

# Aprendendo a programar para iOS



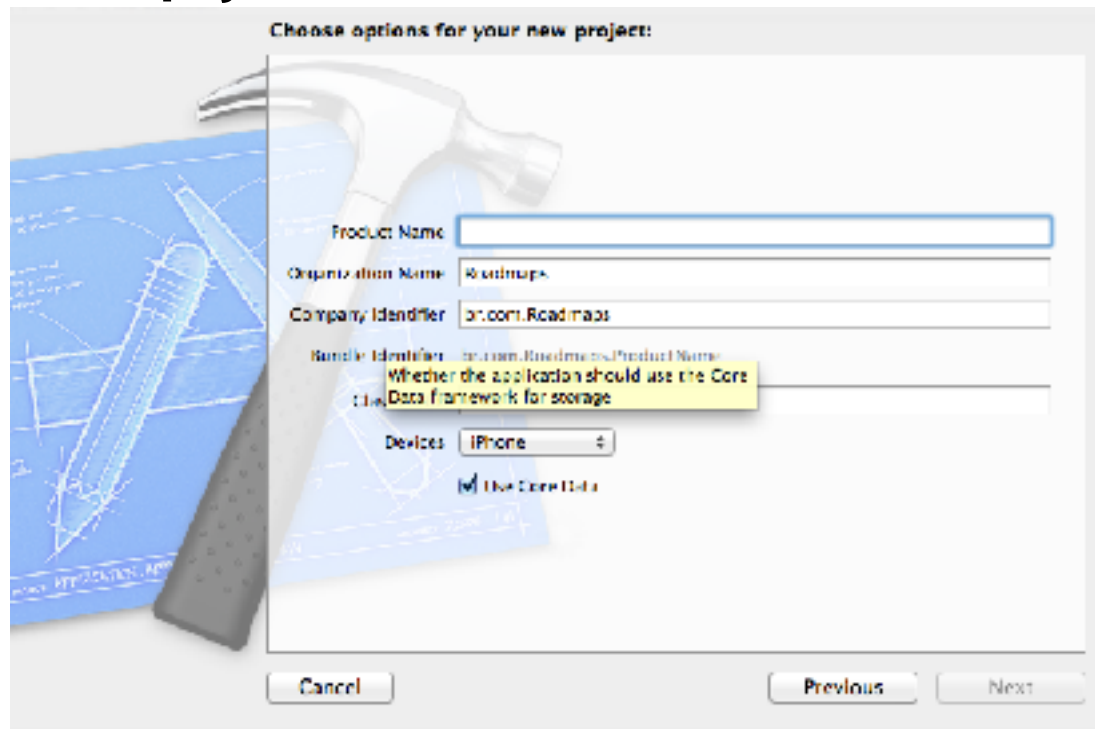
**CORE DATA**

# Usando CoreData

- O CoreData é o banco de dados utilizado em iOS.
  - Não é relacional
  - Pode utilizar SQLite para fazer a persistência dos dados.

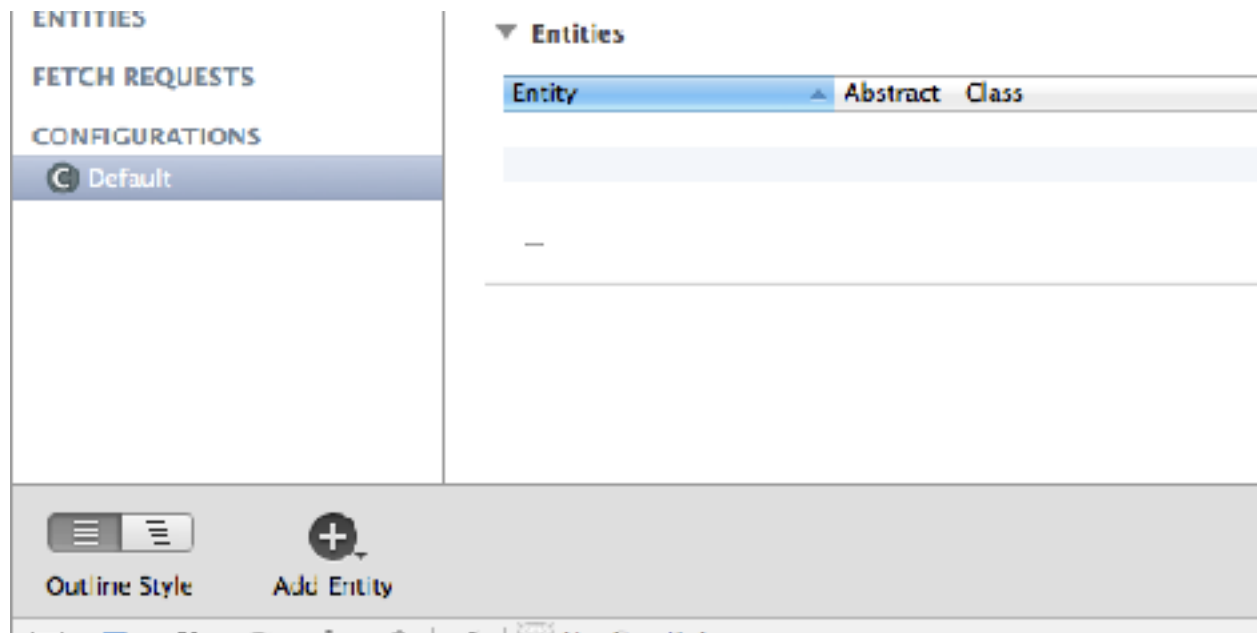
# Core Data

- Criando um Projeto do tipo Empty Application
- Marcar a opção “Use Core Data”



# Core Data

- Vá ao Arquivo NewCoreData.xcdatamodeld
- Crie uma entidade chamada Produtos clicando no botão “add Entity”



# Core Data

- Dê o nome “Produto” ao seu objeto.
- E insira os seguintes atributos.
  - Modelo, Marca, Preço, disponibilidade
  - Adicione um relacionamento Fotos

## ENTITIES

**E** Produto

## FETCH REQUESTS

## CONFIGURATIONS

**C** Default

### ▼ Attributes

Attribute ▲

Type

**U** Nome

Undefined



+ -

### ▼ Relationships

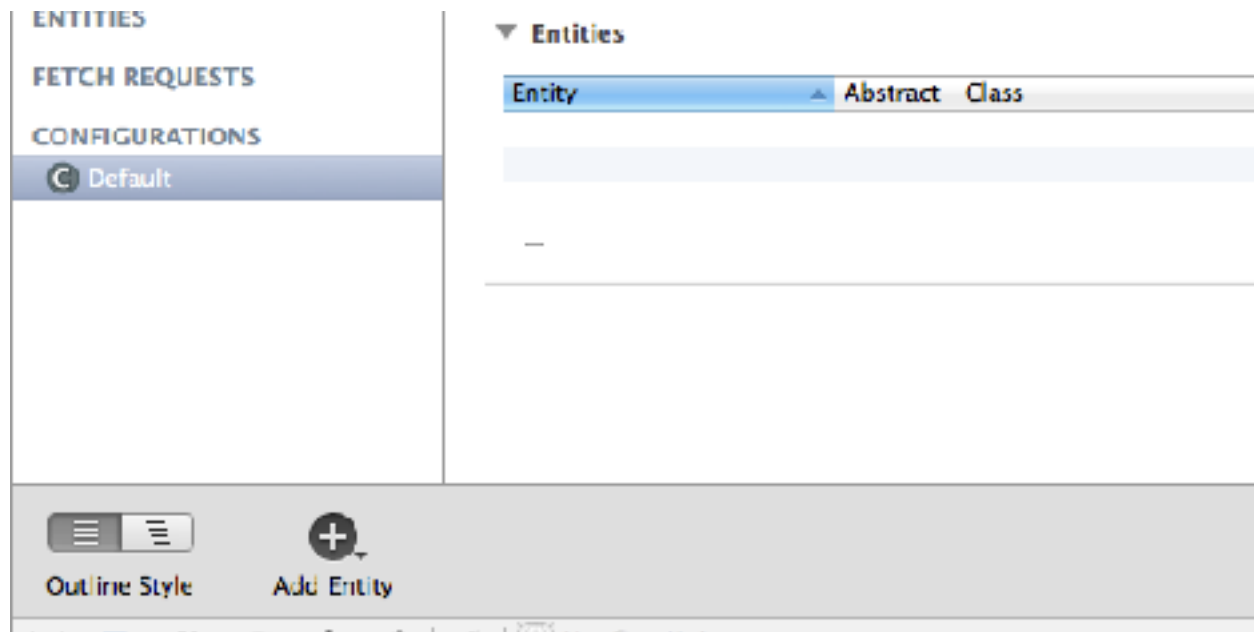
Relationship ▲

Destination

Inverse

# Core Data

- Crie uma nova entidade chamada Fotos clicando no botão “add Entity”



# Core Data

- Dê o nome “Foto” ao seu objeto.
- E insira os seguintes atributos.
  - NSData, nome, extensao
  - Adicione um relacionamento Produtos.

## ENTITIES

**E** Produto

## FETCH REQUESTS

## CONFIGURATIONS

**C** Default

### ▼ Attributes

Attribute ▲

Type

**U**

Nome

Undefined



+ -

### ▼ Relationships

Relationship ▲

Destination




Inverse




# Core Data

- Configurando a Cardinalidade
  - Vá no relacionameto desejado e na guia “Data Model Inspector” altere os valores de “Type”
  - To one, To many

## ▼ Attributes

Attribute ▼	Type	
 nome	String	↕
 extensao	String	↕
 data	Binary Data	↕
+ -		

## ▼ Relationships

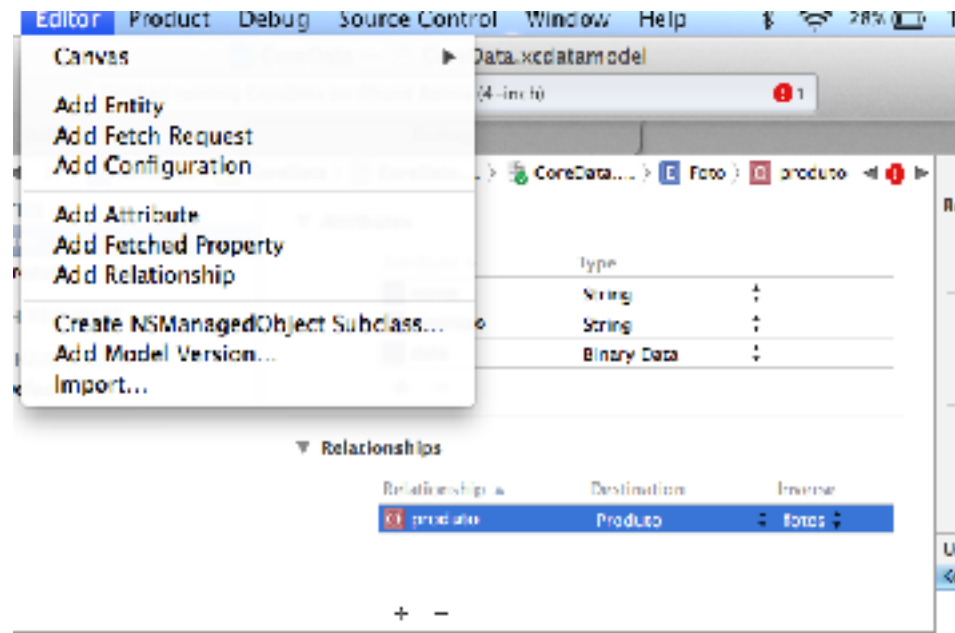
Relationship ▲	Destination	Inverse
 produto	Produto	↕ fotos ↕
+ -		

## Relationship

Name	produto
Properties	<input type="checkbox"/> Transient <input checked="" type="checkbox"/> Optional
Destination	Produto
Inverse	fotos
Delete Rule	Nullify
Type	To One
Advanced	<input type="checkbox"/> Index in Spotlight <input type="checkbox"/> Store in External Record File
User Info	
Key	Value

# Core Data

- Criando as classes
  - Ainda no arquivo NewCoreData.xcdatamodeld selecione a classe desejada.
  - Vá no menu Superior “Editor”
  - Clique em “Create NSManagedObject Subclass



# Core Data

- São gerados alguns arquivos desta operação:
  - Produto+CoreDataClass (.h e .m).
    - Arquivo com a definição da classe. Todos os métodos utilitários que criamos devem estar nestes arquivos.
    - Geralmente criamos métodos para criar, buscar e remover entidades.
  - Produto+CoreDataProperties (.h e .m).
    - Arquivo com os atributos do core data, esse arquivo é gerado automaticamente pelo xcode.

# Core Data

- Métodos sugeridos

- (void) fillWithAttributes: (NSDictionary \*) attributes {  
    id t;

```
    t = attributes[@"nome"];  
    if (t != nil && ![t isKindOfClass:[NSNull class]])  
        self.nome = [t description];
```

```
    t = attributes[@"marca"];  
    if (t != nil && ![t isKindOfClass:[NSNull class]])  
        self.marca = [t description];
```

```
    t = attributes[@"quantidade"];  
    if (t != nil && ![t isKindOfClass:[NSNull class]])  
        self.quantidade = @( [t integerValue] );
```

```
}
```

# Core Data

- Métodos sugeridos

```
+(Produto *)produtoWithDictionary:(NSDictionary *)dic{
    AppDelegate *app_d = (AppDelegate
*)UIApplication.sharedApplication.delegate;
    Produto *p = [NSEntityDescription
                  insertNewObjectForEntityForName:@"Produto"
inManagedObjectContext:app_d.persistentContainer.viewContext];

    [p fillWithAttributes:dic];

    NSError *saveError = nil;
    [app_d.persistentContainer.viewContext save:&saveError];
    return p;
}
```

# Core Data

- Métodos sugeridos

```
+(NSArray *) produtos {
    AppDelegate *app_d = (AppDelegate *)UIApplication.sharedApplication.delegate;
    NSFetchRequest *req = [NSFetchRequest fetchRequestWithEntityName:@"Produto"];
    NSError *error;
    NSArray *ps = [app_d.persistentContainer.viewContext executeFetchRequest:req
error:&error];
    return ps;
}

+(Produtos *) produtoComNome:(NSString *)nome {

    AppDelegate *app_d = (AppDelegate *)UIApplication.sharedApplication.delegate;

    NSPredicate *pred = [NSPredicate predicateWithFormat:@"nome == %@", nome];
    NSFetchRequest *req = [NSFetchRequest fetchRequestWithEntityName:@"Produtos"];
    [req setPredicate:pred];
    NSError *error;

    NSArray *p = [app_d.persistentContainer.viewContext executeFetchRequest:req
error:&error];
    return [p firstObject];
}
```

# Core Data

- Métodos sugeridos

```
+(void) deleteProdutos:(Produtos *) prod {  
    AppDelegate *app_d = (AppDelegate *)UIApplication.sharedApplication.delegate;  
    [app_d.persistentContainer.viewContext deleteObject:prod];  
    NSError *error;  
    [app_d.persistentContainer.viewContext save:&error];  
}
```

# Core Data

- Observações Gerais
  - NSPredicate: A definition of logical conditions used to constrain a search either for a fetch or for in-memory filtering.
  - Para saber mais: Apple Predicate Programming Guide



# Exercício curto

- Incremente o exemplo da aula anterior adicionando a funcionalidade de deletar um item.
- Extra: adicionem um UISearchBar na table-view e façam a busca em tempo real utilizando predicate.
- [bit.ly/aula-ios-7](http://bit.ly/aula-ios-7)

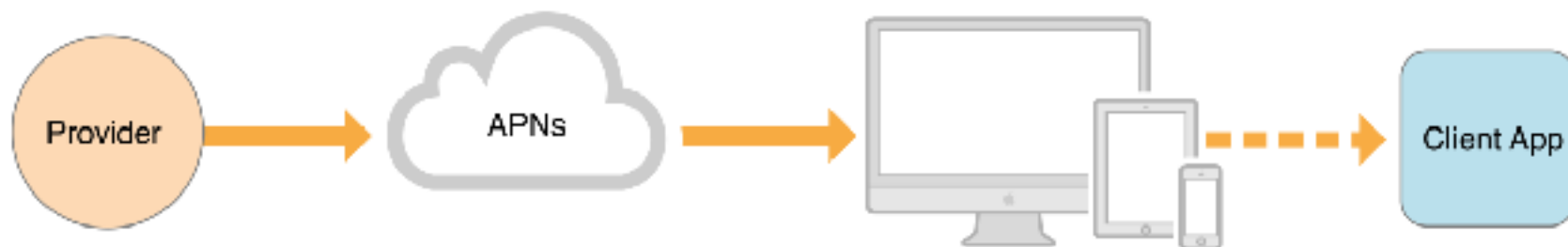
Apple Push Notification Service

**APNS**

# APNS

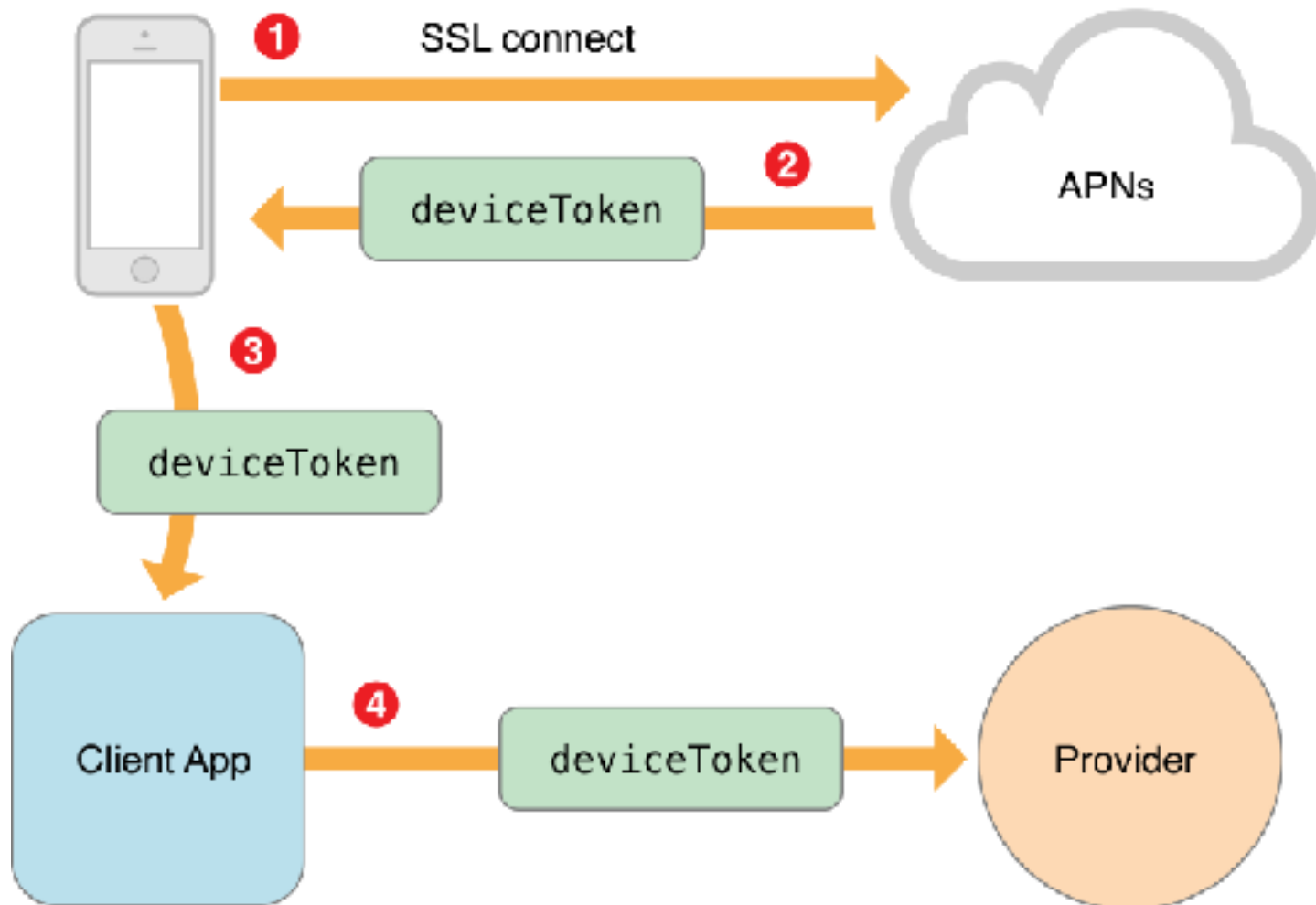
- Apple Push Notification Service
  - Permite o envio de mensagens instantâneas para dispositivos que o permitirem.
  - Mensagens possuem duas partes: Token e Payload.
  - O payload é um hash que especifica como notificar o aplicativo.
  - O payload está limitado a 4 Kbytes (5 pra notificações VOIP).

# APNS



- Notificações são enviadas por servidor para a apple.
- Apple manda para o dispositivo.
- Dispositivo manda para o aplicativo.

# APNS



# APNS - Payload

- Payload pode conter:
  - Som a ser tocado
  - Mensagem a ser exibida
  - Número para colocar como alerta no app
  - Botões da notificação
  - Informações customizadas

# Payload - Exemplo

```
{  
  "aps" : {  
    "alert" : "You got your emails.",  
    "badge" : 9,  
    "sound" : "bingbong.aiff"  
  },  
  "app" : "1",  
  "acme2" : 42  
}
```

- APS: Hash obrigatório, diz informações sobre alerta
- Outras entradas na raiz: informações que o app pode acessar.

# Payload - Exemplo

```
{  
  "aps" : {  
    "alert" : {  
      "title" : "Game Request",  
      "body" : "Bob wants to play poker",  
      "action-loc-key" : "PLAY"  
    },  
    "badge" : 5  
  },  
  "acme1" : "bar"  
}
```

- APS: Hash obrigatório, diz informações sobre alerta
- Outras entradas na raíz: informações que o app pode acessar.



# Payload

- Alert
  - mensagem ou hash com opções sobre o alerta, como título e descrição do alerta, chave de tradução da mensagem
- Badge
  - Número a ser mostrado no ícone do app
- Sound
  - Nome de um arquivo de som presente no app, que vai ser tocado quando a notificação chegar
- Content-available
  - Flag indicando se a notificação deve ser exibida, ou é apenas para algum processamento em background do app
- Category
  - Configuração para resposta da notificação

# Passo a Passo

- Configurar certificados e profiles
- Obter token no app
- Enviar token para servidor
- Enviar notificação, do servidor, utilizando certificado gerado e token enviado

# Configurando Certificados

- Criar um novo App ID (developer.apple.com)
  - Criar sufixo App ID explícito
    - br.com.roadmaps.exemplo-push
  - Marcar suporte a Push Notifications
- Gerar Provisioning Profile com esse App ID
- Gerar Certificado de PUSH com esse App ID

---

## App ID Suffix

### ● **Explicit App ID**

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (\*).

### ● **Wildcard App ID**

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (\*) as the last digit in the Bundle ID field.

Bundle ID:

Example: com.domainname.\*

---

## App Services

Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

- Enable Services:
- ☐ **App Groups**
  - ☐ **Associated Domains**
  - ☐ **Data Protection**
    - ☐ Complete Protection
    - ☐ Protected Unless Open
    - ☐ Protected Until First User Authentication
  - ☒ **Game Center**
  - ☐ **HealthKit**
  - ☐ **HomeKit**
  - ☐ **Wireless Accessory Configuration**
  - ☐ **Apple Pay**
  - ☐ **iCloud**
    - ☐ Compatible with Xcode 5
    - ☐ Include CloudKit support (requires Xcode 6)
  - ☒ **In-App Purchase**
  - ☐ **Inter-App Audio**
  - ☐ **Wallet**
  - ☒ **Push Notifications**
  - ☐ **VPN Configuration & Control**

Select Type

Configure

Generate

Download



## Select App ID.

If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. If you want to create one provisioning profile for multiple apps or don't need a specific Bundle ID, select a wildcard App ID. Wildcard App IDs use an asterisk (\*) as the last digit in the Bundle ID field. Please note that iOS App IDs and Mac App IDs cannot be used interchangeably.

App ID:

Select Type

Request

Generate

Download



## What type of certificate do you need?

### Development

- ☐ **iOS App Development**  
Sign development versions of your iOS app.
- ☒ **Apple Push Notification service SSL (Sandbox)**  
Establish connectivity between your notification server and the Apple Push Notification service sandbox environment. A separate certificate is required for each app you develop.

### Production

Select Type

Request

Generate

Download



## Which App ID would you like to use?

All App IDs you enable to receive push notifications require its own individual Push SSL Certificate. The App ID-specific Push SSL certificate allows your notification server to connect to the Apple Push Notification Service. Note that only explicit App IDs with a specific Bundle Identifier can be used to create an Push SSL Certificate.

### Select an App ID for your Push SSL Certificate (Sandbox)

App ID: 8BRMRY3GUS.br.com.roadmaps.exemplo-push





# Configurando App

- No App Delegate:

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    UIUserNotificationType types = UIUserNotificationTypeBadge |
                                   UIUserNotificationTypeSound |
                                   UIUserNotificationTypeAlert;
    UIUserNotificationSettings *mySettings = [UIUserNotificationSettings
settingsForTypes:types categories:nil];
    [[UIApplication sharedApplication]
registerUserNotificationSettings:mySettings];

    return YES;
}
```

# Configurando App

- No App Delegate (token recebido):

```
- (void)application:(UIApplication *)app
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)devToken {
    NSLog(@"Got device token: %@", [devToken description]);

    self.push_token = [devToken hexadecimalString];
    [self tryToSendPushToken];
}
```

```
- (void)application:(UIApplication *)app
didRegisterUserNotificationSettings:(UIUserNotificationSettings
*)notificationSettings{
    [app registerForRemoteNotifications];
}
```

# Configurando App

- No App Delegate (notificação recebida):

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo {
    // id aps = userInfo[@"aps"];
    NSLog(@"Received remote notification");
}
```

# Configurando Servidor

- Colocar certificado no servidor
- Configurar webservice/página para receber e armazenar tokens
- Fazer envio utilizando biblioteca de envio de Push, o certificado e o token do usuário que deseja enviar

# Pushmeup - Rails

```
APNS.host = 'gateway.push.apple.com'  
# gateway.sandbox.push.apple.com is default and only for development  
# gateway.push.apple.com is only for production  
  
APNS.pem = '/path/to/pem/file' # this is the file you just created  
APNS.pass = '' # Just in case your pem need a password  
  
device_token = '123abc456def'  
APNS.send_notification(device_token, 'Hello iPhone!' )  
APNS.send_notification(device_token, :alert => 'Hello iPhone!', :badge =>  
1, :sound => 'default')
```

Comunicação de rede

# **MULTIPART REQUEST**

# UIImagePickerController

An object that manages customizable, system-supplied user interfaces for taking pictures and movies on supported devices, and for choosing saved images and movies for use in your app.

# Utilizando UIImagePickerController

- Gerencia interações do usuário e entrega o resultado destas interações para um “delegate”.
- O atributo “sourceType” diz de onde vem a imagem:

`UIImagePickerControllerSourceTypeCamera`

`UIImagePickerControllerSourceTypePhotoLibrary`

`UIImagePickerControllerSourceTypeSavedPhotosAlbum`



# Utilizando UIImagePickerController

- Para que o sistema permita que nós utilizemos a câmera ou as imagens do dispositivo, devemos configurar o aviso de permissão para isto, no arquivo info.plist.
  - Privacy - Photo Library Usage Description
  - Privacy - Microphone Usage Description
  - Privacy - Camera Usage Description

# Utilizando UIImagePickerController

1. Verificar se o dispositivo pode escolher fotos da fonte escolhida, chamando o método:  
`[UIImagePickerController isSourceTypeAvailable:TIP0];`
2. Verifique quais tipos de mídia estão disponíveis para a fonte de imagens que você vai usar chamando o método:  
`[UIImagePickerController  
availableMediaTypesForSourceType:TIP0]`
3. Configure a interface pra exibir a interface pra os tipos de mídia que você quiser atribuindo um valor para o atributo “mediaTypes”, que é um array.
4. Apresente a interface, não esquecendo de configurar o delegate e os métodos para tratar o retorno.

# Utilizando UIImagePickerController

- Apresentando interface:

```
UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];  
[self presentViewController: cameraUI animated: YES completion:nil];
```

## Métodos do delegate pra implementar:

```
(void)imagePickerControllerDidCancel:(UIImagePickerController *)picker;  
– (void)imagePickerController:(UIImagePickerController *)picker  
didFinishPickingMediaWithInfo:(NSDictionary<NSString *,id> *)info;
```

# Utilizando UIImagePickerController

```
- (BOOL) startCameraControllerFromViewController: (UIViewController*) controller
                                usingDelegate: (id <UIImagePickerControllerDelegate,
                                UINavigationControllerDelegate>)delegate {

    if (([UIImagePickerController isSourceTypeAvailable:
        UIImagePickerControllerSourceTypePhotoLibrary] == NO)
        || (delegate == nil)
        || (controller == nil))
        return NO;

    UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];
    cameraUI.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

    cameraUI.mediaTypes =
    [UIImagePickerController availableMediaTypesForSourceType:
        UIImagePickerControllerSourceTypePhotoLibrary];
    cameraUI.allowsEditing = NO;

    cameraUI.delegate = delegate;
    [controller presentViewController: cameraUI animated: YES completion:nil];
    return YES;
}
```

# Utilizando UIImagePickerController

```
- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary<NSString *,id> *)info {
    NSString *mediaType = [info objectForKey: UIImagePickerControllerMediaType];
    UIImage *originalImage, *editedImage, *imageToSave;

    // Handle a still image capture
    if ([mediaType isEqualToString:@"public.image"]) {
        editedImage = (UIImage *) [info objectForKey:
                                   UIImagePickerControllerEditedImage];
        originalImage = (UIImage *) [info objectForKey:
                                     UIImagePickerControllerOriginalImage];

        if (editedImage) {
            imageToSave = editedImage;
        } else {
            imageToSave = originalImage;
        }

        UIImageWriteToSavedPhotosAlbum (imageToSave, nil, nil , nil);
    }

    // Handle a movie capture
    if ([mediaType isEqualToString:@"public.movie"]) {
        NSString *moviePath = [[info objectForKey: UIImagePickerControllerMediaURL] path];

        if (UIVideoAtPathIsCompatibleWithSavedPhotosAlbum(moviePath)) {
            UISaveVideoAtPathToSavedPhotosAlbum(moviePath, nil, nil, nil);
        }
    }

    [[picker parentViewController] dismissViewControllerAnimated:YES completion:nil];
}
```

# Utilizando AFNetworking - POST - Multipart Request

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager
manager];
NSDictionary *parameters = @{@"foo": @"bar"};
NSURL *filePath = [NSURL fileURLWithPath:@"file://path/to/image.png"];
[manager
    POST:@"http://example.com/resources.json"
    parameters:parameters
    constructingBodyWithBlock:^(id<AFMultipartFormData> formData) {
        [formData appendPartWithFileURL:filePath name:@"image" error:nil];
    }
    success:^(AFHTTPRequestOperation *operation, id
responseObject) {
        NSLog(@"Success: %@", responseObject);
    }
    failure:^(AFHTTPRequestOperation *operation, NSError
*error) {
        NSLog(@"Error: %@", error);
    }
    ]];
```

# Utilizando AFNetworking - POST - Multipart Request

- Arquivos podem ser adicionadas via NSData ou NSURL:
  - `appendPartWithFileURL:name:error:`
  - `appendPartWithFileData:name:fileName:mimeType:`

# Exercício

- Faça uma aplicação que faz upload de uma imagem selecionada da galeria.
- Api em:  
`teste-aula-ios.herokuapp.com/home/api`



# Exercício - Parte 2

- Adicione uma tela de login na aplicação, anterior ao upload de imagem.

```
[manager POST: @"https://teste-aula-ios.herokuapp.com/users/sign_in.json"
  parameters:@{ @"user" : @{@"email": @"crystian@roadmaps.com.br",
@"password": @"12345678" } }
  success:^(AFHTTPRequestOperation * _Nonnull operation, id _Nonnull
responseObject) {
    NSLog(@"Login success");
  } failure:^(AFHTTPRequestOperation * _Nullable operation, NSError *
_Nonnull error) {
    NSLog(@"Login failure");
  }];
}
```

# Exercício simulado da avaliação

- Crie uma aplicação com:
  - Tela de login
  - Tela com lista de comentários
  - Tela de criação de comentário com foto
  - Lista deve possuir paginação
  - Comentários devem ser salvos no core-data
  - Componente de busca deve existir na tableview