Desenvolvendo Jogos para iOS

Lucas Menge


Cookie Rush
• Versão iPhone
• Versão iPad
• Versão lite
Undead Attack! Pinball
• Versão iPhone
• Versão iPad
• Versão lite

Um longo caminho…


AT&T
9:42 AM
Text
Calendar
Tuesday
9
Photos
Camera
YouTube
Stocks
Maps
280
Weather
73°
Clock
Calculator
Notes
Settings
iTunes
Phone
Mail
Safari
iPod

iPhone

2007

Tela multitoque de 3.5” (320x480)

CPU ARM 620MHz

GPU PowerVR MBX Lite (OpenGL ES 1.1)

128MB de RAM

WiFi 802.11b/g

Câmera de 2MP (Foco fixo)

iPhone 3G

2008

3G

GPS

App Store

iPhone 3GS

2009

Mais rápido (ARMv6)

256MB de RAM

Melhor câmera (vídeos+3MP)

Bússola

OpenGL ES 2.0

iPhone 4

2010

Chip A4 (800MHz)

512MB de RAM

Melhor câmera (5MP+720p)

Câmera frontal

Tela “Retina”

Giroscópio

iPhone 4S

2011

Chip A5 (dual-core)

Melhor câmera (8MP)

Siri

iCloud

iPhone 5

2012

Chip A6 (ARMv7)

1GB de RAM

Tela de 4”

Melhor câmera (1080p+IS)

LTE (não no Brasil)

Conector Lightning

iPhone 5c

2013

Melhor câmera frontal

iPhone 5s

2013

Chip A7 (64-bits)

Chip M7

Câmera de 120fps

TouchID

OpenGL ES 3.0

iPod Touch

Lançado todo ano após o iPhone

Herda funções do iPhone

Sempre com CPU/GPU da geração anterior

iPod Touch

2012

Chip A6

512MB de RAM

Tela de 4”

Conector Lightning

iPad

2010

“Só um iPod Touch grande”

CPU A4

256MB de RAM

Tela de 10” (1024x768)

WiFi 802.11b/g/n [+3G]

iPad 2
2011

Chip A5 (dual-core)

512MB de RAM

Câmeras traseira e frontal

Giroscópio

iPad (3a Geração)
2012

Chip A5X (GPU dobrado)

1GB de RAM

Tela “Retina”

Melhor câmera (720p)

LTE (não no Brasil)

iPad (4a Geração)
2012

Chip A6X

Melhor câmera frontal

Conector Lightning

iPad Air

2013

Chip A7 (1.4GHz)

OpenGL ES 3.0

Borda menor

Quase 200g mais leve

iPad Mini

2012

iPad 2 de 7.9”

iPad Mini with Retina Display

2013

iPad Air de 7.9”

CPU a 1.3GHz ao invés de 1.4GHz

Apple TV

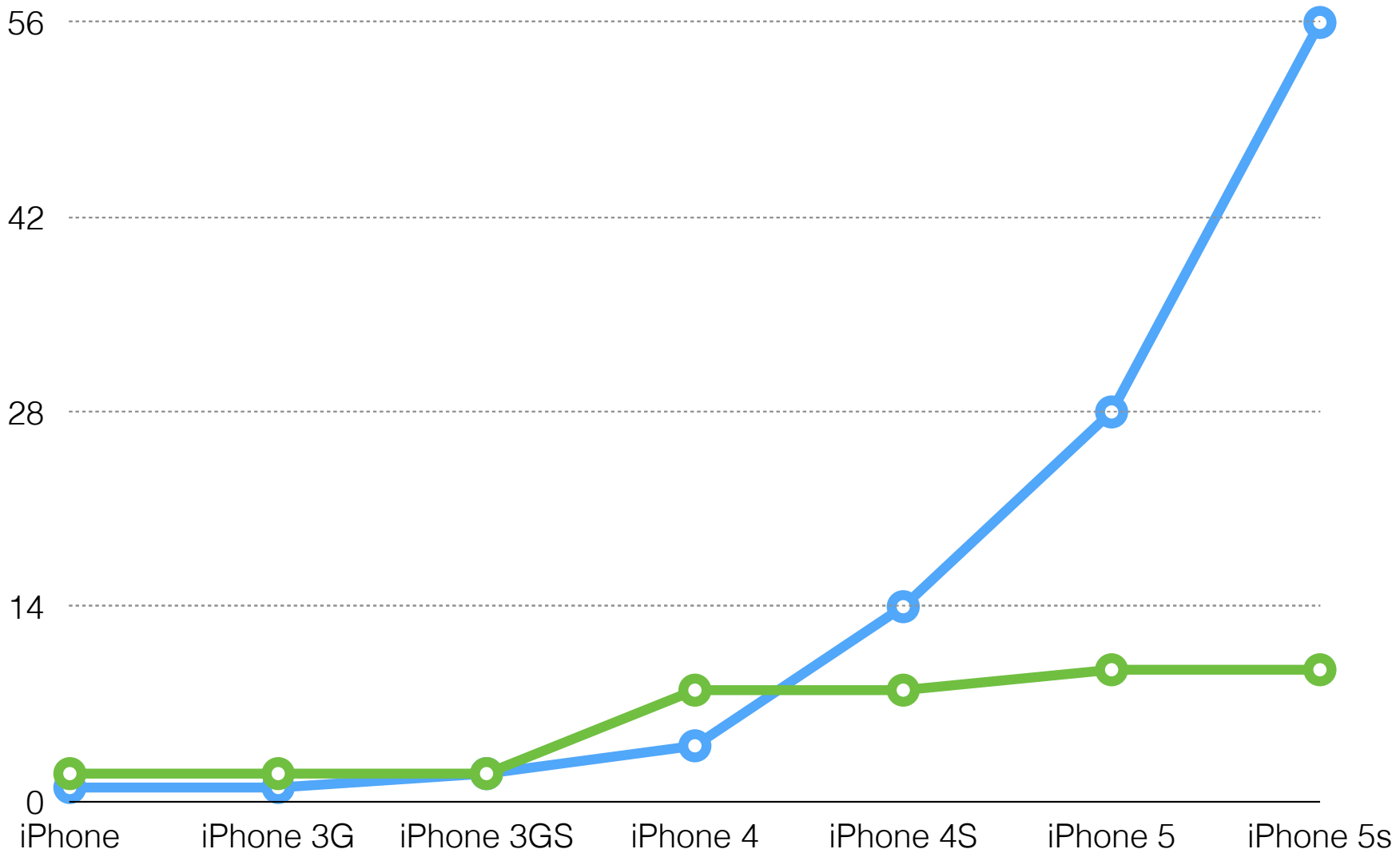
Sempre feito com as “sobras” dos iPads

Saída HDMI e TOSLINK

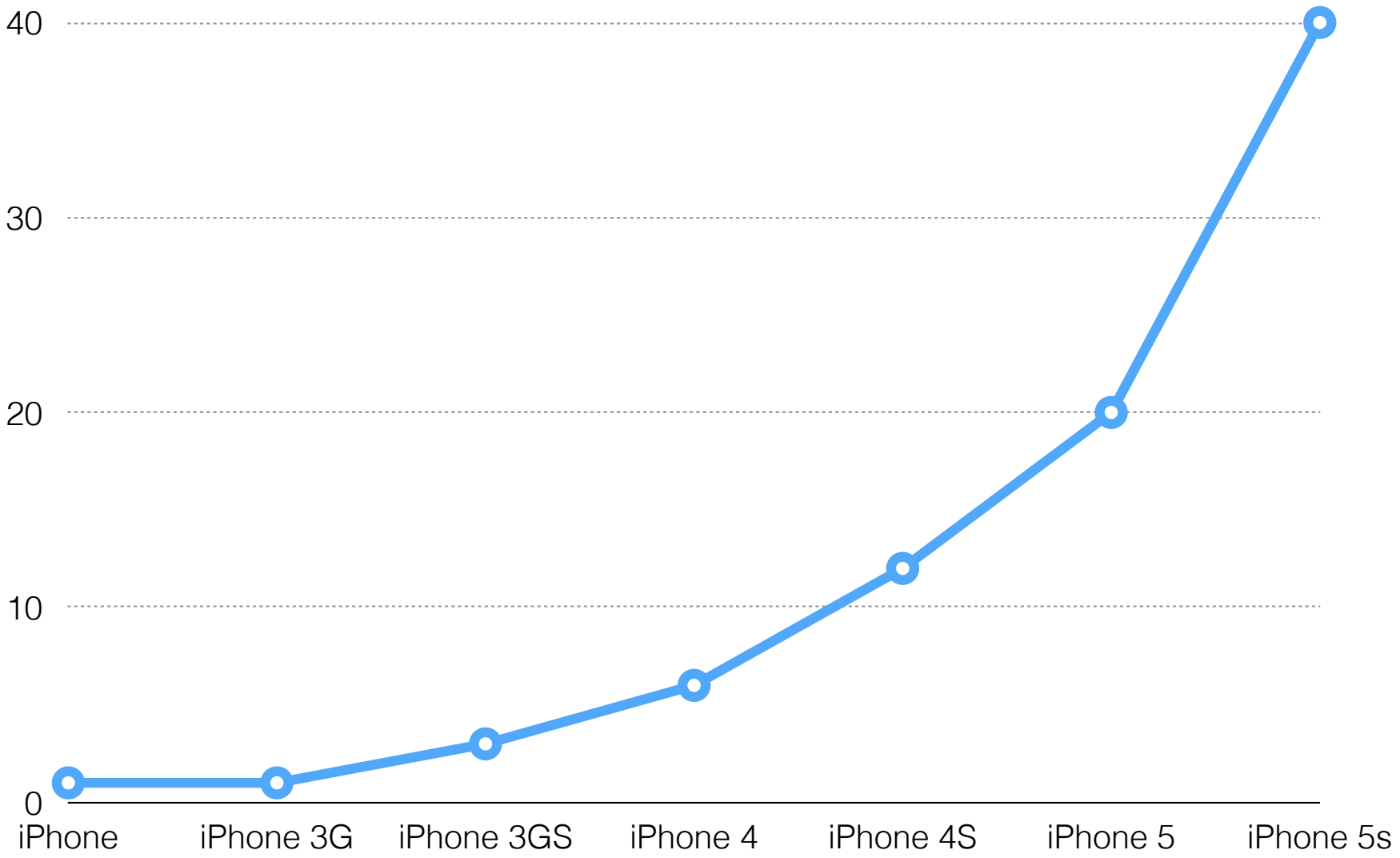
AirPlay Mirroring

MetalStorm: Wingman

Performance de GPU



Performance de CPU




10
TERRY O.

iOS em Números

- 600 milhões de aparelhos iOS vendidos
- Usuários iOS gastam até 5x o que usuários Android gastam
- Crescimento de 26% nas vendas de iPhones em relação ao ano passado
- 34 milhões de iPhones vendidos só em Q3 2013

iOS em Números

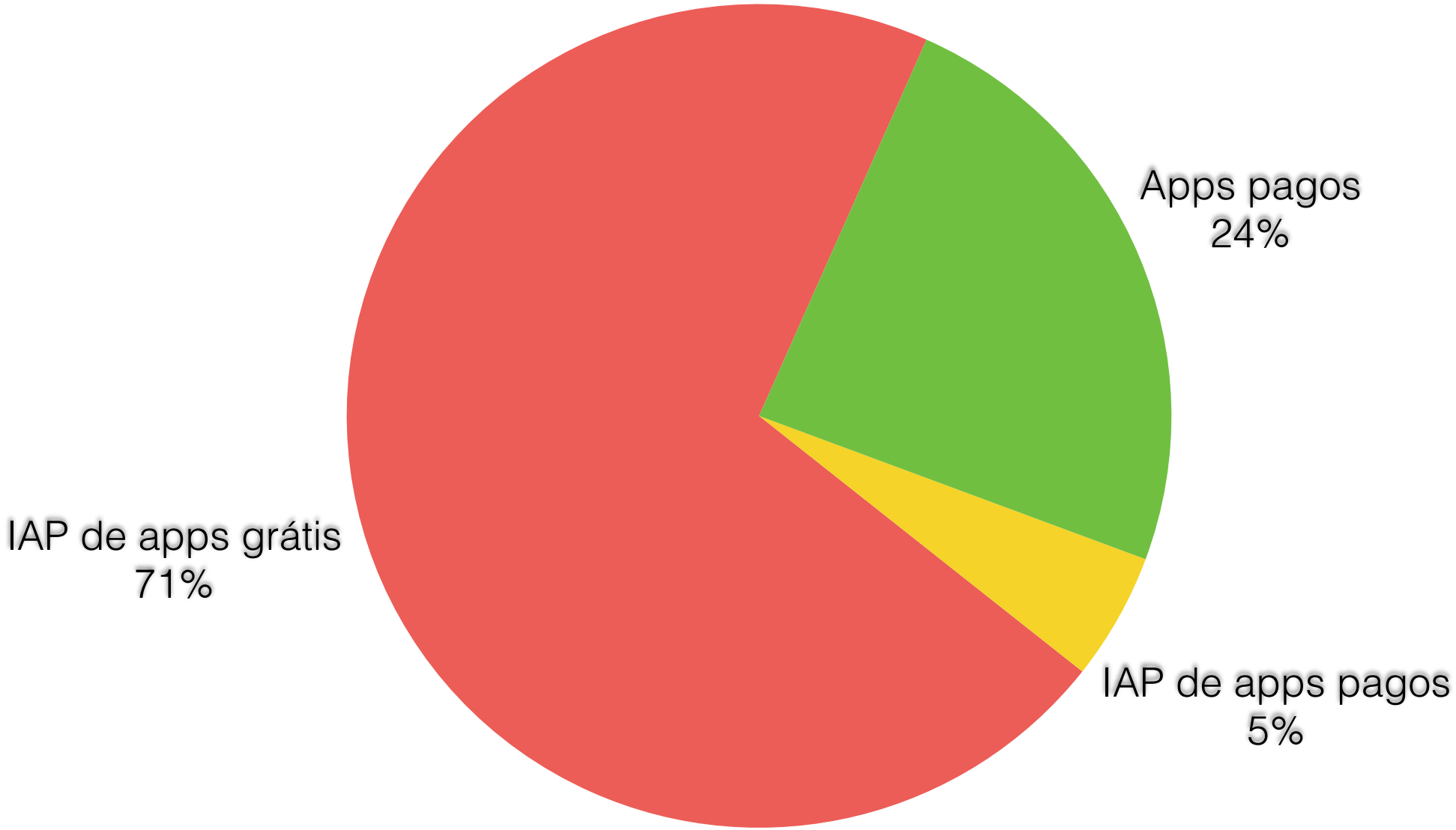
- 1 milhão de apps na App Store
- 375,000+ apps feitos para o iPad
- 90% dos apps são baixados ao menos 1x/mês
- 575 milhões de contas na App Store
- 240 milhões de usuários do Game Center

iOS em Números

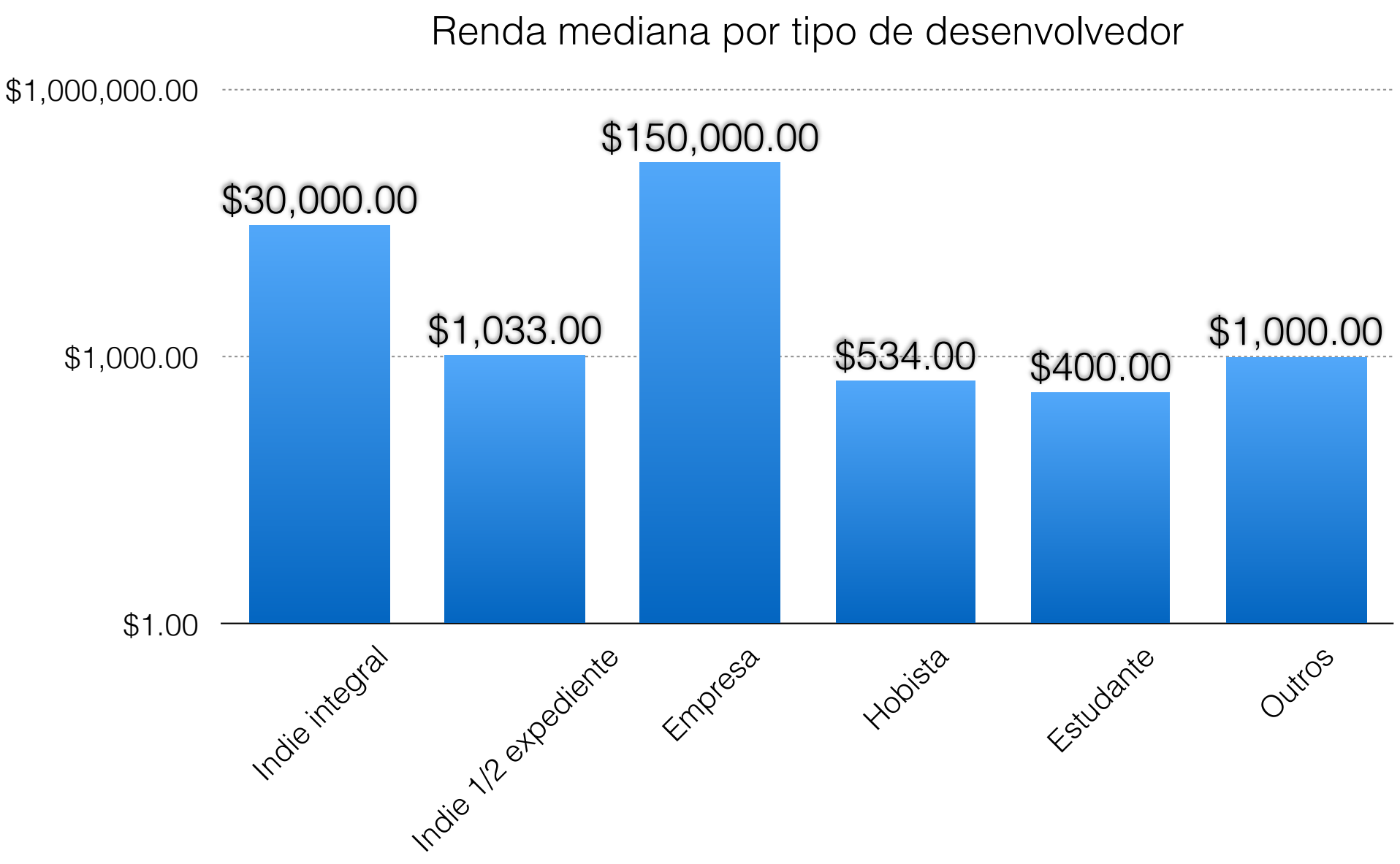
- Apple pagou US$5 bilhões a desenvolvedores só no último ano
- 40% dos downloads da App Store são jogos
- 70% da renda da App Store vem de jogos
- Preço médio de um jogo: US$0.75
- No dia em que o UAP foi lançado, 400 jogos também foram

Freemium ao “Resgate”

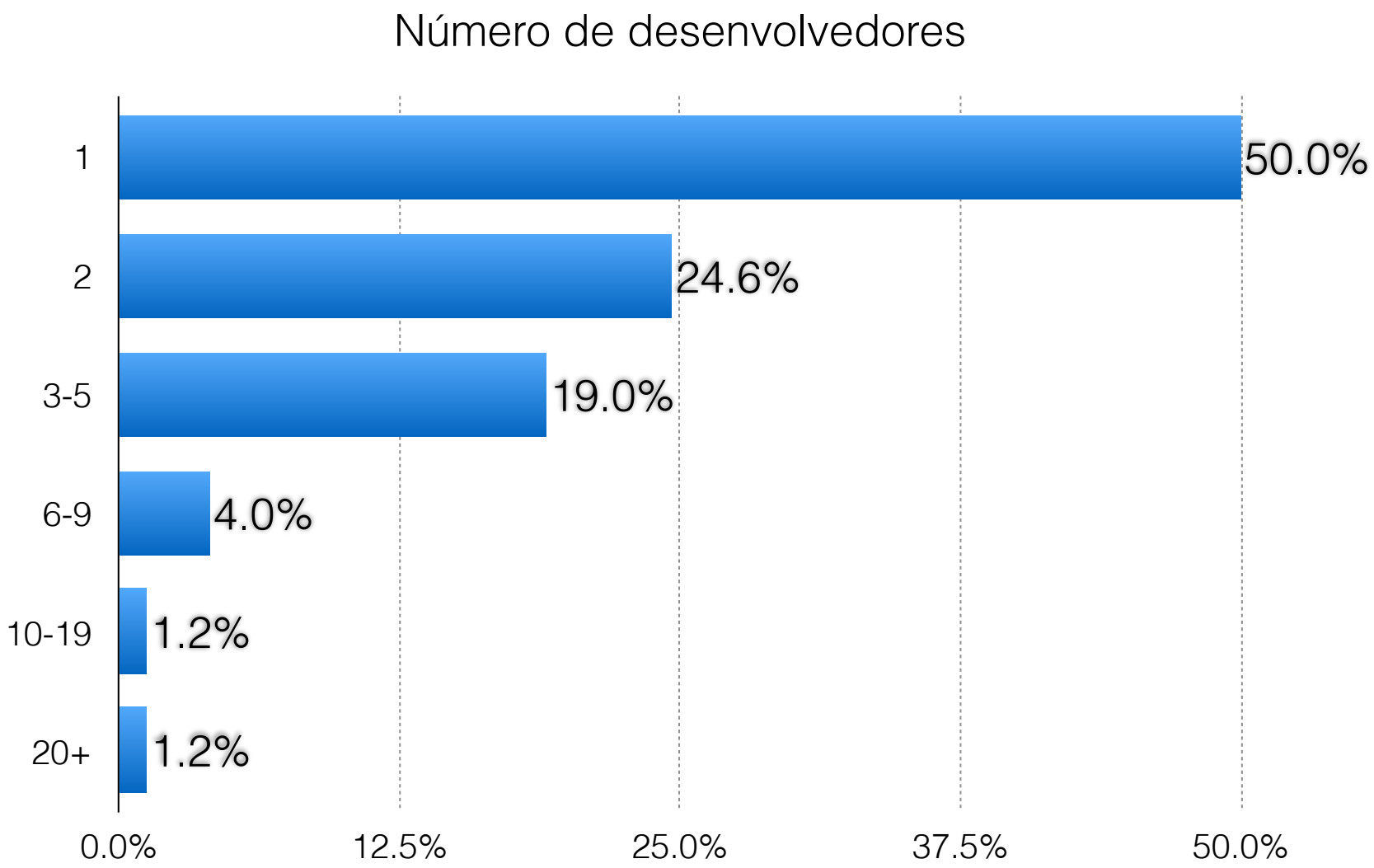
Porcentagem de renda por tipo de pagamento



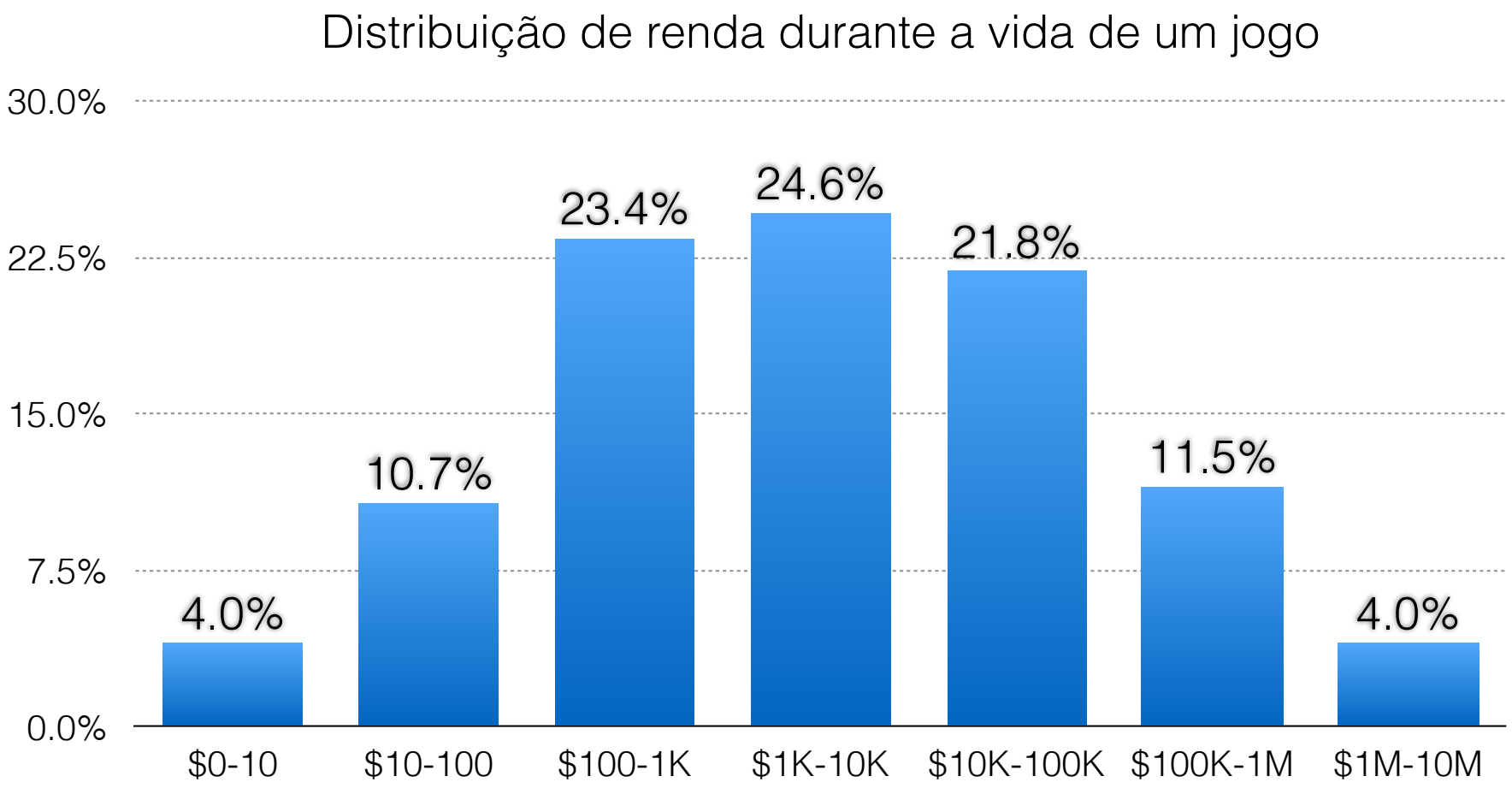
Desenvolvedores



Desenvolvedores



Desenvolvedores



Quem está aqui?

E como desenvolvo?

E como desenvolvo?

E como desenvolvo?

- Mac
  http://store.apple.com/br

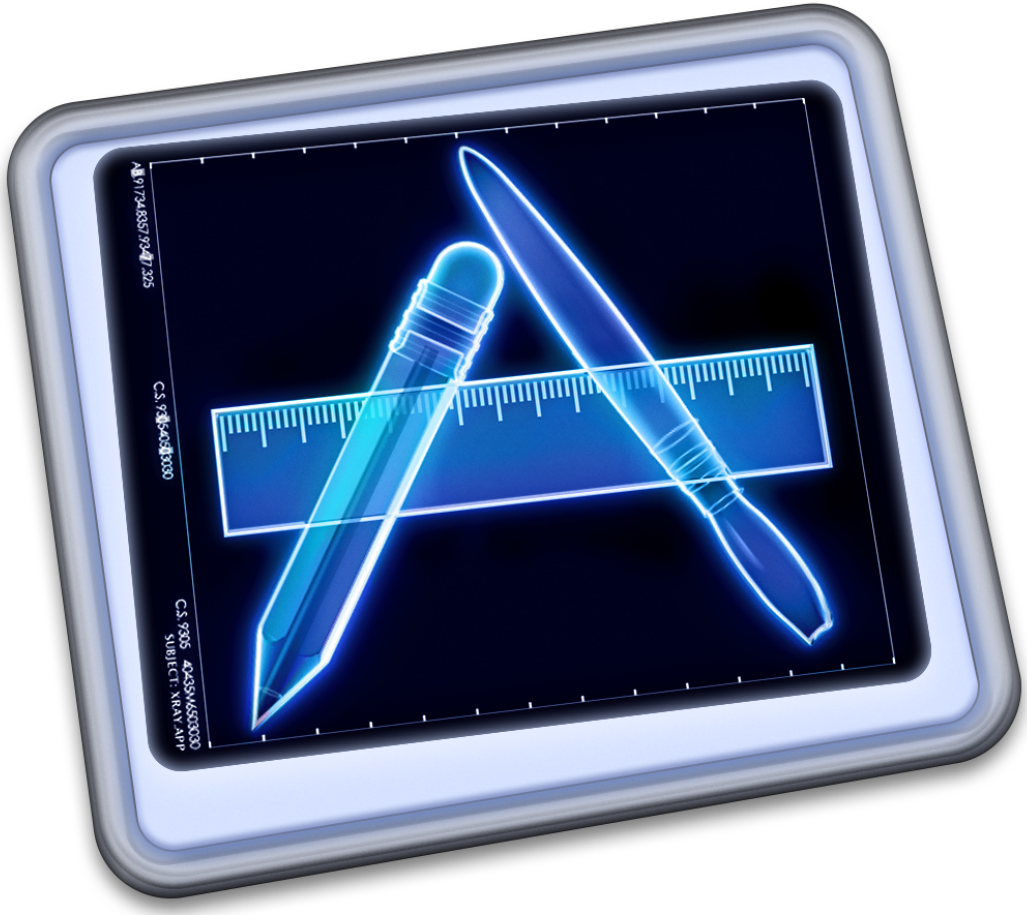
- Software Development Kit (SDK)
  http://developer.apple.com/ios

- iPhone
  http://www.tim.com.br
  http://www.vivo.com.br
  http://www.claro.com.br
  http://www.oi.com.br
  http://store.apple.com/br

Ferramentas

Xcode

- Programação
- Debug
- Análise estática
- Criação de Protótipos
- Layout de UI
- Conexão de eventos
- Localização
- Storyboards
- Documentação

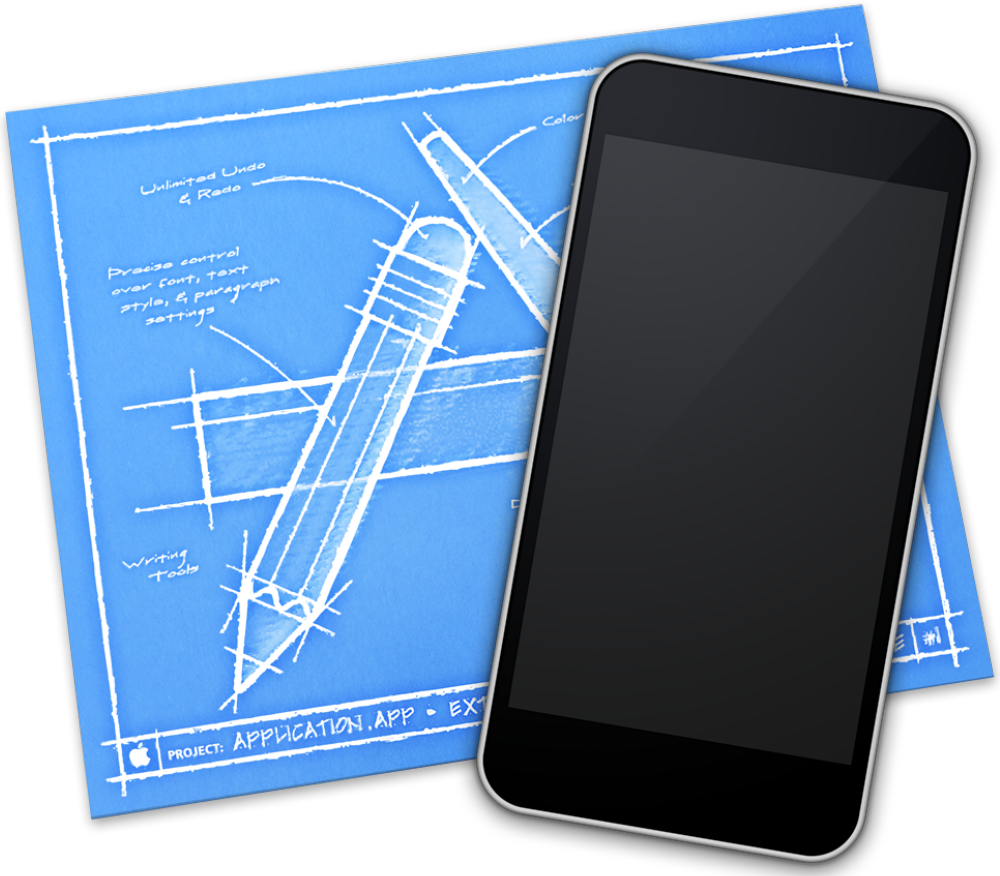


Debug

Análise de dados

Otimização

Instruments



Debug

iOS Simulator

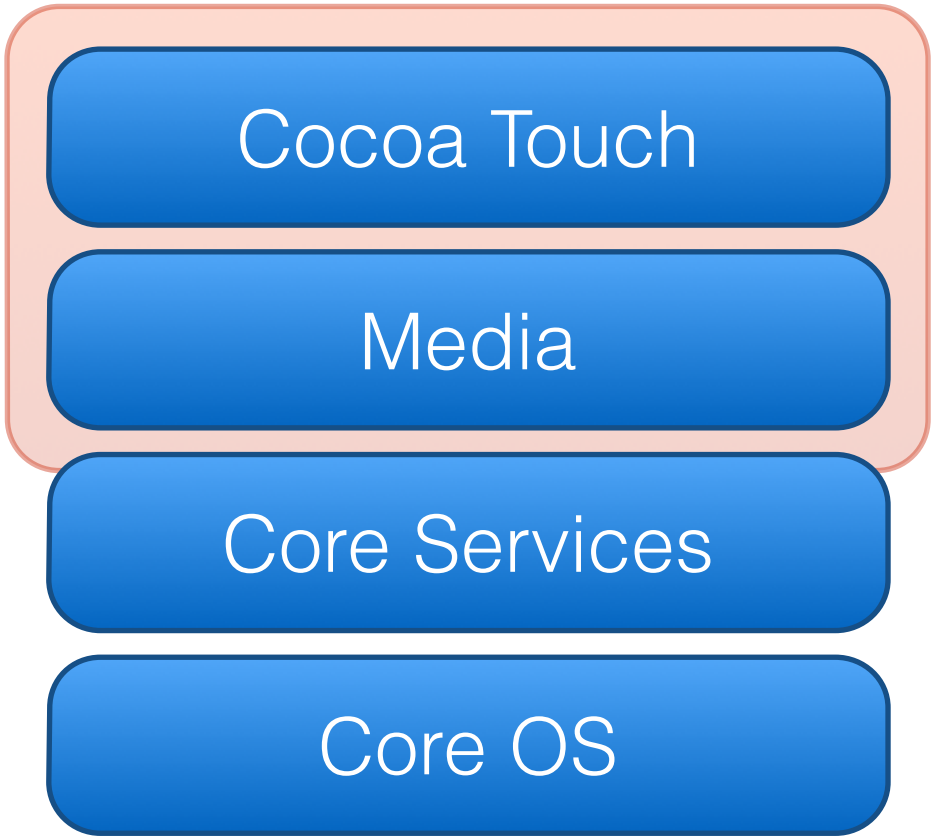
Linguagens Suportadas

C

C++

Objective-C

APIs



APIs

CocoaPods

http://cocoapods.org

APIs – Game Engines

Cocos2D

http://www.cocos2d-iphone.org

Unity

http://unity3d.com

Sparrow

http://gamua.com/sparrow

APIs – Game Engines

GLKit

SpriteKit

Beta Testing

TestFlight

http://testflightapp.com

Objective-C

Objective-C

- Criada na mesma época que C++
- Influenciada por C e Smalltalk
- Linguagem primária do NEXTSTEP da NeXT
- Linguagem primária da API Cocoa
- Superset de C
  - Sintaxe de objetos derivada do Smalltalk
  - Sintaxe não orientada a objetos igual a C

Objective-C

C++	C#	Objective-C
class	class	@interface
-	interface	@protocol
this	self	self
void*	Object	id
NULL	null	nil
true/false	true/false	YES/NO
Methods	Methods	Selectors
-	Partial	Categories
-	Object	NSObject
#include	using	#import

Objective-C

@interface:

Declaração de classes: superclasse, protocolos implementados, variáveis de instância, propriedades e seletores.

Também usado para declarar Categorias.

```
@interface Classe : Superclasse
<Protocolo> {
  [variáveis de instância]
}

[propriedades]
[seletores]

@end
```

```
@interface Classe (Categoria)

[seletores]

@end
```

Objective-C

```
@interface Pessoa : NSObject <Programador> {
  NSString *nome;
  float altura;
  float peso;
}

@property (strong) NSString *nome;
@property float altura;
@property (nonatomic) float peso;

- (void)dizNomeComVolume:(float)volume;

@end
```

Objective-C

```
@interface Pessoa(Higiene)

- (void)tomaBanho;
- (void)escovaDentes;

@end
```

Objective-C

@protocol:

Definição de protocolos de comunicação entre classes via seletores.

```
@protocol Protocolo

[seletores]

@end
```

Objective-C

```
@protocol Programador

- (void)programaProjeto:(NSString*)projeto
            paraCadeira:(NSString*)cadeira;

@end
```

Objective-C

@implementation:

Implementação de classes: declaração de seletores e sintetização de propriedades

```
@implementation Classe

[sintetizadores de propriedades]
[seletores]

@end
```

```
@implementation Classe(Categoria)

[seletores]

@end
```

Objective-C

```
@implementation Pessoa

//@synthesize nome;
//@synthesize altura;
//@synthesize peso;
- (float)peso {
  return peso;
}

- (void)dizNomeComVolume:(float)volume {
  // código para dizer o peso
}
- (void)programaProjeto:(NSString*)projeto
            paraCadeira:(NSString*)cadeira {
  // código para programar o projeto
}

@end
```

Objective-C

```
@implementation Pessoa(Higiene)

- (void)tomaBanho {
  // código para tomar banho
}
- (void)escovaDentes {
  // código para escovar os dentes
}

@end
```

Objective-C

C++	Objective-C
`int Classe::metodo(float param) {`	`- (int)metodo:(float)param {`

`objeto.metodo(5.0f);`
`// ou`
`objeto->metodo(5.0f);`
	`[objeto metodo:5.0f];`

`Classe objeto;`
`// ou`
`Classe* objeto = new Classe();`
	`Classe* objeto = [[Classe alloc] init];`

`Classe::Classe() : Superclasse() {`
`  /* init */`
`}`
	`- (id)init {`
	`  self = [super init];`
	`  if(self) {`
	`  /* init */`
	`  }`
	`  return self;`
	`}`

Objective-C

- NSObject
- NSString
- NSArray / NSMutableArray
- NSDictionary / NSMutableDictionary
- NSNumber
- NSValue
- NSData / NSMutableData
- NSDate

Objective-C

```
NSString* object = [[NSString alloc] initWithString:@"Oi, oi!”];
NSString* alias = [object retain];
// faz algo
[alias release];

[object autorelease];

NSString* autoreleased = [NSString stringWithString:@“alou?"];

NSString* autoreleasedTambem = @"Oi, mundo!";
```

Objective-C ARC

```
NSString* object = [[NSString alloc] initWithString:@"Oi, oi!”];
NSString* alias = object

NSString* autoreleased = [NSString stringWithString:@“alou?"];

NSString* autoreleasedTambem = @"Oi, mundo!";
```

Objective-C ARC

```
NSArray* numeros = @[1, 2, 3, 4];
NSDictionary* dicionario = @{
    @"plataforma" : @"iOS",
    @"dispositivo" : @"iPhone"
};
```

OpenGL ES

OpenGL ES

- OpenGL for Embedded Systems
- Subset de OpenGL
- Sintaxe exatamente igual a OpenGL
- Sem modo imediato
  - glBegin(GLenum) e glEnd(GLenum) removidas
  - Priorizando glDrawArrays e glDrawElements
- Suporta GLSL a partir de OpenGL ES 2.0

Audio

Audio

- OpenAL
- CoreAudio
- AVAudioPlayer
- 32 canais descomprimidos
- 1 canal comprimido e decodificado em hardware

OpenAL

- Igual em todas as plataformas
- Audio avançado:
  - 32 canais
  - Posicionamento de ouvinte/fontes em 3D
  - Baixa latência
- Estrutura de API similar a OpenGL
- Gerência de buffers, bitrates, formatos e sources pelo programador

CoreAudio

- OpenAL é montado em cima dele no iOS
- Todos os benefícios do OpenAL exceto portabilidade
- API proprietário ao OS X / iOS
- Gerência de buffers, bitrates, formatos e sources pelo programador

AVAudioPlayer

- Extremamente simples
- Alta latência
- Melhor para músicas de fundo
- Gerência de qualquer tipo de formato e buffer feito pelo framework

Input

- Multitouch
  - Até cinco dedos simultaneamente (10 no iPad)
  - Eventos como touchesBegan, touchesMoved, touchesEnded, touchesCancelled
- Acelerômetro
  - Por meio de polling
  - Programador define o intervalo
- Voz
- Giroscópio

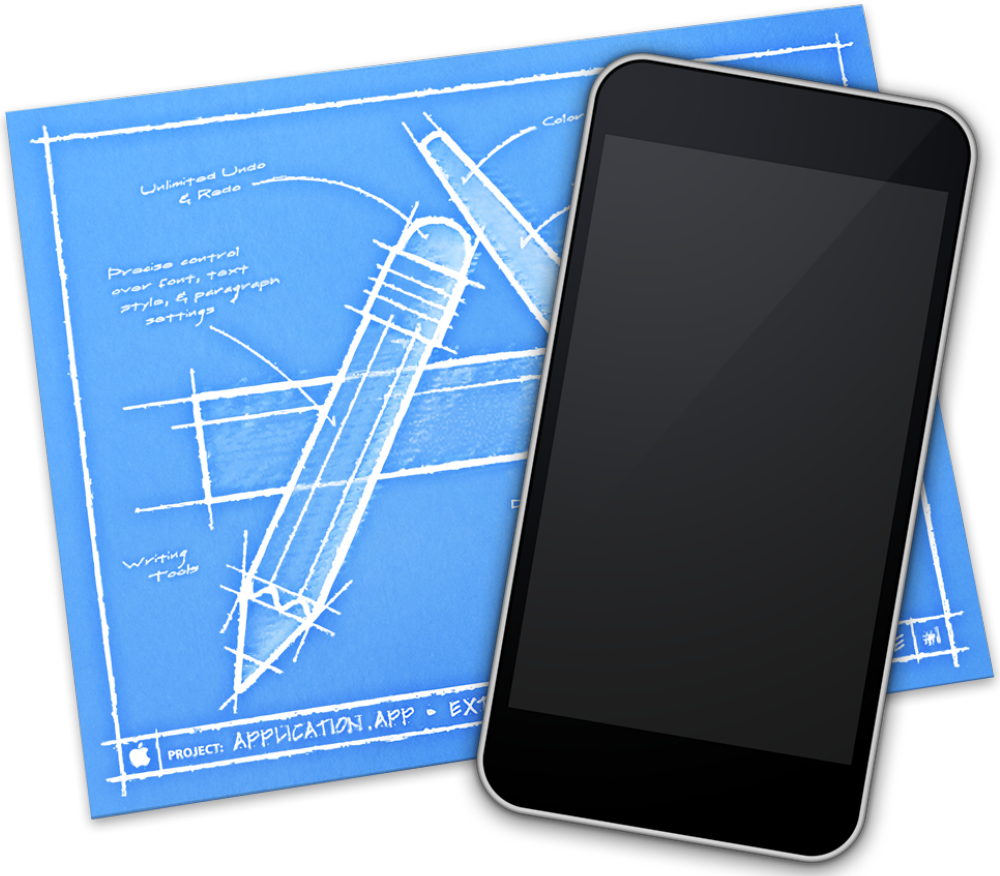
Multiplayer

- Bluetooth P2P
  - Conecta dois dispositivos via Bluetooth
  - Não compatível com aparelhos anteriores ao iPhone 3G
  - Interface própria
- IP
  - Edge, 3G, LTE, WiFi
  - Berkeley sockets
- Game Center

Precisando de ajuda?

- Documentação do Xcode
- Arquivos .h das classes
- Documentação, exemplos, observações técnicas, e vídeos do WWDC no Apple Developer Connection: http://developer.apple.com/

Como testar?



Não precisa assinar binário

Não precisa conta de desenvolvedor

iOS Simulator

Todo binário precisa ser assinado

Precisa conta de desenvolvedor

Aparelho iOS

iPhone é fechado

- Distribuição amarrada por Certificados
- Aplicativos identificados por um AppID
- Só aplicativos assinados são executados
- Instalação feita por IPAs
- Sem certificado? Sem testar no aparelho!

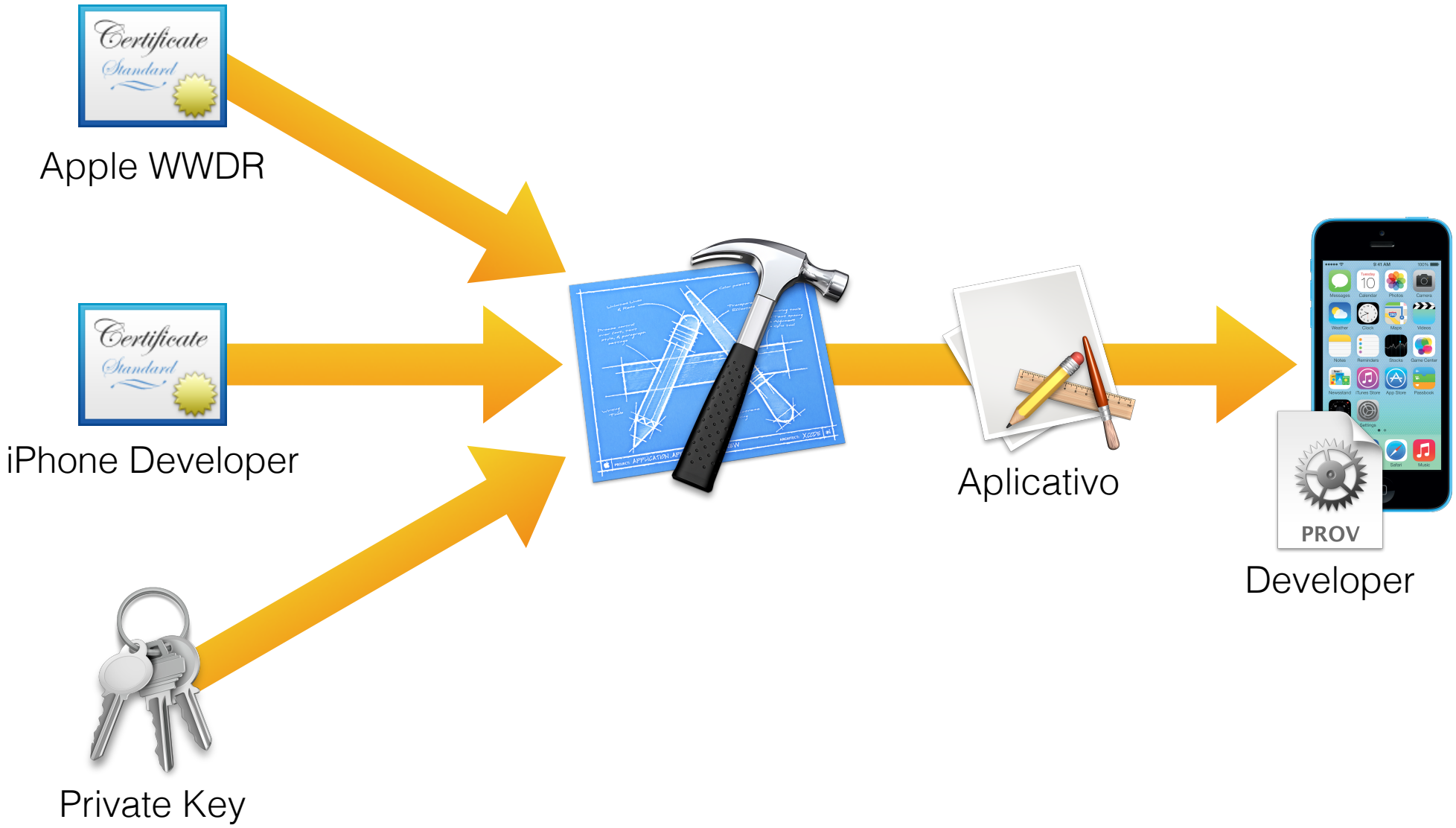
Certificados

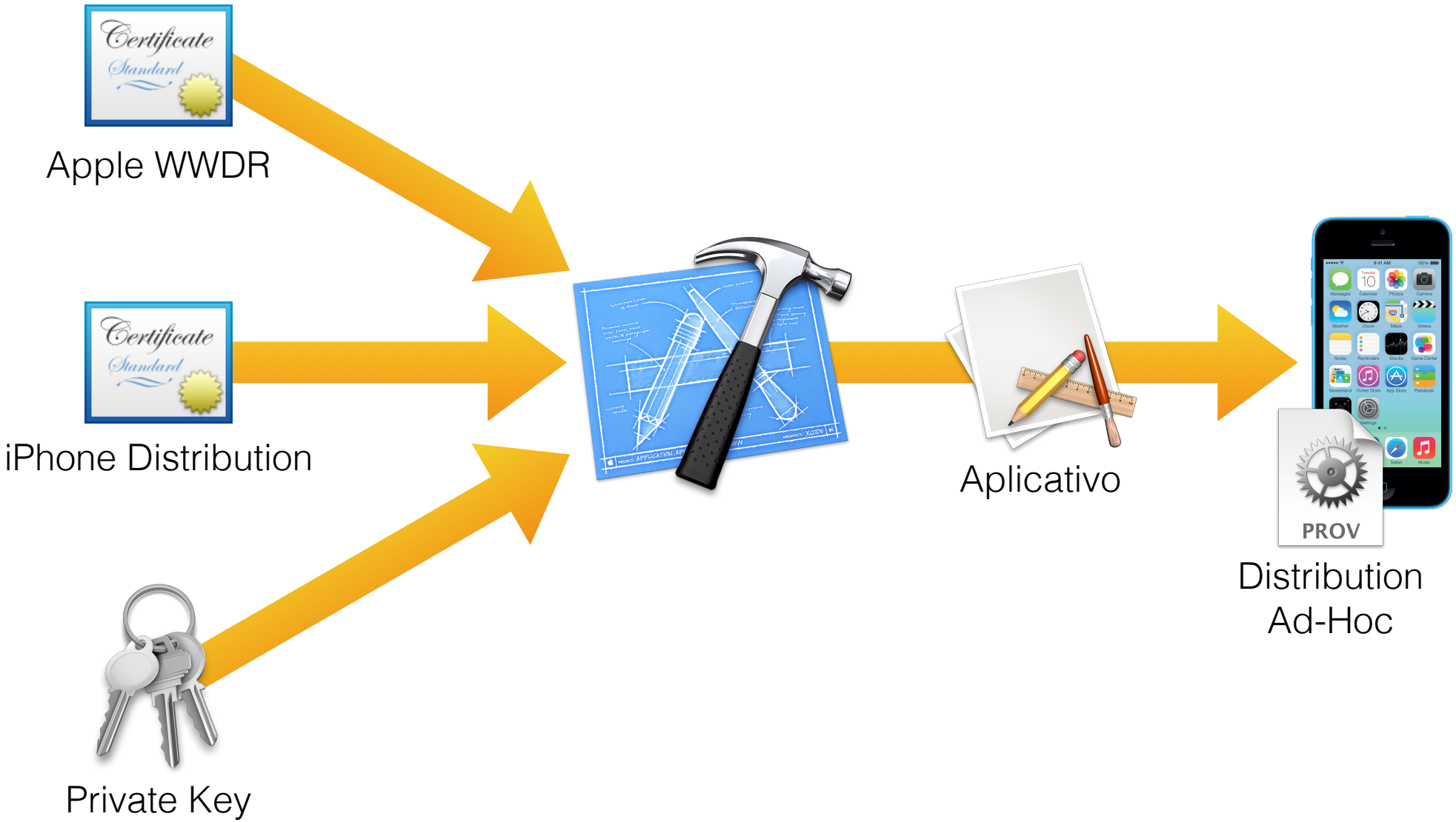
- Apple WWDR
- iPhone Developer
  - Sua permissão para desenvolver
- iPhone Distribution
  - Sua permissão para distribuir

Perfis de Provisionamento

- Desenvolvimento
  - Presente em todo build
  - Específico para alguns dispositivos
- Distribuição
  - Ad Hoc
  - App Store


Aplicativo


Certificate
Standard
Apple WWDR
Certificate
Standard
iPhone Developer
Private Key
Aplicativo
PROV
Developer


Certificate
Standard
Apple WWDR
Certificate
Standard
iPhone Distribution
Private Key
Aplicativo
PROV
Distribution
Ad-Hoc

Como publicar?

Como publicar?

1. Criar uma conta no iPhone Dev Center
   http://developer.apple.com/ios
   Identificado por um Apple ID

2. Escolher um plano:

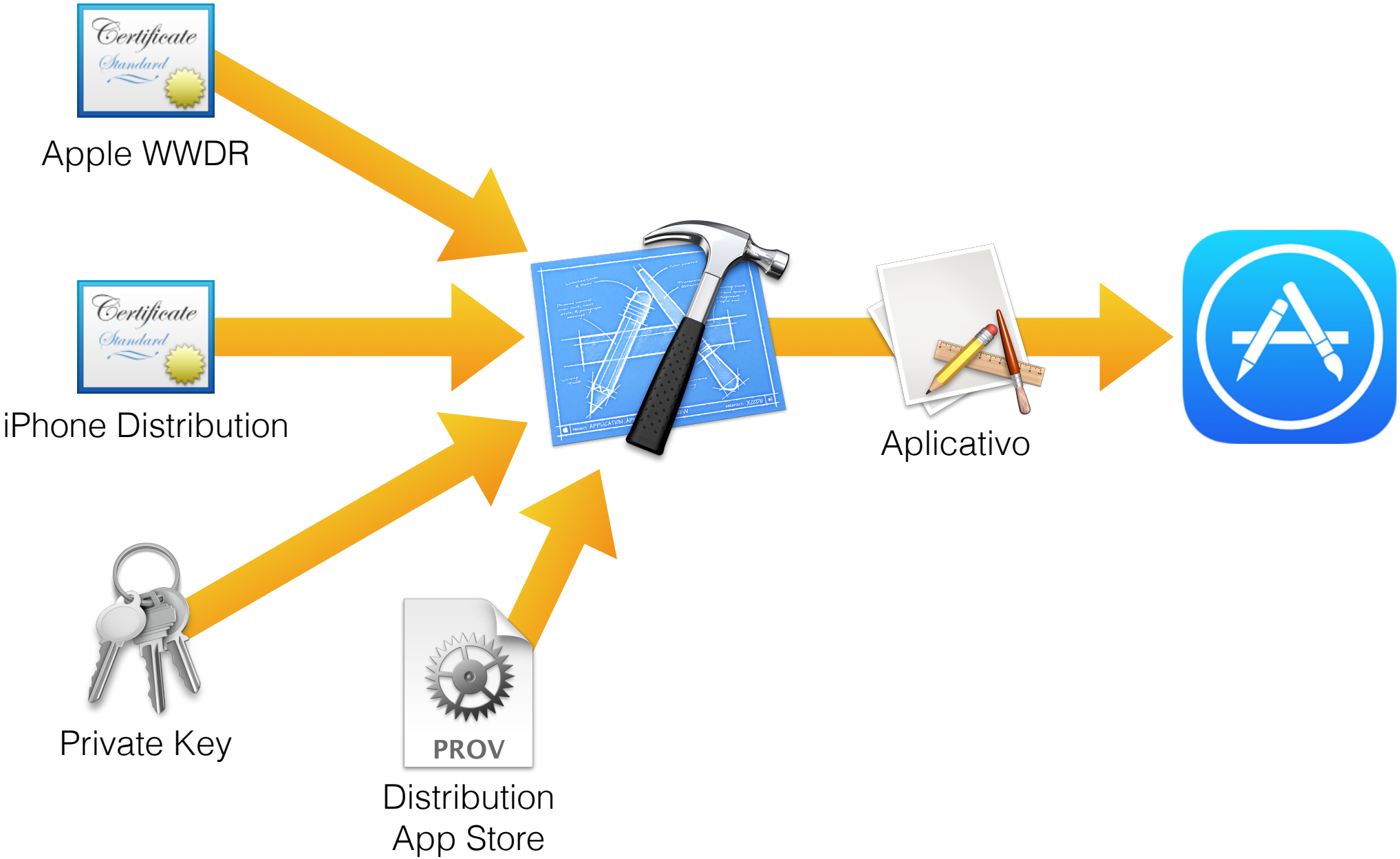
   - Individual (PF ou PJ) – US$99 ao ano
   - Enterprise – US$299 ao ano

3. Preencher a papelada e enviar à Apple

4. Esperar…

Envio à Apple

- http://itunesconnect.apple.com/
  - Applications
  - In-App Purchases
  - Relatórios de vendas
  - Relatórios financeiros
  - Contratos


Apple WWDR
iPhone Distribution
Private Key
PROV
Distribution
App Store
Aplicativo

NÃO PERCA SUA PRIVATE KEY

Envio à Apple

- Cada aplicativo contém:
  - Ícone
  - 1 a 5 screenshots por idioma por dispositivo
  - Descrição em um ou mais idiomas
  - Avaliação etária (self-service)
  - Preço
  - Escolha de lojas
  - Binário enviado à Apple através do Xcode

Envio à Apple

- Processo de aprovação é caixa preta parcialmente explicado para desenvolvedores pelas regras em: https://developer.apple.com/appstore/guidelines.html
- Comunicação entre Apple-Desenvolvedor sob NDA
- Espera em torno de 5 dias (2013-12-05) nos casos simples
- Quando aprovado, entra na loja em data especificada ou no dia seguinte

Fiz um jogo sensacional!

E agora?

Marketing

- Seu produto deve ser encontrado pelo seu público
- Seu produto deve demonstrar seu valor para este público
- Um relacionamento forte deve ser criado com o consumidor
- Esses aspectos devem ser considerados desde o período de pré-produção

Marketing

- Nome, preço, ícone, descrição e até a ordem das fotos na página do jogo são importantes

Onde está Wally?

Marketing

- Na App Store, ranking é (quase) tudo
- Sua colocação depende do número de unidades vendidas nos últimos dias
- Milhares de vendas por dia para se conseguir entrar no top 100 da categoria de Action Games nos EUA (uma das mais competitivas)

Marketing

- Aprenda a lidar com os rankings
- Manipule bem o preço do seu jogo
- Uma versão “lite” com bom ranking pode aumentar a exposição da versão paga
- A versão lite deve ser bem preparada para garantir uma boa conversão
- Manipule bem a categoria do seu jogo
- Updates!

Marketing

- Website (com suporte a usuários e atendimento à mídia eficientes)
- Blog
- Twitter
- Trailer
- Presença ativa em comunidades (Touch Arcade é uma boa)

Marketing

- Crie “hype” para o jogo antes do lançamento com trailers e “exclusivas” para sites de grande visitação
- Mantenha uma boa relação com os administradores desses sites
- Exposição em sites de reviews não costuma aumentar muito as vendas, mas pode ajudar a Apple a te notar para um “feature”

Marketing

- Funcionalidades sociais no jogo
- Crie uma marca
- Parcerias com outros desenvolvedores

Perguntas?

lmmenge+cin@gmail.com

@lmmenge