explicada no capítulo Projeto do iPresence. Mais

[2] http://www.appannie.com/

informações sobre o SAV podem ser encontradas no manual online [3].

Figura 2.4: Sala de Aula Virtual - Chamada

2.4 Resumo

Nesse capítulo foram apresentadas as diferentes plataformas móveis presentes no mercado. As plataformas Android, iOS, Windows Phone, Blackberry, Symbian OS e Firefox OS foram apresentadas e suas principais características comentadas. A plataforma iOS foi a escolhida para o desenvolvimento do presente trabalho, portanto na subseção 2.1.7 foi apresentada uma justificativa para a escolha.

Os diferentes métodos de desenvolvimento de aplicativos foram expostos e comparados. Também foram apresentados trabalhos semelhantes e o por que deles não satisfazerem as necessidades que o Sistema Móvel de Controle de Presença vem saciar.

[3] www1.ufrgs.br/Tutoriais/salaaulavirtual/manual/Manual_SaladeAulaVirtual.html

3 PROJETO DO IPRESENCE

Neste capítulo serão detalhadas as questões relacionadas ao projeto do sistema proposto. Primeiramente, na seção 3.1, serão expostas as diferentes partes que compõe o sistema bem como a forma com que elas interagem. Na seção 3.2 o sistema será explicado por meio de *User Stories* e das entidades criadas para representar o problema.

3.1 Arquitetura do Sistema Móvel de Controle de Presença

O Sistema Móvel de Controle de Presença consiste na soma de três sistemas. A primeiro e mais importante é o Sala de Aula Virtual (SAV), já citado na subseção 2.3.3. O SAV é um sistema já existente e é não somente a fonte de dos dados necessários para o presente trabalho como também é o destino dos dados gerados pelo aplicativo. A segunda parte mais importante é o *Web-Service* localizado no Centro de Processamento de Dados (CPD). Esse *Web-Service* serve como ponte entre o banco de dados do SAV e o aplicativo móvel e é através dele que o aplicativo obtém as informações necessárias e atualiza o SAV com as chamadas feitas pelos professores. Por fim, o aplicativo móvel desenvolvido para iOS é a maneira como os professores poderão alterar os dados do SAV referentes a frequência dos alunos.

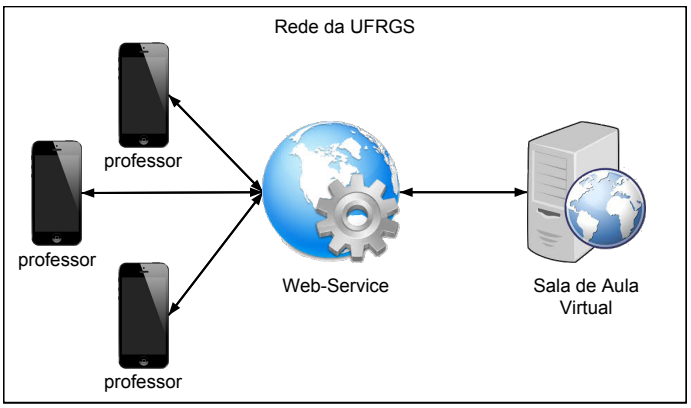

Figura 3.1: Visão Geral do Sistema

O foco do presente trabalho é o aplicativo desenvolvido para a plataforma iOS e o *Web-Service* que se localiza nos servidores do CPD da UFRGS, tendo em vista que o SAV é um sistema independente e que já está em funcionamento desde 2011.

3.1.1 *Web-Service*

O *Web-Service* foi desenvolvido utilizando Yii, um *framework* PHP de alta performance para o desenvolvimento de aplicações *Web* 2.0. O Yii é compatível com importantes padrões, como, MVC, DAO, cache e muito mais, o que reduziu o tempo de desenvolvimento de forma significativa. Nota-se também que este *framework* já é bastante usado pelo CPD de modo que a manutenção futura do serviço será facilitada.

O *Web-Service* está hospedado nos servidores do CPD e pode ser acessado somente através da rede da UFRGS. Dessa forma, não é permitido que pessoas sem vinculo com a Universidade tenham acesso, o que garante maior segurança.

A comunicação entre o *Web-Service* e o aplicativo móvel é feita utilizando os protocolos HyperText Transfer Protocol Secure (HTTPS) e Simple Object Access Protocol (SOAP). Por questões de performance, o "corpo"do envelope SOAP, que possui a informação que transita do *Web-Service* para o aplicativo e vice-versa, foi colocado em formato JavaScript Object Notation (JSON).

3.1.2 Aplicativo Móvel - iPresence

O aplicativo móvel foi desenvolvido de forma nativa para a plataforma iOS. A linguagem de programação utilizada é o Objective-C. O Aplicativo recebe os dados provenientes do SAV através do *Web-Service* em formato JSON. Os dados por sua vez são parceados e transformados em objetos que são persistidos em um banco de dados SQLite. Para a organização do projeto foi usado o padrão de design Model View Controller (MVC) que será explicado na subseção a seguir.

3.1.3 Model View Controller - MVC

O padrão MVC representa uma boa visão de design, separando o código usado na apresentação dos dados (*View*), do código que faz operações sobre os dados (*Model*) e do que gerencia os dados (*Controller*). O *Model* serve para fazer a representação dos dados bem como operações sobre eles. A *View* é a responsável por mostrar informação de forma visual. O *Controller* é responsável por determinar quais dados serão passados para quais *Models* e quais *Views* (HOLZNER, 2006).

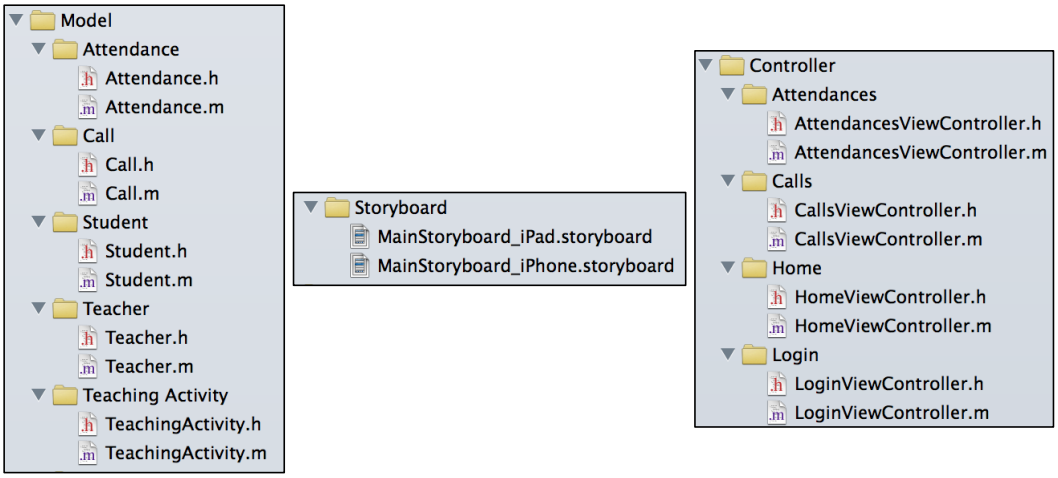

Figura 3.2: Separação de conceitos usando MVC no aplicativo iOS

Na plataforma iOS, o padrão MVC se encaixa perfeitamente. A Figura 3.2 mostra como esse padrão foi usado para separar os conceitos do aplicativo móvel. Nota-se que a pasta nomeada *Storyboard* contém os arquivos onde são implementadas as *Views* tanto para iPhone quanto para iPad. Por fim, vale ressaltar que as entidades representadas na parte de *Model* foram implementadas tanto no aplicativo quanto no *Web-Service*

3.2 Modelagem do Sistema

Nesta seção serão explicados os requisitos do sistema por meio de *User Stories*. Logo após serão apresentados os tipos de dados que serão usados para representar o banco de dados da aplicação.

3.2.1 *User Stories*

Aqui serão apresentados os requisitos do aplicativo proposto por meio de *User Stories* ou histórias de usuário em português. *User Stories*, segundo Cohn, são descrições simples e breves de uma funcionalidade da perspectiva de uma pessoa que deseja essa nova função. Essa pessoa geralmente é um usuário ou um cliente do sistema. *User Stories* costumam seguir a seguinte forma: **Sendo um <tipo de usuário>, eu quero <algum objetivo> para que <alguma finalidade>** (COHN, 2004). Nas subseções a seguir serão apresentadas as histórias de usuário criadas após reuniões realizadas com o professor Leandro Krug Wives.

3.2.1.1 User Story - 1

Sendo um usuário, eu quero fazer login utilizando um *token* gerado pelo SAV para que não seja necessário criar um novo usuário e senha.

3.2.1.2 User Story - 2

Sendo um usuário, eu quero que o aplicativo utilize meus dados de login para se autoconfigurar com as minhas turmas para que o aplicativo esteja operante com o menor esforço possível.

3.2.1.3 User Story - 3

Sendo um usuário, eu quero ver as turmas das quais sou responsável em uma lista para que eu possa escolher de qual turma farei chamada.

3.2.1.4 User Story - 4

Sendo um usuário, eu quero que ao selecionar uma turma eu possa ver uma lista com as datas das chamadas e que a data mais próxima esteja no topo da tela para que eu possa chegar rapidamente na chamada atual.

3.2.1.5 User Story - 5

Sendo um usuário, eu quero que ao selecionar uma data eu veja todos os alunos da disciplina em uma lista com seu estado atual de presente/ausente/dispensado para saber a situação de cada aluno na data selecionada.

3.2.1.6 User Story - 6

Sendo um usuário, eu quero que ao selecionar uma data eu possa alterar o estado de presença de um aluno e quando terminar enviar ao SAV para que a chamada seja feita e seu resultado persistido e disponível para os alunos.

3.2.1.7 User Story - 7

Sendo um usuário, eu quero que seja possível atualizar o aplicativo com os dados mais recentes das minhas disciplinas para que eu possa ter o conteúdo sempre atualizado depois do período de ajuste e também em semestres futuros.

3.2.1.8 User Story - 8

Sendo um usuário, eu quero que caso o envio de uma chamada falhe ela seja enviada ao SAV na próxima vez que eu abrir o aplicativo para que toda a chamada feita pelo aplicativo seja em algum momento persistida no banco do SAV.

3.2.2 Entidades

Nas subseções a seguir serão descritos os tipos de dados usados para modelar o problema. Os modelos aqui demonstrados são utilizados tanto no *Web-Service* quanto no aplicativo. Alguns atributos servem apenas para identificar de forma única uma entidade e foram herdados do banco de dados atual do SAV. Tais atributos não tem seu valor visível ao usuário e estarão marcados com o simbolo "*". A Figura 3.3 apresenta o diagrama entidade relacionamento do banco de dados.

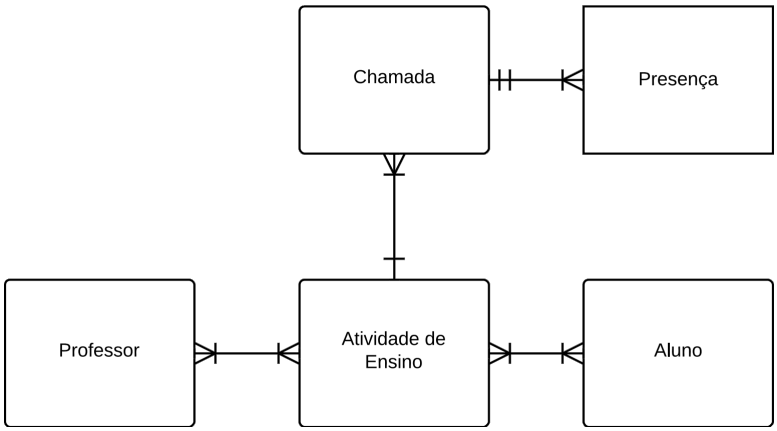

Figura 3.3: Modelo Entidade Relacionamento

3.2.2.1 Entidade Atividade de Ensino (Teaching Activity)

Esta entidade representa algum curso que será ministrado ao longo de um semestre e que possuí uma quantidade mínima .

- **calls** - Lista de todas as chamadas dessa disciplina;
- **classLetterCode** - Letra identificadora da turma, *e.g.*, "U";
- **code***;
- **configurationSequenceNumber***;
- **courseLevel** - Nível do curso, *e.g*, "G", para atividades de ensino da graduação;
- **mainTeacherCode** - Código do professor responsável;
- **mainTeacherName** - Nome do professor responsável;
- **name** - Nome da atividade de ensino, *e.g.*, "Modelos de Linguagem de Programação";
- **sequenceNumber***;
- **shortName** - Código da atividade de ensino, *e.g.*, "INF-01001";
- **students** - Lista com todos os alunos da atividade de ensino;
- **teachers** - Lista com todos os professores vinculados a atividade de ensino;
- **type***;
- **validityPeriod** - Período de vigência dessa atividade de ensino;

3.2.2.2 Entidade Aluno (Student)

Esta entidade representa um aluno de uma determinada disciplina. Servirá como referência no momento de fazer a chamada.

- **code** - Código identificador do aluno, *e.g.*, "00159963";
- **name** - Nome do aluno, *e.g.*, "Fernando Sehnem Heck";

3.2.2.3 Entidade Professor (Teacher)

Esta entidade representa um professor. Futuramente poderá ser implementado no aplicativo a capacidade de ver os professores que ministram uma certa disciplina.

- **code** - Código identificador do aluno, *e.g.*, "00159963";
- **name** - Nome do aluno, *e.g.*, "Leandro Krug Wives";
- **teacherType** - Tipo do professor, podendo ser "Professor";

3.2.2.4 Entidade Chamada (Call)

Esta entidade representa uma lista de chamada com o estado de cada aluno naquele momento. Cada aluno pode estar presente, ausente ou dispensado.

- **sequenceNumber***;
- **dateSequenceNumber***;
- **date** - Data da chamada, *e.g.*, "28/05/2014";
- **startTime** - Horário da chamada, *e.g.*, "08:30";
- **attendances** - Lista com as presenças de todos os alunos referente a essa chamada;
- **modified** - Valor de controle usado no aplicativo para determinar se uma chamada foi modificada localmente e ainda não tenha sido enviada ao SAV;

3.2.2.5 Entidade Presença (Attendance)

Esta entidade representa o estado de um determinado aluno em uma chamada.

- **studentCode** - Código identificador do aluno, *e.g.*, "00159963";
- **callingSequenceNumber*** - Código usado para identificar a qual chamada essa presença pertence;
- **callingDateSequenceNumber*** - Código usado para identificar a qual chamada essa presença pertence;
- **attendance** - Valor inteiro que indica a situação do aluno nessa chamada. As alternativas são presente, ausente e dispensado;

3.3 Resumo

No presente capítulo foram apresentadas questões sobre a arquitetura e modelagem do sistema. Ambos os pontos são comentados do ponto de vista do *Web-Service* e do aplicativo móvel.

Na seção de arquitetura, foi apresentado, um padrão de design de *software*, o MVC, que é bastante útil e comumente usado e aplicativos iOS. Foi explicado, também, como a comunicação entre o *Web-Service* e o aplicativo é realizada. A comunicação é feita através de protocolos como o SOAP e HTTPS. Lembrando que os dados trafegados dentro do envelope SOAP são serializados em formato JSON.

Na seção sobre modelagem foram apresentados os requisitos do sistema através de *User Stories* e as entidades. Tais entidades tiveram seus atributos comentados de modo a mostrar ao leitor como elas se relacionam com o problema aqui resolvido.

4 DESENVOLVIMENTO DA APLICAÇÃO

No presente capítulo serão apresentadas questões relevantes da implementação da aplicação. Primeiramente serão apresentadas as questões referentes ao *Web-Service* e logo em seguida sobre o aplicativo iPresence.

4.1 *Web-Service*

O *Web-Service* é uma parte fundamental do sistema. É através dele que o aplicativo obtém as informações necessárias para operar e por fim atualiza o SAV. A seguir serão apresentados os detalhes de sua implementação.

4.1.1 Estrutura

A estrutura do *Web-Service* é a mostrada na Figura 4.1. As o conteúdo das pastas mais importantes do ponto de vista da implementação são explicadas nas subseções a seguir.

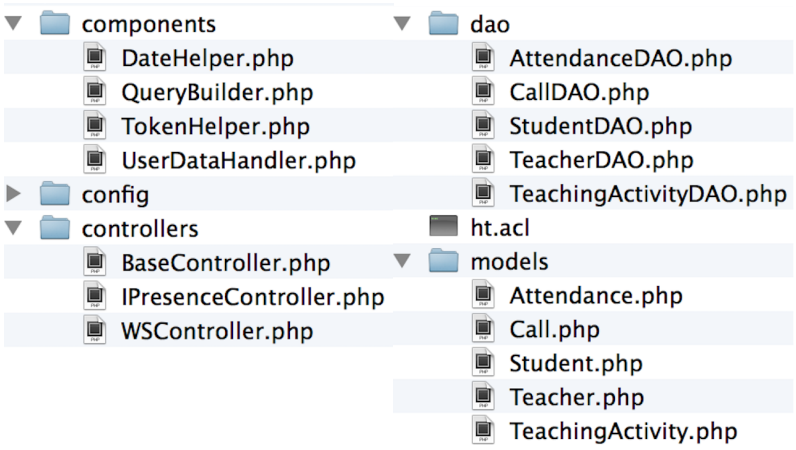

Figura 4.1: Estrutura de Arquivos

4.1.1.1 ***components***

Esta pasta contém as classes auxiliares. A classe *QueryBuilder* contém as definições de todas as *queries* usadas pelas classes da pasta *dao*. A classe *DateHelper* serve para criar uma *string* com a data do semestre atual junto com informações necessárias para pesquisas no banco de dados. A classe *TokenHelper* é responsável pela validação do

token. A classe *UserDataHandler* é a classe que auxilia o controlador nas tarefas de requisitar e atualizar dados.

4.1.1.2 ***controllers***

Esta pasta contém os controladores do serviço. A classe mais importante é a *IPresenceController* que é responsável por definir os métodos visíveis expostos pelo *Web-Service*.

4.1.1.3 ***dao***

Nesta pasta estão contidos todas as classes Data Access Object (DAO). Essas classes são desenvolvidas especificamente para uma entidade e servem para fazer as operações de pesquisa e *update* no banco de dados.

4.1.1.4 ***models***

Esta pasta contém todos os modelos usados para representar as entidade comentadas na seção 3.2.

4.1.2 Métodos

Os métodos expostos pelo *Web-Service* usam um *Token* como método de autenticação do usuário. Esse *Token* é um número gerado pelo SAV durante a configuração da lista de presença. A seguir serão apresentados os métodos implementados juntamente com os valores recebidos como parâmetros e os valores retornados.

4.1.2.1 login(token)

Descrição: O método *login* serve para determinar se o usuário pode ter acesso ao aplicativo. Para tanto o usuário deve possuir um *token* relacionado a um código de professor no banco de dados do SAV.

Parâmetro: *token* gerado através do SAV

Retorno: Existem dois tipos de retorno. Caso o login seja bem sucedido, então, o *token* recebido é enviado de volta ao aplicativo. Caso o *token* seja inválido ou seu valor não corresponda a nenhum código de professor, os valores, "INVALID_TOKEN"e "TOKEN_MATCH_ERROR"são retornados, respectivamente.

```
/**
 * @param string token
 * @return string token if user is valid and INVALID_TOKEN otherwise
 * @soap
 */
public function login($token) {
    $teacherCode = TokenHelper::getTeacherCodeForToken($token);
    if ($teacherCode == "TOKEN_MATCH_ERROR" || $teacherCode == "INVALID_TOKEN")
        return $teacherCode;
    return $token;
}
```

Figura 4.2: Método *login*

4.1.2.2 *getFullData(token)*

Descrição: Este método é usado para pegar as informações das disciplinas do professor. Esse método serve para configurar o aplicativo. Vale ressaltar que para uma disciplina constar no retorno deste método ela deve estar devidamente configurada no SAV.

Parâmetro: *token* gerado através do SAV

Retorno: JSON contendo todos os dados explicados na subseção 3.2.2.1. Em caso de problemas com o *token* os erros comentados no método de login são retornados.

```
/**
 * @param string token
 * @return string teaching activities as json or error message
 * @soap
 */
public function getFullData($token) {
    $teacherCode = TokenHelper::getTeacherCodeForToken($token);
    if ($teacherCode == "TOKEN_MATCH_ERROR" || $teacherCode == "INVALID_TOKEN")
        return $teacherCode;
    $userDataHandler = new UserDataHandler();
    $response = $userDataHandler->getTeachingActivities($teacherCode);
    return trim(json_encode($response));
}
```

Figura 4.3: Método *getFullData*

4.1.2.3 *sendCallAtendances(token, call)*

Descrição: Este método serve para atualizar o banco de dados do SAV com as informações geradas pelo aplicativo iPresence. Dessa forma as informações ficam disponíveis para consulta dos alunos.

Parâmetros: *token* gerado através do SAV. Objeto do tipo "Call", contendo informações sobre as presenças dos alunos da disciplina em uma determinada data em formato JSON.

Retorno: Caso exista algum problema com o *token*, os mesmos erros definidos no método login são retornados. Caso a autenticação seja bem sucedida esse método retorna o valor "OK"se as presenças foram atualizadas corretamente no banco de dados e retorna "ERROR"caso contrário.

```
/**
 * @param string token
 * @param string teste with attendances
 * @return string login error or OK or ERROR according to database response
 * @soap
 */
public function sendCallAttendances($token, $call) {
    $teacherCode = TokenHelper::getTeacherCodeForToken($token);
    if ($teacherCode == "TOKEN_MATCH_ERROR" || $teacherCode == "INVALID_TOKEN")
        return $teacherCode;
    $call = json_decode($call);
    $userDataHandler = new UserDataHandler();
    $attendancesIncerted = $userDataHandler->persistCallAttendances($teacherCode, $call);
    $returnValue = "ERROR";
    if($attendancesIncerted == count($call->attendances))
        $returnValue = "OK";
    return $returnValue;
}
```

Figura 4.4: Método *sendCallAttendances*

4.2 iPresence

O aplicativo iPresence é o objetivo principal deste trabalho. Nas subseções a seguir serão apresentadas questões sobre sua implementação.

4.2.1 Estrutura

A estrutura do projeto é semelhante a qualquer projeto iOS e está representada na Figura 4.5. A seguir serão comentados os conteúdos das pastas mais importantes.

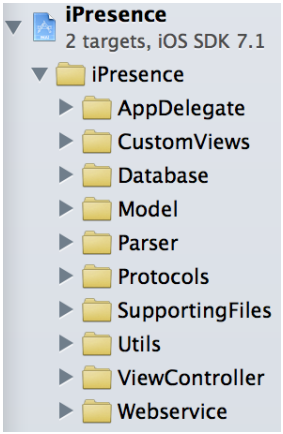

Figura 4.5: Estrutura de Arquivos do iPresence

4.2.1.1 Database

Nesta pasta estão contidas as classes que fazem a persistência dos dados em um banco de dados SQLite. Maiores informações sobre a implementação do banco de dados serão comentadas na subseção 4.2.2.

4.2.1.2 Model

Esta pasta contém todos as classes que implementam as entidades definidas na seção 3.2.

4.2.1.3 Protocols

A pasta *Protocols* é onde estão definidas todas as interfaces criadas. Em Objective-C uma interface tem o nome de *protocol*. A Figura 4.7 mostra como foi implementada a interface usada para comunicar ao controlador que o *Web-Service* retornou resultado para o método de *login*.

```
10 @protocol LoginHelperDelegate <NSObject>
11 @required
12 - (void)loginSuccess;
13 - (void)loginFailed;
14 @end
```

Figura 4.6: LoginHelperProtocol

Na linha 10, a interface é declarada com o nome *LoginHelperProtocol* e é definido

que ela herda de *NSObject*, equivalente à classe *Object* do Java. Na linha 11 está sendo informado que os métodos definidos a seguir são do tipo *required*, ou seja, a classe que implementar essa interface deve obrigatoriamente implementar esses métodos.

4.2.1.4 Utils

Esta pasta contém as classes usadas para ajudar no desenvolvimento. Dentre elas, as mais importantes são:

- *DataManager:* Esta classe provê uma abstração aos controladores com relação a fonte e destino dos dados.

```
15 @interface DataManager : NSObject <WSIPresenceDelegate, WSCallDelegate>
16 @property (nonatomic, weak) NSObject<DataManagerDelegate> *delegate;
17 + (DataManager *)sharedDataManager;
18 - (void)asyncPersistCall:(Call *)call withToken:(NSString *)token;
19 - (void)asyncFetchTeachingActivities;
```

Figura 4.7: DataManager

  Na linha 15 a classe é declarada como subclasse de *NSObject* e é definido que essa classe implementa as interfaces e *WSCallDelegate*. A interface *WSIpresenceDelegate* que é usada para comunicar a essa classe que há resposta do método *getFullData*. A interface *WSCallDelegate* é usada para informar a essa classe que há resposta do método de enviar chamadas do *Web-Service*. Na linha 16 está definido o atributo da classe que implementa a interface *DataManagerDelegate* e que será notificado das mudanças nos dados. Nas linhas 18 e 19 são declarados os métodos que persistem as chamas das no *Web-Service* e que buscam as atividades de ensino respectivamente.

- *LoginHelper:* Classe usada para verificar se o usuário informou um *token* correto. Essa classe acessa o *Web-Service*.

- *SoapHelper:* Classe criada para criar os envelopes SOAP necessários para consumir os três métodos do *Web-Service*.

4.2.1.5 ViewController

Nesta pasta se encontram os controladores criados para prover as funcionalidades das telas do aplicativo.

4.2.1.6 Webservice

Esta pasta contém as classes usadas para acessar o *Web-Service*. Tais classes fazem o acesso usando os protocolos SOAP e HTTPS.

4.2.2 Banco de Dados

O banco de dados do aplicativo iPresence foi implementado em SQLite pois possui suporte nativo na plataforma iOS. Todas as *queries* executadas são criadas em classes DAO, como as mostradas na Figura 4.8.

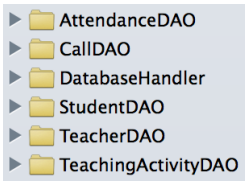

Figura 4.8: Classes do Banco de Dados

Cada entidade definida na seção 3.2 possui uma classe DAO correspondente. As classes DAO servem para criar as *queries* de busca e atualização de dados. Depois de criadas as *queries* essas são passadas a uma instância da classe *DatabaseHandler*, Figura 4.9, que por sua vez cuida das operações necessárias como abrir o banco de dados, executar as *queries* e fechar banco de dados.

```
35 @interface DatabaseHandler : NSObject
36
37 + (DatabaseHandler *)sharedInstance;
38
39 - (BOOL)createDatabase;
40
41 - (NSArray *)executeSelectQuery:(NSString *)query onComplete:(SelectBlockType)selectBlock;
42 - (BOOL)executeInsert:(NSString *)insertQuery;
43 - (BOOL)executeInserts:(NSArray *)insertQueries;
44 - (BOOL)executeUpdate:(NSString *)updateQuery;
45 - (BOOL)executeDelete:(NSString *)deleteQuery;
```

Figura 4.9: Classes do Banco de Dados

4.3 Resumo

No presente capítulo foram apresentados detalhes da implementação tanto do *Web-Service* quanto do aplicativo iPresence. Em ambos os casos foram comentados os conteúdos de pastas importantes para os projetos. Algumas classes mais interessantes foram detalhadas mais aprofundadamente.

5 FUNCIONAMENTO DO SISTEMA

Neste capítulo serão apresentadas as telas do sistema com as quais o usuário irá interagir. Primeiramente, será exibida uma pequena reformulação realizada pelo CPD na plataforma SAV. Logo em seguida serão expostas as visões do iPresence bem como uma breve descrição das funcionalidades presentes.

5.1 Sala de Aula Virtual - SAV

O Sala de Aula Virtual (SAV), como dito anteriormente, é uma plataforma desenvolvida peloCentro de Processamento de Dados (CPD) UFRGS que visa facilitar a troca de informação entre alunos e professores. O presente trabalho tem forte integração com o SAV, desse modo essa plataforma é também responsável pela geração do *Token* que autentica o usuário. Na subseção a seguir serão mostradas as alterações que foram necessárias no SAV como consequência do presente trabalho.

5.1.1 Geração do *Token*

A geração do *Token* é feita durante a configuração da lista de chamadas no SAV. A Figura 5.1 mostra a interface remodelada para mostrar um *Token* ao professor que esteja configurando a lista de chamadas e que deseja realizar a chamada através do aplicativo móvel.

Figura 5.1: Visão Geral do Sistema

5.2 iPresence UFRGS

O aplicativo **iPresence UFRGS** conta com quatro telas simples. O aplicativo possui poucas telas e todas elas foram projetadas de modo a levar o usuário o mais rápido possível para a tela onde são marcadas as presenças dos alunos. As próximas seções mostram as telas do aplicativo e explicam a funcionalidade presente em cada uma.

5.2.1 Login

A tela de login é a primeira visível no aplicativo. A informação que deve ser inserida é o *Token*, previamente gerado no sistema SAV. A função dessa tela é a de autenticar o usuário usando o mínimo de informações pessoais possíveis. Logo após o primeiro login bem sucedido o aplicativo acessa o *Web-Service* para buscar todos os dados necessários do professor e deixar o sistema pronto para o uso.

Figura 5.2: Tela de Login

5.2.2 Atividades de Ensino

A tela de atividades de ensino é a segunda tela a ser apresentada. Essa tela aparece quando o login é bem sucedido. Aqui encontram-se todas as atividades de ensino as quais o professor tem permissão de fazer a chamada. As atividades de ensino estão agrupadas de acordo com o seu tipo, como por exemplo, "Graduação". Vale lembrar que serão apresentadas aqui apenas as atividades de ensino que foram corretamente configuradas no SAV.

5.2.3 Chamadas

Nesta tela são listadas todas as chamadas da disciplina ordenadas por data. Para comodidade do professor a chamada mais próxima é apresentada mais ao topo.

Figura 5.3: Tela de Atividades de Ensino

5.2.4 Presenças

A lista de presenças contém o estado de cada aluno em uma determinada chamada. Nessa tela é possível que o professor altere o estado de um aluno. Para que se possa editar as presenças dos alunos basta apertar no botão "Editar"que um menu de auxilio aparece e todas as células se tornam clicáveis. Apenas células que possuem o valor presente e ausente podem ter seu estado alterado.

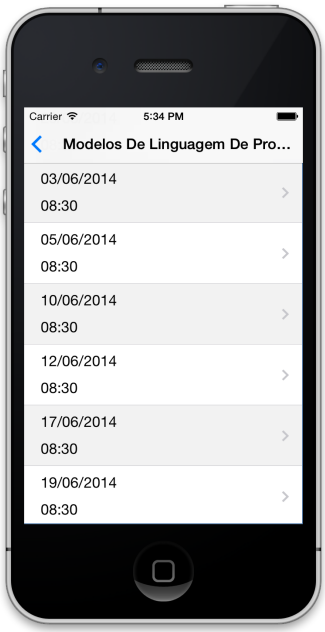

Figura 5.4: Tela de Chamadas

Figura 5.5: Tela de Presenças

6 CONCLUSÃO

Durante o desenvolvimento do trabalho deixou-se claro a importância do mercado móvel em um panorama mundial. Outro ponto levantado durante o trabalho é a tarefa tediosa porém necessária dos professores de marcar a presença dos alunos. Por esses motivos foi criado o Sistema Móvel de Controle de presença. Esse sistema tem o intuito de facilitar o dia a dia dos professores e também de ser o primeiro aplicativo móvel da UFRGS em parceria com o CPD.

Antes de iniciar o trabalho foi realizada uma pesquisa de aceitação com os professores do Instituto de Informática da UFRGS. Ao analisar os resultados foi verificado que praticamente todos os professores do Instituto possuem um dispositivo móvel ou Android ou iOS. Foi verificado também que para um possível novo sistema de controle de presenças esses mesmos professores preferem um sistema tanto *Web* quando por aplicativo móvel. De posse dos resultados foi decidido, então, criar um aplicativo integrado ao SAV.

Após definidos os objetivos e realizada a pequisa de aceitação foram apresentados estudos sobre as plataformas moveis existentes e também sobre os modos diferentes de se desenvolver um aplicativo. Tais estudos foram usados de modo a justificar a escolha de um aplicativo iOS e também do modo de desenvolvimento nativo. Em seguida foram apresentados os requisitos do sistema e a definição da arquitetura. A arquitetura dividiu o sistema em três partes, o SAV, o *Web-Service* e o aplicativo iPresence. O *Web-Service* foi pensado de modo que possa ser consumido por qualquer plataforma. Depois disso foram apresentados os detalhes do desenvolvimento e também uma apresentação das telas criadas no aplicativo.

Como resultado desse trabalho surgiu o Sistema Móvel de Controle de Presença. Esse trabalho pode ser visto como um marco para a Universidade, visto que, conta com o primeiro aplicativo criado em parceria com o CPD.

6.1 Trabalhos Futuros

Por fim, esse trabalho permite diversas possibilidades, visto que foi aberto um precedente para a utilização de tecnologias móveis em parceria com serviços da Universidade. Algumas ideias para trabalhos futuros são:

- Multi plataformas. Durante a realização deste trabalho foi desenvolvido um *Web-Service* que é usado para sincronizar o aplicativo iPresence com o SAV. Tal *Web-Service* foi construído de forma genérica de modo que é possível usá-lo para aplicativos em outras plataformas.

- Mais serviços. Uma possível extensão desse trabalho seria incluir mais serviços tanto para alunos quanto para professores. Tais serviços poderiam ser dar/ver notas de trabalhos e provas, fazer a matricula e quaisquer outros serviços importantes da Universidade.

REFERÊNCIAS

BLACKBERRY, W. Disponível em: http://developer.blackberry.com/. Acesso em: 13/11/2013.

BUDIU, R. **Mobile**: native apps, web apps, and hybrid apps. Disponível em: http://www.nngroup.com/articles/mobile-native-apps/. Acesso em: 05/05/2014.

COHN, M. **User stories applied**: for agile software development. [S.l.]: Addison-Wesley Professional, 2004.

FLING, B. **Mobile design and development**: practical concepts and techniques for creating mobile sites and web apps. [S.l.]: O'Reilly Media, Inc., 2009.

HOLZNER, S. **Design Patterns For Dummies**. [S.l.]: LibreDigital, 2006.

IDC. **Tablet Shipments Forecast to Top Total PC Shipments in the Fourth Quarter of 2013 and Annually by 2015**. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS24314413. Acesso em: 13/11/2013.

IDC. **Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains**. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS24257413. Acesso em: 13/11/2013.

PETZOLD, C. **Microsoft XNA Framework Edition**: programming windows phone 7. [S.l.]: Microsoft press, 2010.

STEELE, J.; TO, N. **The Android developer's cookbook**: building applications with the android sdk. [S.l.]: Pearson Education, 2010.

VERGE. **OS**: a visual history. Disponível em: http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad. Acesso em: 13/11/2013.

ZUCKER, D.; RISCHPATER, R. **Beginning Nokia Apps Development**: qt and html5 for symbian and meego. [S.l.]: Apress, 2010.