Introdução à Arquitetura Apple iOS

Adriano Mendonça Rocha[1], **Roberto Mendes Finzi Neto**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Goiás (UFG) – Campus Catalão
Avenida Dr. Lamartine Pinto de Avelar – 1120 – Setor Universitário –
CEP: 75704-020 – Catalão – GO – Brasil

adriano.comp@hotmail.com, robertofinzi@hotmail.com

Abstract. *This paper aims to present a study about the iOS operating system architecture, approaching the main frameworks present in the layers that form the architecture of the iOS.*

Resumo. *Este artigo tem como objetivo apresentar um estudo sobre a arquitetura do sistema operacional iOS, abordando os principais frameworks presentes nas camadas que formam a arquitetura do iOS.*

1. Introdução

Esse artigo tem como objetivo apresentar um estudo sobre a arquitetura do sistema operacional (iPhone *Operating System*) iOS que roda nos dispositivos móveis da Apple Inc.: iPhone, iPad e iPod touch, abordando os principais *frameworks* presentes nas camadas que compõem a arquitetura do iOS. Também será feito uma comparação com os principais sistemas operacionais para dispositivos móveis.

Desenvolvedores que ainda não conhecem a plataforma iOS podem utilizar esse artigo como base. É muito importante entender como é organizada a arquitetura do iOS antes mesmo de começar a desenvolver aplicativos para essa plataforma. Conhecendo o conjunto de *frameworks* presente nas camadas do iOS, o desenvolvedor será capaz de desenvolver aplicativos completos e robustos.

Outra vantagem de se utilizar os *frameworks* oferecidos pelas camadas da arquitetura do iOS, é que as funções já foram testadas, assim o desenvolvedor ganha tempo durante o processo de desenvolvimento. Além disso, esses *frameworks* oferecem uma portabilidade maior, o desenvolvedor pode escrever um programa para o iPhone utilizando esses *frameworks*, e depois rodar o mesmo programa no iPad sem problemas.

2. Arquitetura do iOS

A arquitetura do iOS é formada por quatro camadas, sendo que cada uma delas oferece um conjunto de *frameworks* que podem ser utilizados durante o desenvolvimento de aplicativos para os dispositivos móveis da Apple Inc..

A arquitetura do iOS é semelhante à arquitetura básica encontrada no Mac OS X [Anvaari and Jansen 2010] e [Grissom 2008]. No nível mais alto, o iOS atua como intermediário entre o *hardware* subjacente e os aplicativos que aparecem na tela, como mostrado na Figura 1.

Os aplicativos desenvolvidos para o iOS raramente se comunicam diretamente com o *hardware* do dispositivo, ao invés disso, os aplicativos se comunicam com o *hardware* através de um conjunto de interfaces de sistema bem definidas que protegem seu aplicativo de alteração de *hardware* [Apple 2008].

O iOS foi projetado para atender às necessidades de um ambiente móvel. Desenvolvedores do Mac OS X vão encontrar muitas tecnologias familiares, mas também vão encontrar tecnologias que só estão disponíveis no iOS [Gonzalez-Sanchez and Chavez-Echeagaray 2010] e [Rogers 2009].

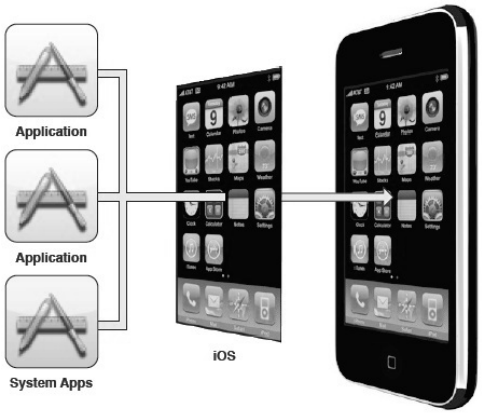

Figura 1. Aplicações em camadas em cima do iOS [Apple 2008]

A arquitetura do iOS é formada pelas camadas: *Core OS, Core Services, Media e Cocoa Touch* [Yates 2010], como mostrado na Figura 2.

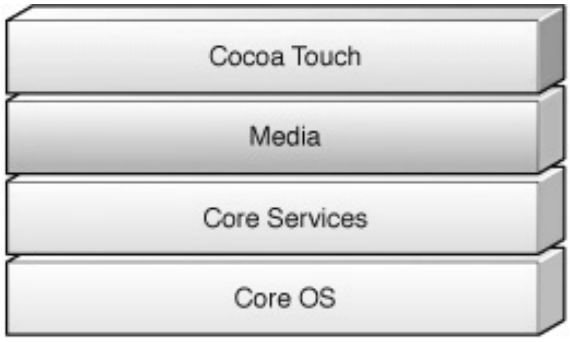

Figura 2. Camadas do iOS [Yates 2010]

Nas camadas superiores estão as tecnologias e serviços mais sofisticados. O desenvolvedor deve olhar primeiro, sempre que possível, os serviços das camadas superiores, pois nestas camadas estão os *frameworks* que fornecem abstração orientada a objetos das camadas de níveis inferiores. Estas abstrações geralmente facilitam o processo de escrita de código, pois reduzem a quantidade de código que o desenvolvedor tem que escrever, e encapsula características complexas, tais como *threads*.

Nas camadas inferiores do sistema estão os serviços fundamentais e as tecnologias dos quais todos os aplicativos dependem. Embora as tecnologias de níveis superiores resumam as tecnologias de níveis inferiores, os desenvolvedores ainda podem usar essas útimas que não estão presentes nas camadas superiores [Apple 2010].

2.1. Camada Cocoa Touch

Os principais *frameworks* para a construção de aplicações são encontrados na camada *Cocoa Touch*. Esta camada define a infra-estrutura para as tecnologias fundamentais, tais como multitarefa, serviço de notificação Apple *push* e diversos serviços de alto nível do sistema.

Ao projetar um aplicativo, os desenvolvedores devem investigar as tecnologias presentes nesta primeira camada para ver se elas atendem às suas necessidades. Nessa camada as principais tecnologias disponíveis são: multitarefa, proteção de dados e serviço de notificação Apple *push*.

Quando o usuário pressiona o botão *Home* do iPhone, por exemplo, o aplicativo não termina, em vez disso, ele muda para um contexto de execução em segundo plano. Essa característica da multitarefa é muito importante para preservar a vida da bateria, a maioria dos aplicativos ficam suspensos pelo sistema quando entram em segundo plano. A aplicação suspensa permanece na memória, mas nenhum código é executado. Esse comportamento permite que um aplicativo retome rapidamente quando ele é reiniciado, sem consumo de bateria no mesmo período [Apple 2010].

A proteção de dados permite que os aplicativos que trabalham com dados confidenciais do usuário aproveitem o sistema de criptografia disponível em alguns dispositivos. Quando um aplicativo designa um arquivo específico como sendo protegido, o sistema armazena o arquivo no disco em um formato criptografado. Enquanto o dispositivo estiver bloqueado, o conteúdo do arquivo é inacessível para o aplicativo e todos os intrusos em potencial. No entanto, quando o aparelho é desbloqueado pelo usuário, uma chave de decodificação é criada para permitir que o aplicativo acesse esse arquivo.

O serviço de notificação Apple *push* fornece uma maneira de alertar os usuários de novas informações, mesmo quando o aplicativo não está ativo. Através deste serviço, o desenvolvedor pode adicionar notificações de texto, adicionar um emblema no ícone do aplicativo, ou acionar alertas sonoros nos dispositivos do usuário. Essas mensagens fazem com que os usuários saibam que eles deveriam abrir o aplicativo para receber novas informações.

2.2. Camada *Media*

A camada *Media* contém as tecnologias de gráfico, áudio e vídeo. As tecnologias nessa camada foram projetadas para tornar mais fácil a implementação de aplicativos multimídia.

Os *frameworks* de nível superior oferecem tecnologias que tornam mais fácil a criação de gráficos e animações, enquanto os *frameworks* de nível inferior permitem o acesso às ferramentas fundamentais que o desenvolvedor pode utilizar para criar aplicativos mais robustos e complexos [Apple 2008].

2.2.1. Tecnologias de Gráficos

Nesta camada o desenvolvedor pode utilizar o *frameworks* (*User Interface Kit*) UIKit que oferece várias tecnologias de gráficos e animações. Se o aplicativo exigir uma animação simples, o sistema pode fazer isso com facilidade, se houver situações onde o desenvolvedor precise ir além dos gráficos simples, ele pode usar as seguintes tecnologias:

- *Core Graphics* lida com renderização de vetores com base em imagens 2D.
- *Core Animation* fornece suporte avançado para animar visualizações e outros conteúdos.
- *OpenGL ES* oferece suporte para renderização 2D e 3D acelerados por *hardware* usando interfaces.
- Texto *Core* fornece um layout de texto sofisticado e motor de renderização.
- *Image I/O* fornece interfaces de leitura e escrita para maioria dos formatos de imagem.
- O *framework Assets Library* fornece acesso a fotos e vídeos na biblioteca de imagens e vídeos do usuário.

2.2.2. Tecnologias de Áudio

As tecnologias de áudio disponíveis na camada *Media* fornecem uma capacidade de reproduzir e gravar áudio de alta qualidade, além de dispor de recursos de vibração em determinados dispositivos.

O sistema oferece várias maneiras para reproduzir e gravar conteúdo de áudio. Os frameworks na lista a seguir estão ordenados do alto nível para o baixo nível. Ao escolher uma tecnologia de áudio, o desenvolvedor tem que ter em mente que os *frameworks* de níveis superiores são mais fáceis de usar e são geralmente preferidos. Os *frameworks* de níveis inferiores ofereçam mais flexibilidade e controle, mas exigem mais trabalho da parte do desenvolvedor.

- O *framework Media Player* fornece acesso fácil à biblioteca do iTunes do usuário e suporte para a reprodução de faixas e playlists.
- O *framework AV Foundation* fornece um conjunto de interfaces escritas em *Objective-C* para o gerenciamento de reprodução e gravação de áudio.
- Os *frameworks Core Audio* oferecem interfaces simples e sofisticadas para reproduzir e gravar conteúdo de áudio. O desenvolvedor pode usar essas interfaces para reproduzir sons de alerta do sistema, provocar vibrações no dispositivo e reproduzir conteúdo local ou *streaming* de áudio.

2.2.3. Tecnologias de Vídeo

Além das tecnologias de áudio, a camada *Media* oferece tecnologias para reproduzir e gravar conteúdo baseado em vídeo. Em dispositivos com o *hardware* de vídeo apropriado, o desenvolvedor pode usar essas tecnologias para capturar e incorcorar vídeos em sua aplicação. Os *frameworks* na lista a seguir estão ordenados do alto nível para o baixo nível.

- A classe *UI ImagePickerController* presente no *framework* UIKit fornece uma interface padrão para gravação de vídeo em dispositivos com câmera.

- O *framework Media Player* fornece um conjunto de interfaces que podem ser usadas para apresentar filmes completos ou parciais no seu aplicativo.

- O *framework AV Foundation* fornece um conjunto de interfaces em *Objective-C* para o gerenciamento de captura e reprodução de filmes.

- *Core Media* descreve os tipos de dados de baixo nível usados pelos *frameworks* de níveis superiores e oferece interfaces de baixo nível para a manipulação de mídias.

As tecnologias de vídeos no iOS suportam a reprodução de arquivos de filme com as extensões de arquivo mov, mp4, m4v e .3gp.

2.3. Camada *Core Services*

A camada *Core Services* contém os serviços fundamentais do sistema que todos os aplicativos utilizam. Mesmo se o desenvolvedor não usar esses serviços diretamente, muitas partes do sistema são construídas em cima deles [Apple 2010]. As principais tecnologias disponíveis na camada *Core Services* são: *grand central dispatch*, *in-app purchase*, *SQLite* e *XML support*.

Grand central dispatch é uma tecnologia que o desenvolvedor pode utilizar para gerenciar a execução de tarefas em seu aplicativo. Essa tecnologia combina um modelo de programação assíncrona, com um núcleo altamente otimizado para oferecer uma alternativa mais eficiente para *threading*. O *grand central dispatch* também fornece alternativas para muitos tipos de tarefas de baixo nível, como ler e escrever em arquivos, implementação de temporizadores e monitoramento de sinais e eventos do processo.

In-app dispatch é uma tecnologia que os desenvolvedores podem utilizar para vender seus conteúdos e serviços dentro de suas aplicações. Este recurso é implementado usando a estrutura *Kit Store*, que fornece a infra-estrutura necessária para processar as transações financeiras usando a conta do iTunes do usuário.

A biblioteca *SQLite* permite incorporar um banco de dados *SQL* leve em sua aplicação, sem ter que executar um processo de servidor de banco de dados remoto. O desenvolvedor pode criar arquivos de banco de dados local e gerenciar as tabelas e registros nesses arquivos a partir do seu aplicativo. Essa biblioteca é otimizada para fornecer acesso rápido aos registros do banco de dados.

A classe *NSXMPLParser* pode ser usada para recuperar elementos de um documento XML. A biblioteca *libxml2* fornece suporte adicional para a manipulação de

conteúdo XML, além de fornecer suporte adicional para a transformação do conteúdo XML para HTML.

2.4. Camada *Core OS*

A camada *Core OS* contém características de baixo nível que foram utilizadas na implementação de outras tecnologias. Em situações onde o desenvolvedor precisa lidar explicitamente com segurança ou comunicação com acessório de *hardware* externo, ele pode fazer isso utilizando os *frameworks* nessa camada.

O *framework Accelerate* é uma tecnologia presente na camada *Core OS* que contém interfaces para a realização de cálculos matemáticos. A vantagem de usar esse *framework* são as interfaces otimizadas para todas as configurações de *hardware* presentes em dispositivos baseados no iOS. Portanto, o desenvolvedor pode escrever seu código uma vez e ter certeza que ele será executado de forma eficiente em todos os dispositivos.

O *framework External Accessory* é uma tecnologia presente na camada *Core OS* que fornece interfaces de comunicação com acessórios de *hardware* conectados a um dispositivo baseado no iOS. Os acessórios podem ser conectados através de um conector *dock* de 30 pinos, ou através do *Bluetooth*. Através das interfaces desse *framework* o desenvolvedor pode manipular o acessório diretamente usando os comandos que ele suporta.

No nível do sistema está presente o ambiente *kernel*, *drivers* e interfaces de baixo nível do sistema operacional UNIX. A biblioteca *kernel* é responsável por todos os aspectos do sistema operacional. Ela gerencia o sistema de memória virtual, *threads*, sistema de arquivos, rede e comunicação entre processos. Os *drivers* nesta camada também fornecem a interface entre o *hardware* disponível e os *frameworks* do sistema. O iOS fornece um conjunto de interfaces para acessar muitas características de baixo nível do sistema operacional. O aplicativo acessa esses recursos através da biblioteca *Libsystem*. As interfaces são baseadas em C e fornece suporte para o seguinte:

- *Threading*
- *Networking*
- Acesso ao sistema de arquivos
- *Standard I / O*
- *Bonjour* e serviços DNS
- Informação *Locale*
- Alocação de memória
- Cálculos matemáticos

3. Comparação com Outros Sistemas Operacionais para Dispositivos Móveis.

O desenvolvimento de aplicativos para dispositivos móveis vem crescendo em um ritmo acelerado, e a demanda por profissionais qualificados é enorme. Nesse artigo foi proposto

um estudo da arquitetura do sistema operacional iOS que roda nos dispositivos móveis da empresa Apple Inc., focando nos frameworks presente nas camadas que compõem a arquitetura do iOS.

Existem outros sistemas operacionais para dispositivos móveis, tais como: Android, Tizen e Symbian OS. Nessa seção será feito uma comparação entre o sistema operacional iOS e os sistemas operacionais citados acima.

O iOS atua como um intermediário entre o hardware e os aplicativos instalados nos dispositivos, desta forma, aplicativos desenvolvidos para essa plataforma podem ser executados em diferentes dispositivos com o iOS rodando, garantindo uma portabilidade maior. Já o Android tem problemas em relação à portabilidade. Quando um aplicativo desenvolvido para smartphone que roda android é executado em dispositivos com telas maiores, o layout das aplicações é desconfigurado.

O iOS foi projetado para rodar tanto aplicativos nativos quanto aplicativos web, isso é vantajoso, pois certas regiões ainda não oferecem serviço de internet para dispositivos móveis com qualidade, assim os aplicativos nativos instalados no dispositivo de armazenamento do aparelho podem ser executados independentemente se há internet ou não. Já o Tizen, novo sistema operacional da Linux Foundation em conjunto com a Limo Foundation será centralizado em aplicativos web, assim tirando a necessidade dos aplicativos serem instalados nos dispositivos. Como os aplicativos para Tizen dependem da internet, nas regiões onde a infra-estrutura desse serviço ainda é limitada, os usuários desse sistema operacional terão dificuldades em acessar os aplicativos.

O Symbian OS é um sistema operacional que roda na maioria dos telefones móveis modernos. Uma das vantagens do Symbian OS é a capacidade de evitar ao máximo o desperdício dos recursos do dispositivo, como bateria e memória. O iOS também tem essa característica, os aplicativos que não estão sendo utilizados ficam em segundo plano, ou seja, os aplicativos ficam na memória, mas nenhum código é executado, preservando a vida da bateria do dispositivo.

4. Conclusão

A demanda por aplicativos para os dispositivos móveis da Apple Inc. é enorme, já foram baixados mais de dez bilhões de aplicativos na App Store, serviço criado pela Apple Inc. que permite aos usuários navegar e baixar aplicativos. Os primeiros passos para desenvolver aplicativos para esses dispositivos é entender as camadas que compõem a arquitetura do iOS.

As tecnologias oferecidas pelas camadas do iOS tornam o processo de desenvolvimento mais rápido e seguro. Os desenvolvedores podem utilizar as tecnologias das camadas superiores que oferecem abstrações orientada a objetos, tornando a programação mais simples e eficiente. Se as tecnologias das camadas superiores não forem suficientes, os desenvolvedores ainda podem utilizar as tecnologias das camadas inferiores que são mais flexíveis e permite um maior controle sobre os recursos dos dispositivos.

Referências

Anvaari, M. and Jansen, S. (2010). Evaluating architectural openness in mobile software platforms. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 85–92, New York, NY, USA. ACM.

Apple, I. (2008). *iPhone OS Programming Guide*. Apple Inc.

Apple, I. (2010). *iOS Technology Overview*. Apple Inc.

Gonzalez-Sanchez, J. and Chavez-Echeagaray, M. E. (2010). iphone application development. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, SPLASH '10, pages 321–322, New York, NY, USA. ACM.

Grissom, S. (2008). iphone application development across the curriculum. *J. Comput. Small Coll.*, 24:40–46.

Rogers, M. (2009). It's for you!: an iphone development primer for the busy college professor. *J. Comput. Small Coll.*, 25:94–101.

Yates, II, M. (2010). Practical investigations of digital forensics tools for mobile devices. In *2010 Information Security Curriculum Development Conference*, InfoSecCD '10, pages 156–162, New York, NY, USA. ACM.