Unidade 6

Aula 1

No curso de Android, criamos um backend estático. Ele simplesmente retornava um JSON que estava em um Dropbox. Este JSON era imutável e não condizia com o real objetivo da aplicação. A maioria das aplicações devem ser dinâmica e devem transcender os dados entre as diversas plataformas. Esta tarefa pode ser complicada sem um serviço rodando em um backend. Na maioria das vezes, quem desenvolve aplicativos (front-end) não se mete no desenvolvimento de serviços web (back-end) e vice-e-versa, pois são áreas muito distintas, mas o Firebase veio para ajudar o desenvolvedor front-end.

# Firebase

O desenvolvedor front-end muitas vezes fica preso em um escopo muito pequeno de possibilidades se não houver um serviço web por traz da aplicação. Neste caso ficamos totalmente dependente de desenvolvedores back-end.

Muitas vezes precisamos de um serviço web simples, de obter e adicionar dados de um banco de dados, e ficamos fadados a encontrar um serviço web pronto que supra estas necessidades, contratar um desenvolvedor para criar isto para nós, ou metermos a cara de desenvolvermos por nós mesmo. Isto tudo é um gasto de principalmente tempo, e tempo é o que menos temos.

Para suprir esta carência de um backend para criação de aplicações dinâmicas, surgiu o Firebase.

O Firebase é uma plataforma web voltada para a centralização de dados e informações de suas aplicações. Nele você poderá criar bancos de dados não relacionais (informações armazenadas em JSONs) para seu aplicativo consumir, poderá realizar autenticação de usuário, armazenar arquivos, executar testes nos seus aplicativos, coletar dados analíticos de uso do app entre muitas outras coisas. Tudo isto de forma online e gratuita.

Com o Firebase é simples. Podemos criar nosso próprio backend sem escrever uma linha sequer de código. Mas vamos entender o que é o Firebase.

## O que é o Firebase

O Firebase é uma plataforma móvel que ajuda a desenvolver aplicativos de alta qualidade, expandir sua base de usuários e lucrar mais. O Firebase é composto de recursos complementares que você pode combinar de acordo com suas necessidades.

O time de desenvolvimento do Firebase está instalado em São Francisco e em Mountain View, na California. A companhia foi fundade em 2011 por Andrew Lee James Tamplim. O produto inicial qua lançou o Firebase foi o banco de dados em tempo real, que provê uma API que permite que os desenvolvedores armazenem e sincronizem dados entre múltiplos clientes. Ao longo do tempo, ele foi expandido para outras linhas de produto que o transformou em um pacote completo para desenvolvimento de aplicativos.

O Firebase foi comprado pela Google em outubro de 2014, que por sua vez adicionou uma quantidade significativa de novos recursos na plataforma em maio de 2016.

## Recursos do Firebase

O Firebase ajuda você a desenvolver aplicativos de alta qualidade, expandir sua base de usuários e lucrar mais. Cada recurso funciona de forma independente, mas juntos funcionam ainda melhor.

Exitem diversos recursos que podem ser alcançados com o Firebase. No site do Firebase temos as seguintes premissas:

* **Seja rápido**: O Firebase é uma plataforma móvel que ajuda a desenvolver aplicativos de alta qualidade, expandir sua base de usuários e lucrar mais. O Firebase é composto de recursos complementares que você pode combinar de acordo com suas necessidades.
* **Esqueça a infraestrutura**: Implementar o Firebase é rápido e fácil. Com APIs intuitivas reunidas em um só SDK, você pode se concentrar em solucionar os problemas dos clientes sem perder tempo criando uma infraestrutura complexa.
* **Tome decisões inteligentes baseadas em dados**: O Firebase Analytics é a solução de análise gratuita e ilimitada integrada diretamente ao Firebase. Obtenha informações sobre seus usuários, de cliques em anúncios ao uso do aplicativo. O Firebase Analytics trabalha junto com outros recursos do Firebase. Assim, você pode executar ações com base em quaisquer dados, de taxas de cliques a falhas do aplicativo.
* **Trabalhe em qualquer plataforma**: Forneça aplicativos multiplataformas com APIs reunidas em SDKs únicos para iOS, Android, JavaScript e C++. Expanda para outras plataformas sem modificar sua infraestrutura.

FIRAuth.auth()?.addStateDidChangeListener(){ (auth, user) in

if ((user) != nil) {

var email = user?.email

// ...

}

}

FIRAuth.auth()?.signIn(withEmail: email, password: password) { (\_, error) in

if ((error) != nil) {

// ...

}

}

* **Comece gratuitamente, expanda com facilidade:** A maioria dos recursos do Firebase é gratuita para sempre em qualquer escala. Nossos recursos pagos têm uma camada gratuita generosa e dois planos pagos para quando você começar a expandir seus negócios. Quando seu aplicativo ficar famoso, você não precisará se preocupar em expandir o código do seu servidor ou provisionar mais capacidade. O Firebase cuidará disso para você.
* **Receba suporte gratuitamente:** Oferecemos suporte gratuito por e-mail a cada desenvolvedor, e a equipe do Firebase e os especialistas do Google Developer são ativos em comunidades on-line como o Stack Overflow e o GitHub. Estamos aqui para ajudar.

Nós iremos estudar duas ferramentas em específico para aplicar em nosso projeto: o Firebase Authentication e o Firebase Database.

## Firebase Database

Armazene e sincronize dados com o banco de dados na nuvem NoSQL (não relacional). Os dados são sincronizados com todos os clientes conectados em tempo real e permanecem disponíveis quando seu aplicativo é desconectado.

O Firebase Realtime Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria aplicativos multiplataforma com nossos SDKs para iOS, Android e JavaScript, todos os seus clientes compartilham uma instância de Realtime Database e automaticamente recebem atualizações com os dados mais recentes.

### Principais funcionalidades

* **Tempo real:** Em vez de solicitações HTTP típicas, o Firebase Realtime Database usa a sincronização de dados, onde sempre que os dados são alterados, qualquer dispositivo conectado receberá esta atualização em milissegundos. Ele fornece experiências colaborativas e imersivas sem pensar em códigos de rede.
* **Off-line:** Aplicativos do Firebase permanecem responsivos mesmo off-line, pois o Firebase Realtime Database SDK mantém seus dados em cache no disco. Quando a conectividade é restabelecida, o cliente recebe as alterações perdidas, sincronizando-se ao estado atual do servidor.
* **Acessível a todos dispositivos clientes:** O Firebase Realtime Database pode ser acessado diretamente de um dispositivo móvel ou navegador da web; não é preciso ter um servidor de aplicativos. A segurança e a validação dos dados estão disponíveis por meio das regras de segurança do Firebase Realtime Database, regras baseadas em expressões que são executadas quando dados são lidos ou gravados.

### Como funciona?

O Firebase Realtime Database permite que você crie aplicativos colaborativos avançados ao permitir o acesso seguro ao banco de dados diretamente através do código no lado do cliente (front-end). Os dados são mantidos localmente, mesmo off-line, e eventos em tempo real continuam a ser acionados, proporcionando uma experiência responsiva ao usuário final. Quando o dispositivo recupera a conexão, o Realtime Database sincroniza as alterações nos dados locais com as atualizações remotas que ocorreram quando o cliente estava off-line, mesclando qualquer conflito automaticamente.

O Realtime Database fornece uma linguagem de regras baseadas em expressões flexíveis, chamadas regras de segurança do Firebase Realtime Database, que define como seus dados devem ser estruturados e quando podem ser lidos e gravados. Junto da integração com o Firebase Authentication, os desenvolvedores podem definir quem tem acesso a quais dados e como esses dados podem ser acessados.

O Realtime Database é um banco de dados NoSQL e, como tal, tem diferentes otimizações e recursos em comparação a um banco de dados relacional. A Realtime Database API foi projetada para permitir somente operações que podem ser executadas com rapidez. Isso permite que você desenvolva uma ótima experiência em tempo real que pode atender a milhões de usuários sem comprometer sua capacidade de resposta. Por isso, é importante considerar como os usuários precisam acessar seus dados e estruturá-los de forma adequada.

## Firebase Authentication

A maioria dos aplicativos precisam saber a identidade de um usuário. Saber a identidade de um usuário permite que o aplicativo salve seus dados na nuvem de maneira segura e forneça a mesma experiência personalizada em todos os dispositivos desse usuário.

O Firebase Authentication fornece serviços de backend, SDKs fáceis de usar e bibliotecas de UI para autenticar os usuários em seu aplicativo, prontas para o uso. Ele oferece suporte a autenticação usando senhas, provedores populares de identidades de identidades federadas, como o Google, Facebook e Twitter, e muito mais.

O Firebase Authentication é estreitamente integrado com outros serviços do Firebase e utiliza padrões do setor como OAuth 2.0 e OpenID Connect, portanto, pode ser facilmente integrado com seu backend personalizado.

### Principais funcionalidades

Você pode conectar usuários ao seu aplicativo Firebase usando o FirebaseUI como uma solução de autenticação de drop-in completa ou usando o Firebase Authentication SDK para integrar manualmente um ou mais métodos de login no seu aplicativo. Por enquanto o FirebaseUI está em fase de testes (beta) então iremos realizar a integração manualmente na próxima aula.

* **Autenticação baseada em e-mail e senha:** Autentique usuários com endereços de e-mail e senhas. O Firebase Authentication SDK oferece métodos para criar e gerenciar usuários que usem endereços de e-mail e senhas para fazer login. É possível também enviar e-mails de redefinição de senha.
* **Integração de provedores de identidades federadas:** Autentique usuários ao integrar com provedores de identidades federadas. O Auth oferece métodos que permitem que os usuários façam login com contas do Google, Facebook, Twitter e GitHub.
* **Integração de sistemas de autenticação personalizados:** Conecte o sistema de login existente do aplicativo ao Firebase Authentication SDK e obtenha acesso ao Firebase Realtime Database e a outros serviços do Firebase.
* **Autenticação anônima:** Use os recursos do Firebase que exigem autenticação sem que os usuários precisem fazer login primeiro ao criar contas anônimas temporárias. Se o usuário depois optar por fazer login, você poderá transformar a conta anônima em uma conta regular para que ele continue de onde parou.

### Como funciona?

Para conectar um usuário ao seu aplicativo, primeiro você deve receber credenciais de autenticação do usuário. Essas credenciais podem ser o endereço de e-mail e a senha desse usuário ou um token OAuth de um provedor de identidades federadas. Em seguida, passe essas credenciais para o Firebase Authentication SDK. Os serviços de backend do Firebase verificarão essas credenciais e retornarão uma resposta para o cliente.

Após um login bem-sucedido, você poderá acessar as informações de perfil básicas do usuário e controlar o acesso do usuário aos dados armazenados em outros produtos do Firebase. Você também pode usar o token de autenticação fornecido para verificar a identidade dos usuários nos seus próprios serviços de backend.

Observação: Por padrão, usuários autenticados podem ler e gravar dados no Firebase Realtime Database e no Firebase Storage.

## Criando um projeto no Firebase

No site do Firebase você poderá, além de encontrar documentações e tutoriais, gerenciar sua aplicação de forma 100% online, onde você estiver.

Este recurso de gerenciamento é chamado Firebase Console. É nele que você criará novos projetos e aplicações, fará modificações, gerenciará dados analíticos dos seus aplicativos e armazenará todos os dados que suas aplicações irão consumir (Firebase Database). Além disso, é aqui que é feito o gerenciamento de login de usuário (Firebase Auth) e podemos também fazer armazenamento de arquivos (Firebase Storage), como músicas, vídeos, documentos que serão posteriormente utilizados pela nossa aplicação. No nosso caso iremos armazenar as imagens dos álbuns musicais tal como as próprias músicas.

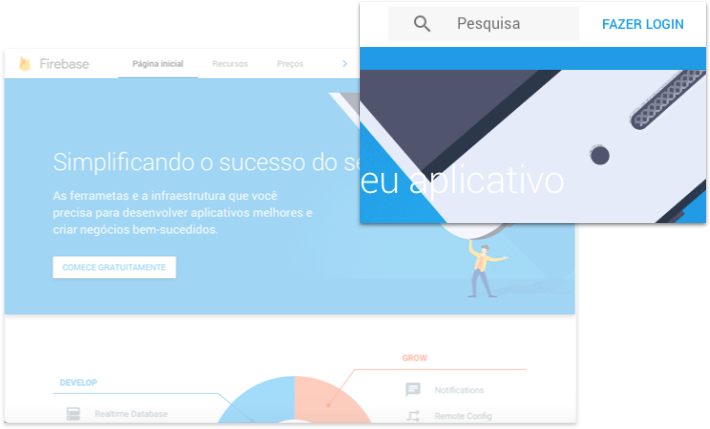
Para começar a utilizar o projeto, temos 3 etapas. Uma delas foi feito na unidade 5, que é adicionar o Firebase via CocoaPods no projeto. As outras duas é a criação de um novo projeto no Firebase e Configurar o projeto no Xcode. Vamos então fazer isto.

### Criando um novo projeto no Firebase Console

1. Entre no site do firebase: <https://firebase.google.com/?hl=pt-br>



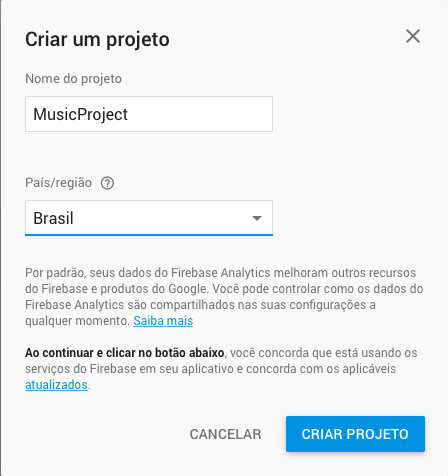
1. Faça login com sua conta google:



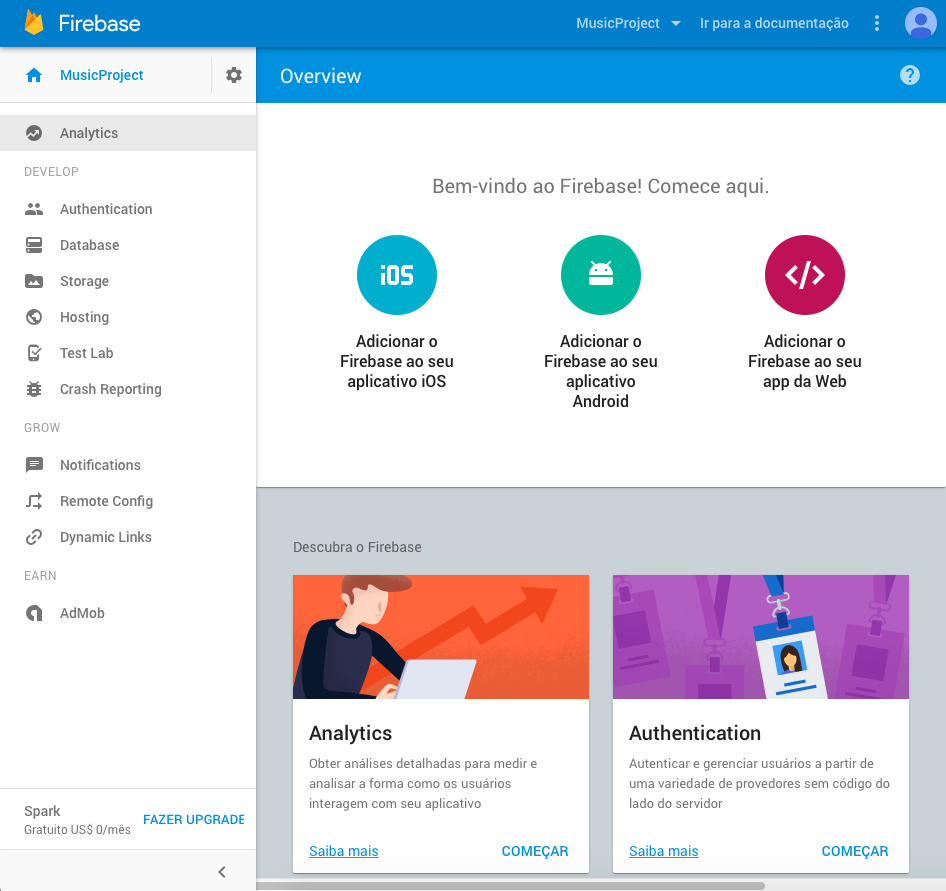
1. Se você ainda não tem uma conta Google, crie uma.
2. Ao fazer login, você estará no Firebase Console. Clique na opção **CRIAR NOVO PROJETO**:



1. Escolha o nome **MusicProject**, em selecione País/Região o **Brasil** e clique em **CRIAR PROJETO:**



1. O projeto estará criado e a seguinte tela irá aparecer:



Com estes passos temos um projeto pronto para ser integrado nas plataformas iOS, Android e Web. Cada uma tem sua forma de integração. Aqui veremos como integrar em um aplicativo iOS.

Ao adicionar uma plataforma, começamos a receber dados analíticos provenientes dela. Após a configuração do projeto no Xcode, o aplicativo passa a mandar dados básicos de seções para o Firebase. Com isto você poderá gerenciar com mais precisão a taxa de uso do seu aplicativo para o grande público. Neste curso não vamos focar nisso, mas pode ser interessante para um estudo futuro.

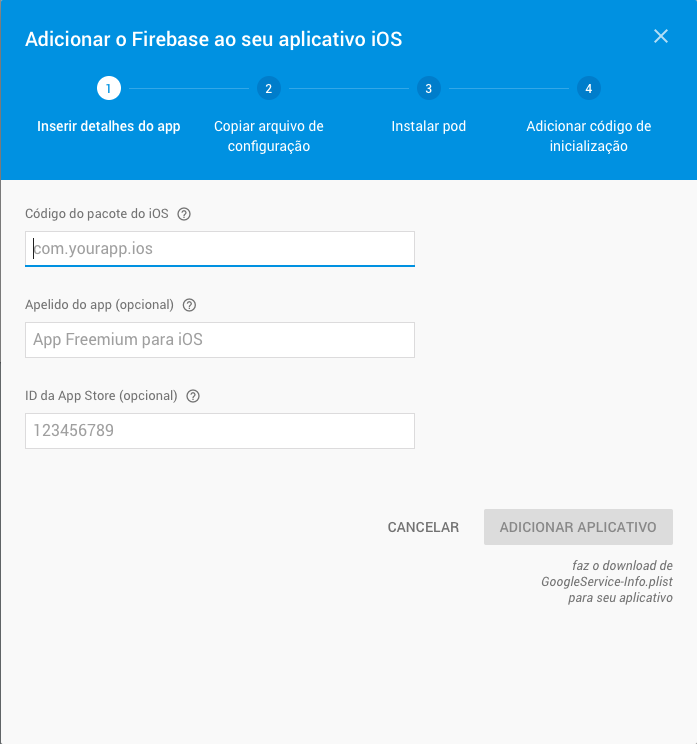
### Configurando o projeto no Xcode

O Firebase console agora precisa se comunicar com o aplicativo para obter dados básicos e para que os componentes que iremos utilizar futuramente funcionem corretamente. Vamos então aplicar as configurações no nosso app.

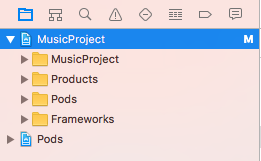
1. Clique na opção **Adicionar Firebase ao seu aplicativo iOS**.



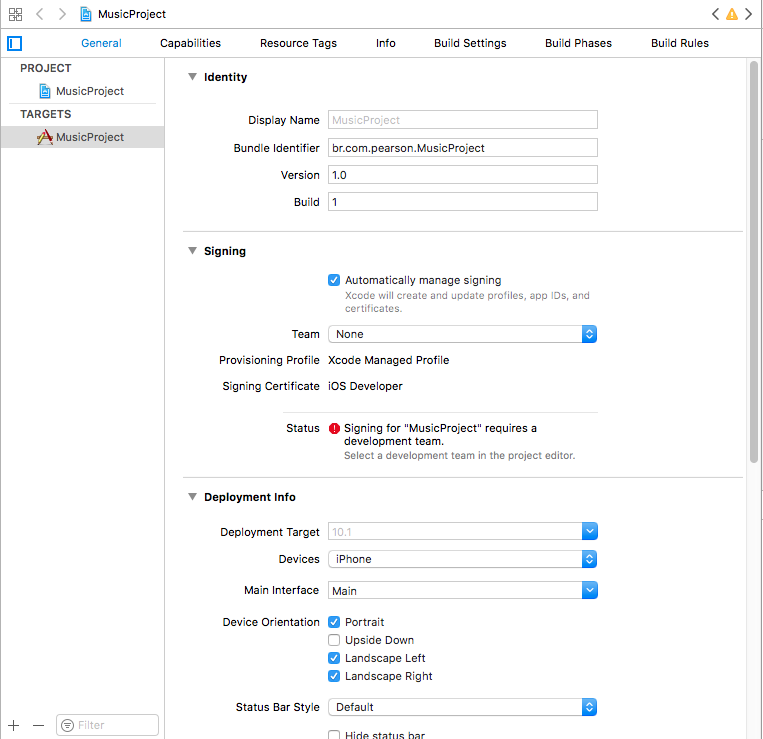
1. Na seguinte tela você terá que preencher apenas o campo **Código do pacote do iOS**. Veremos onde conseguir esta informação no passo seguinte.



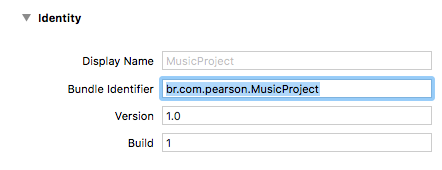
1. O código do pacote do iOS pode ser encontrado no seu projeto do Xcode, então:
   1. Abra o projeto **MusicProject do seu Xcode** e localize o arquivo de configuração de projeto. Ele estará no topo do painel de navegação:



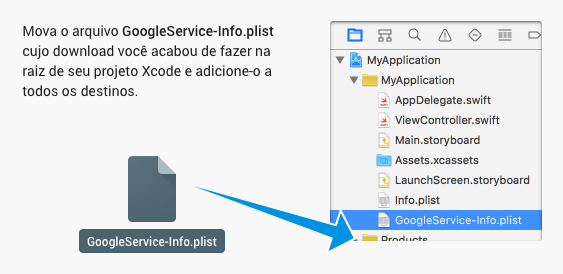
* 1. O arquivo terá esta aparência:



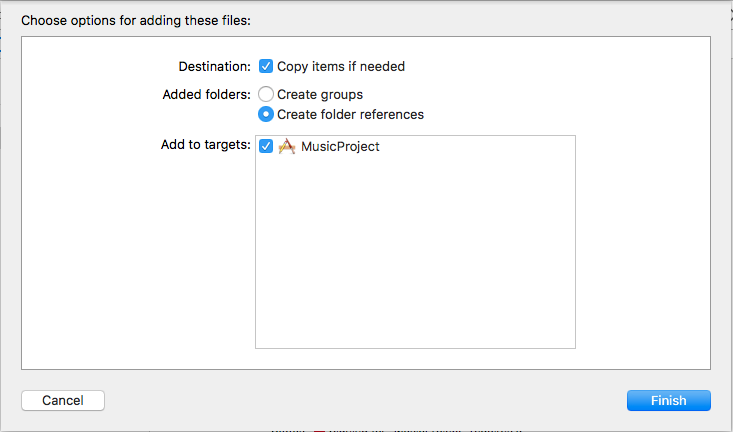
* 1. Agora basta pegarmos a informação que precisamos. Ela se encontra no campo **Bundle ID**. Copie-o para utilizarmos no Firebase.



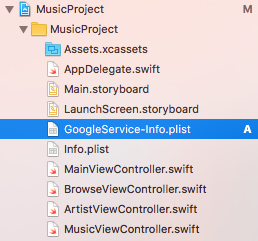
1. Voltando ao Firebase Console, insira **o bundle ID** no campo **Código do pacote do iOS.**
2. Clique em **ADICIONAR APLICATIVO** para continuar.
3. Na próxima tela o arquivo **GoogleService-Info.plist** será baixado automaticamente. Adicione-o ao seu projeto, como descrito na imagem do site. Basta arrastá-lo:



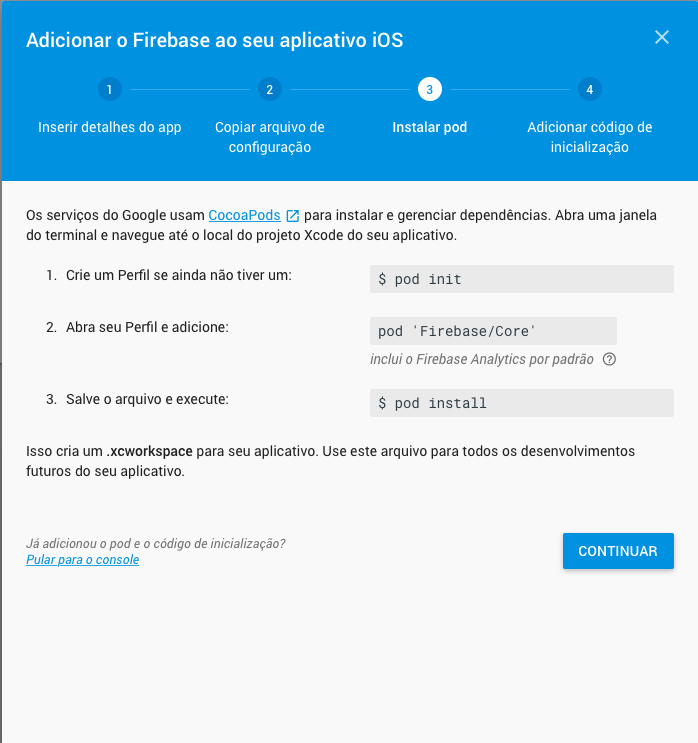
1. Antes de adicionar o Xcode mostrará uma tela com configurações. Selecione a opção **Copy items if needed**, isto fará com que o arquivo seja copiado na pasta do projeto. Em seguida clique em **Finish**.



1. O novo arquivo estará adicionado:

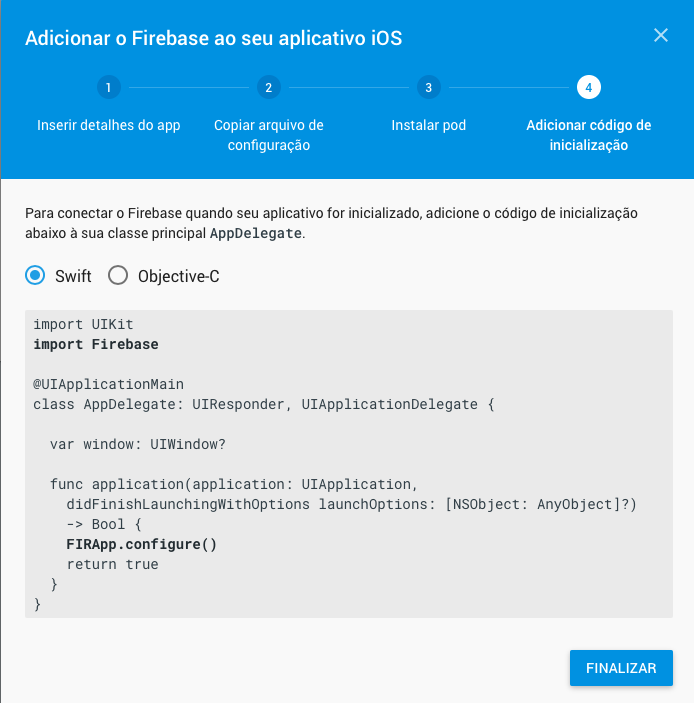


1. Na próxima tela do console do Firebase, é pedido que instalemos o Pod. Já fizemos isto na aula 3 da unidade 5, então pule para o próximo passo.



1. Para o último passo, precisamos inicializar o código no nosso projeto. Então siga as instruções dadas na nova tela do Firebase console:

*"Para conectar o Firebase quando seu aplicativo for inicializado, adicione o código de inicialização abaixo à sua classe principal AppDelegate."*



Com estas configurações estamos prontos para consumir os serviços do Firebase. Como dito acima, vamos utilizar o serviço Firebase Auth e o Firebase Database, para fazer login e obter JSONs com informações de músicas respectivamente. Outro serviço que será utilizado, mas não vamos abordar nesta aula é o Firebase Storage. Nele armazenaremos as músicas e imagens dos álbuns.

## Resumo

Por enquanto só aprendemos um pouco sobre o que é o Firebase, mas já conseguimos perceber que é uma ferramenta poderosa e útil para criarmos apps inteligentes, responsivos e dinâmicos. Com ele podemos centralizar as informações da nossa aplicação, assim o app em cada plataforma só se preocupa em exibi-la ou manda-la para o back-end do Firebase. Neste ponto, podemos dizer que o Firebase nos dá o poder de um desenvolvedor Full-Stack, que é o desenvolvedor que cuida tanto do front-end quanto do back-end de uma aplicação. Na aula seguinte faremos uso do Firebase Authentication, e na unidade 7 iremos utilizar o Firebase Realtime Database para finalmente tocar nossas músicas no app.

Aula 2

# Login com Firebase

## Firebase Authentication

O Firebase Authentication fornece serviços de backend, SDKs fáceis de usar e bibliotecas de IU prontas para o uso para autenticar os usuários em seu aplicativo. Ele oferece suporte a autenticação usando senhas, provedores populares de identidades de identidades federadas, como o Google, Facebook e Twitter, e muito mais.

### Usuários em projetos do Firebase

Um objeto Firebase User representa a conta de um usuário que se inscreveu em um aplicativo no seu projeto do Firebase. Aplicativos geralmente têm muitos usuários registrados e todos os aplicativos em um projeto do Firebase compartilham um banco de dados de usuários.

Uma instância de Firebase User independe de uma instância de Firebase Auth. Isso significa que você pode ter várias referências a diferentes usuários no mesmo contexto e ainda chamar qualquer um de seus métodos.

Propriedades do usuário

Um Firebase User tem um conjunto fixo de propriedades básicas — um ID exclusivo, um endereço de e-mail primário, um nome e um URL de foto — armazenadas no banco de dados de usuários do projeto, que pode ser atualizado pelo usuário (iOS, Android, web). Você não pode adicionar outras propriedades diretamente no objeto Firebase User; em vez disso, pode armazenar as propriedades adicionais em seu Firebase Realtime Database.

Quando um usuário se inscreve em seu aplicativo pela primeira vez, os dados de perfil desse usuário são preenchidos com as informações disponíveis:

* Se o usuário se inscreveu com um endereço de e-mail e uma senha, somente a propriedade do endereço de e-mail primário será preenchida
* Se o usuário se inscreveu com um provedor de identidades federadas, como o Google ou o Facebook, as informações de conta disponibilizadas por esse provedor serão usadas para preencher o perfil do usuário do Firebase
* Se o usuário se inscreveu com seu sistema de autenticação personalizado, você deve adicionar as informações desejadas explicitamente no perfil do usuário do Firebase.

Quando uma conta de usuário é criada, você pode recarregar as informações do usuário para incorporar qualquer alteração que ele possa ter feito em outro dispositivo.

### Provedores de login

Você pode conectar usuários em seus aplicativos do Firebase usando diversos métodos: um endereço de e-mail e uma senha, provedores de identidades federadas e seu sistema de autenticação personalizado. Você pode associar mais de um método de login com um usuário: por exemplo, os usuários podem fazer login na mesma conta usando um endereço de e-mail e uma senha ou usando o Google Sign-In.

Uma instância de Firebase User acompanha todos os provedores vinculados ao usuário. Isso permite que você atualize as propriedades do perfil vazio usando as informações fornecidas por um provedor. Consulte Gerenciar usuários (iOS, Android, Web).

### Usuário atual

Quando um usuário se inscreve ou faz login, esse usuário se torna o usuário atual da instância de Auth. A instância de Firebase Auth persiste o estado do usuário, portanto, atualizar a página (em um navegador) ou reiniciar o aplicativo não perde as informações do usuário.

Quando o usuário faz logout, a instância de Auth deixa de manter uma referência ao objeto User e de persistir seu estado; não há mais um usuário atual. No entanto, a instância do usuário continua totalmente funcional: se você mantiver uma referência a ela, ainda será possível acessar e atualizar os dados do usuário.

### Ciclo de vida do usuário

A maneira recomendada de acompanhar o estado atual da instância do Firebase Auth é usar ouvintes (também chamados de "observadores" no JavaScript). Um ouvinte do Auth é notificado sempre que algo relevante acontece com o objeto Auth. Consulte Gerenciar usuários (iOS, Android, web).

Um ouvinte do Auth é notificado nas seguintes situações:

* O objeto Auth termina de inicializar e um usuário já fez login em uma sessão anterior ou foi redirecionado do fluxo de login de um provedor de identidades
* Um usuário faz login (o usuário atual é definido)
* Um usuário faz logout (o usuário atual se torna nulo)
* O token de acesso do usuário atual é atualizado. Esse caso pode acontecer nas seguintes condições:
  + O token de acesso expira: essa é uma situação comum. O token de atualização é usado para obter um novo conjunto de tokens válidos.
* O usuário altera a senha: O Firebase emite novos tokens de acesso e atualização e considera os antigos como expirados. Isso automaticamente desconecta o usuário de todos os dispositivos, por motivos de segurança
* O usuário é reautenticado: algumas ações exigem que as credenciais do usuário tenham sido emitidas recentemente, como excluir uma conta, definir um endereço de e-mail primário e alterar uma senha. Em vez de desconectar o usuário e conectá-lo novamente, obtenha novas credenciais do usuário e passe-as para o método de reautenticação do objeto User.

## Gerenciar usuários no Firebase

### Criar um usuário

Para criar um novo usuário no projeto Firebase, chame o método createUser(withEmail:password:completion:) ou conecte um usuário pela primeira vez usando um provedor de identidades federadas, como o Google Sign-In ou o login do Facebook.

Você também pode criar novos usuários autenticados com senha na seção Authentication do Firebase console, na página Users.

### Obter o usuário atualmente conectado

A maneira recomendada de obter o usuário atual é definir um ouvinte no objeto Auth:

FIRAuth.auth()?.addStateDidChangeListener { auth, user in

if let user = user {

// O usuário está logado.

} else {

// O usuário não está logado.

}

}

Ao usar um ouvinte, você garante que o objeto Auth não esteja em um estado intermediário — como inicialização — ao obter o usuário atual.

Você também pode obter o usuário conectado atualmente usando a propriedade currentUser . Se um usuário não estiver conectado, currentUser será nulo:

if let user = FIRAuth.auth()?.currentUser {

// O usuário está logado.

} else {

// O usuário não está logado.

}

Observação: currentUser também poderá ser nulo, pois o objeto Auth não concluiu a inicialização. Se você usar um ouvinte para acompanhar o status de login do usuário, não será preciso processar este caso.

### Obter o perfil de um usuário

Para obter as informações de perfil de um usuário, use as propriedades de uma instância de FIRUser. Por exemplo:

if let user = FIRAuth.auth()?.currentUser {

let name = user.displayName

let email = user.email

let photoUrl = user.photoURL

let uid = user.uid;

} else {

// O usuário não está logado.

}

### Obter as informações de perfil específicas de provedor de um usuário

Para obter as informações de perfil recuperadas dos provedores de login vinculados a um usuário, use a propriedade providerData. Por exemplo:

if let user = FIRAuth.auth()?.currentUser {

for profile in user.providerData {

let providerID = profile.providerID

let uid = profile.uid; // Provider-specific UID

let name = profile.displayName

let email = profile.email

let photoURL = profile.photoURL

}

} else {

// O usuário não está logado.

}

### Atualizar o perfil de um usuário

Você pode atualizar as informações de perfil básicas de um usuário (o nome de exibição do usuário e o URL da foto do perfi) com a classe FIRUserProfileChangeRequest. Por exemplo:

let user = FIRAuth.auth()?.currentUser

if let user = user {

let changeRequest = user.profileChangeRequest()

changeRequest.displayName = "Jane Q. User"

changeRequest.photoURL =

NSURL(string: "https://example.com/jane-q-user/profile.jpg") as URL?

changeRequest.commitChanges { error in

if let error = error {

// Um erro aconteceu.

} else {

// Perfil atualizado.

}

}

}

### Definir o endereço de e-mail de um usuário

Você pode definir o endereço de e-mail de um usuário com o método updateEmail(completion:). Por exemplo:

let user = FIRAuth.auth()?.currentUser

user?.updateEmail("user@example.com") { error in

if let error = error {

// Um erro aconteceu.

} else {

// Email atualizado.

}

}

Importante: Para definir o endereço de e-mail de um usuário, ele deve ter feito login recentemente. Consulte Reautenticar um usuário.

### Definir a senha de um usuário

Você pode definir a senha de um usuário com o método updatePassword(completion:). Por exemplo:

let user = FIRAuth.auth()?.currentUser

let newPassword = getRandomSecurePassword()

user?.updatePassword(newPassword) { error in

if let error = error {

// Um erro aconteceu.

} else {

// Password atualizado.

}

}

Importante: Para definir a senha de um usuário, ele deve ter feito login recentemente. Consulte Reautenticar um usuário.

### Enviar um e-mail de redefinição de senha

Você pode enviar um e-mail de redefinição de senha para um usuário com o método sendPasswordResetWithEmail:completion:. Por exemplo:

let email = "user@example.com"

FIRAuth.auth()?.sendPasswordReset(withEmail: email) { error in

if let error = error {

// Um erro aconteceu.

} else {

// E-mail de redefinição de senha enviado.

}

}

Você pode personalizar o modelo de e-mail usado na seção de Authentication do Firebase console, na página Email Templates. Consulte Modelos de e-mail na Central de Ajuda do Firebase.

Você também pode enviar e-mails de redefinição de senha pelo Firebase console.

### Excluir um usuário

Você pode excluir uma conta de usuário com o método deleteWithCompletion. Por exemplo:

let user = FIRAuth.auth()?.currentUser

user?.delete { error in

if let error = error {

// Um erro aconteceu.

} else {

// Conta deletada

}

}

Você também pode excluir usuários da seção Authentication do Firebase console, na página Users.

Importante: Para excluir um usuário, ele deve ter feito login recentemente. Consulte Reautenticar um usuário.

### Reautenticar um usuário

Algumas ações que apresentam riscos à segurança — como excluir uma conta, definir um endereço de e-mail primário e alterar uma senha — exigem que o usuário tenha feito login recentemente. Se você realizar uma dessas ações e o usuário tiver feito login há muito tempo, a ação falhará com o erro FIRAuthErrorCodeCredentialTooOld . Quando isso ocorrer, reautentique o usuário obtendo novas credenciais de login do usuário e passando-as para reauthenticate. Por exemplo:

let user = FIRAuth.auth()?.currentUser

var credential: FIRAuthCredential

user?.reauthenticate(with: credential) { error in

if let error = error {

// Um erro aconteceu

} else {

// Usuário re-autenticado.

}

}

## Implementando login por e-mail e senha

Nesta aula iremos implementar o básico para obtermos um login via e-email e senha utilizando o Firebase.

Para esta aula criamos a tela de login para você ela se encontra no material de apoio

**Material de apoio**

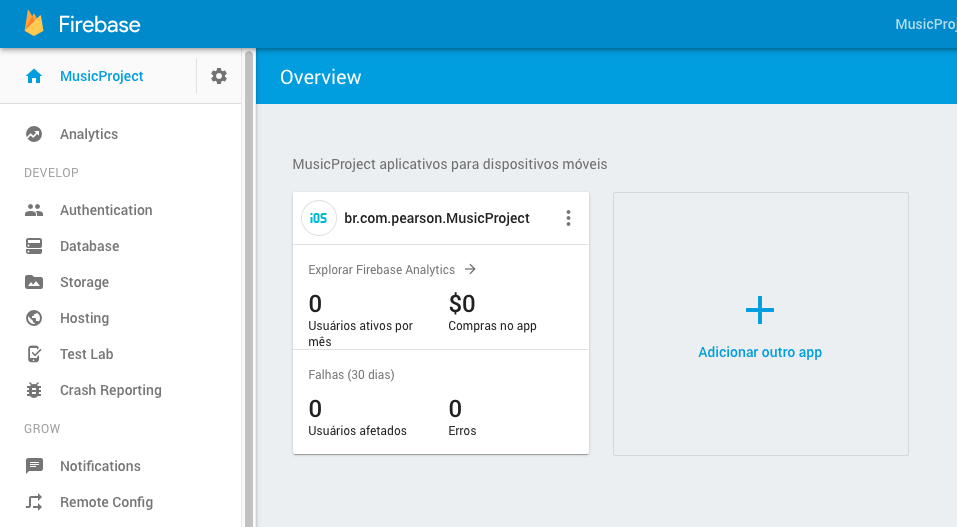
O ponto de partida para esta atividade se encontra no arquivo **Unidade\_6\_-\_Aula\_2\_-\_Exemplo\_1.zip**, na pasta do material de apoio.

Nesta aula termos que utilizar o Firebase. Então você precisará utilizar o mesmo arquivo **GoogleServices-Info.plist** que você utilizou na aula passada. Caso você opte por utilizar o arquivo que já está no material de apoio, você não conseguirá ver os usuários registrados pelo console do firebase, mas conseguirá prosseguir até o final da aula normalmente.

### Ativando o provedor de login

Para conseguirmos usar este recurso, devemos primeiro habilitar a opção de login via email/senha no Firebase console. Siga os passos:

1. Entre no Firebase Console.
2. Entre no projeto MusicProject



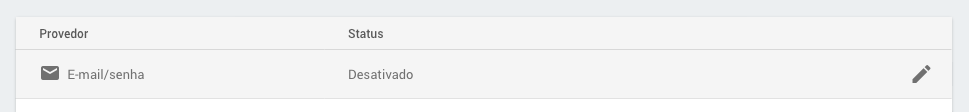
1. Clique na opção Athentication no menu lateral.



1. Clique em CONFIGURAR MÉTODO DE LOGIN



1. Selecione o provedor de login **E-mail/senha**



1. Ative-o e finalize



### Instalando dependências

Para utilizar o serviço de autenticação, precisamos instalá-lo via CocoaPods, então:

1. Abra o Podfile e adicione mais um pod:

pod 'Firebase/Auth'

1. Seu Podfile ficará parecido com este:

target 'MusicProject' do

use\_frameworks!

pod 'Firebase/Core'

pod 'Firebase/Auth'

end

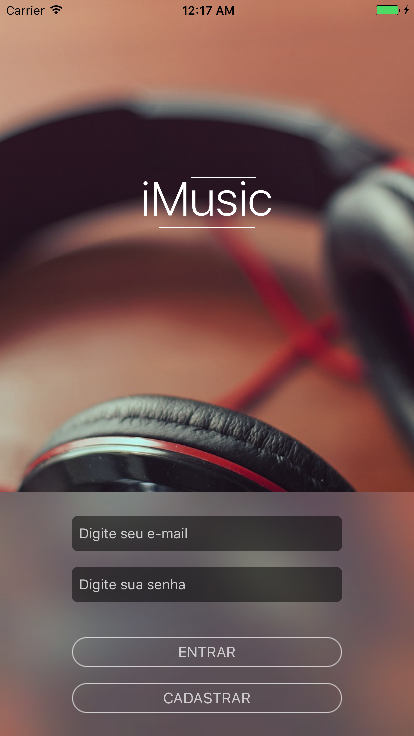
1. Abra seu terminal e navegue até o diretório do projeto, então execute o comando de instalação:

pod install

O Firebase Authentication será instalado depois disso.

### Implementando os métodos do SDK Firebase Authentication

Primeiro veja a tela de Login que foi criada:



Nela temos o campo e-mail e senha, normalmente. Também temos um botão para cadastro e outro para login (entrar).

A primeira ação que iremos implementar é o cadastro de usuários então siga os seguintes passos:

1. Abra o arquivo **LoginViewController.swift,** e importe a biblioteca Firebase. Insira a seguinte linha no topo do arquivo:

import Firebase

1. Veja que temos dois @IBActions, um para login outro para cadastro. Localize o método de cadastro **registerButtonTouchUpInside().**
2. Nele vamos inserir os métodos de cadastro. Primeiro vamos validar se os campos estão vazios. Crie um método:

func isValid() -> Bool {

if (emailTextField.text?.isEmpty)! || (passwordTextField.text?.isEmpty)! {

let alert = UIAlertController(title: "Atenção", message: "Todos os campos são obrigatórios. Preencha todos os campos.", preferredStyle: .alert)

let okAction = UIAlertAction(title: "OK", style: .default, handler: nil)

alert.addAction(okAction)

present(alert, animated: true, completion: nil)

return false

}

return true

}

DICA: Vamos precisar desta validação no método de Login, então separamos em um método. Este método mostra um Alert quando os campos se mostram inválidos (vazios)

1. No método **registerTouchUpInside()** faça uso do método de validação:

@IBAction func registerButtonTouchUpInside(\_ sender: UIButton) {

if isValid() {

// ...

}

}

1. Agora basta chamarmos o serviço do Firebase para cadastrarmos um novo usuário no backend:

@IBAction func registerButtonTouchUpInside(\_ sender: UIButton) {

if isValid() {

FIRAuth.auth()?.createUser(withEmail: emailTextField.text!, password: passwordTextField.text!) { (user, error) in

if error != nil {

// Tratamento do erro

}else{

let alert = UIAlertController(title: "Sucesso", message: "Você foi cadastrado com sucesso!", preferredStyle: .alert)

let okAction = UIAlertAction(title: "OK", style: .default) { \_ in

self.performSegue(withIdentifier: "ShowHomeScreenSegue", sender: self)

}

alert.addAction(okAction)

self.present(alert, animated: true, completion: nil)

}

}

}

}

1. Trate as possíveis mensagens de erro que o serviço poderá retornar:

@IBAction func registerButtonTouchUpInside(\_ sender: UIButton) {

if isValid() {

FIRAuth.auth()?.createUser(withEmail: emailTextField.text!, password: passwordTextField.text!) { (user, error) in

if error != nil {

var alertMessage = ""

switch (error as! NSError).code {

case FIRAuthErrorCode.errorCodeInvalidEmail.rawValue:

alertMessage = "E-mail inválido."

break

case FIRAuthErrorCode.errorCodeEmailAlreadyInUse.rawValue:

alertMessage = "E-mail já está em uso."

break

case FIRAuthErrorCode.errorCodeWeakPassword.rawValue:

alertMessage = "Sua senha é muito fraca. Por favor registre outra"

break

default:

alertMessage = "Um erro inesperado aconteceu. Tente novamente mais tarde."

break

}

let alert = UIAlertController(title: "Erro", message: alertMessage, preferredStyle: .alert)

let okAction = UIAlertAction(title: "OK", style: .default, handler: nil)

alert.addAction(okAction)

self.present(alert, animated: true, completion: nil)

}else{

let alert = UIAlertController(title: "Sucesso", message: "Você foi cadastrado com sucesso!", preferredStyle: .alert)

let okAction = UIAlertAction(title: "OK", style: .default) { \_ in

self.performSegue(withIdentifier: "ShowHomeScreenSegue", sender: self)

}

alert.addAction(okAction)

self.present(alert, animated: true, completion: nil)

}

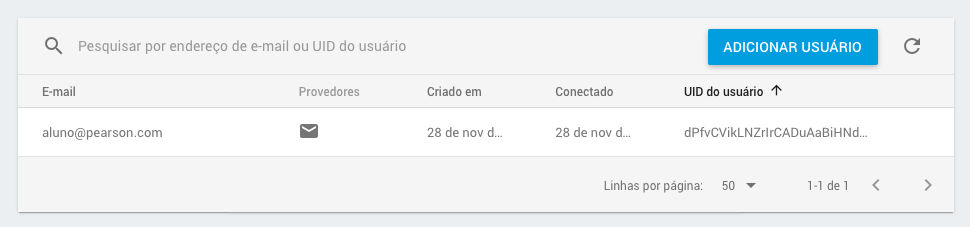
}

}

}

DICA: Este método trata erros conhecidos do Firebase. Todos estes erros estão documentados no site do Firebase

1. O método de cadastro está pronto. Execute o app.
2. Cadastre-se
3. Uma tela de sucesso aparecerá. Se você inserir dados inválidos, um erro aparecerá
4. Você poderá olhar no Firebase Console o novo usuário cadastrado:



1. O método de login será similar:

Feito o procedimento de login e cadastro, agora podemos utilizar o recurso de login automático do Firebase, ou seja, uma vez logado, o usuário não precisará fornecer suas credenciais novamente. Para isto siga os passos:

1. Abra o arquivo AppDelegate.swift
2. No método application(didFinishLaunchingWithOptions:) insira as seguinte instruções:

let mainStoryboard = UIStoryboard(name: "Main", bundle: Bundle.main)

if FIRAuth.auth()?.currentUser == nil {

window?.rootViewController = mainStoryboard.instantiateViewController(withIdentifier: "LoginViewController")

}

1. No Storyboard Main, configure a MainViewController como Initial View Controller.

É só isto que precisamos. O que este trecho faz é:

* Obter a referência do Storyboard Main
* Verificar não existe um usuário autenticado através do SDK do Firebase
* Se sim, pulamos para a tela com Intentifier “LoginViewController” que é a nossa cena de login. Se não deixamos o fluxo normal da aplicação.