Unidade 6

Aula 1

No curso de Android criamos um *backend* (que é o lado do servidor, mais visível ao programador) 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 deve ser dinâmica e 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 (*frontend,* que é olado do cliente, mais visível ao usuário) não se mete no desenvolvimento de serviços *web* (*backend*) e vice-e-versa, pois são áreas muito distintas, mas o Firebase veio para ajudar o desenvolvedor frontend.

# Firebase

O desenvolvedor *frontend* 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 dependentes de desenvolvedores *backend*.

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

Para suprir a 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. Nela você poderá criar bancos de dados não relacionais (informações armazenadas em JSONs) para seu aplicativo consumir, poderá realizar autenticação de usuários, 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 podemos criar nosso próprio *backend* sem escrever uma linha sequer de código, tudo de forma bem simples. Mas vamos entender o que é o Firebase.

## O que é o Firebase?

Mas o que é realmente o Firebase? É um produto da Google disponível online que oferece um serviço BaaS (Backend as a Service), em outras palavras, é um serviço de backend.

Ele basicamente oferece serviços de banco de dados não relacionais, serviços de sincronização em *real time* para aplicativos móveis e sistemas web e possui SDKs nativos para Android, iOS e Javascript que possibilitam o uso *off-line* destes dados que ele gerencia. Além disso, nele estão integrados os serviços de Google Analytics, Storage (armazenamento de arquivos na nuvem), ferramentas de testes para integração contínua dos apps e muito mais.

Recentemente a Google adquiriu o Fabric da Twitter e está prestes a integrá-lo no conjunto de ferramentas do Firebase.

O Fabric é uma plataforma online onde podemos integrar nossos aplicativos para receber relatório de erros (com o Fabric Crashlytics), dados analíticos de uso do app (com o Fabric Answers), distribuição para testes beta (com o Fabric Beta) e muitas outras ferramentas integradas com a plataforma Fabric (como o Digits, Fastlane e etc). Então podemos esparar uma melhoria ainda maior do Firebase, na qual irá suprir muitas necessidades no desenvolvimento de aplicativos móveis.

O time de desenvolvimento do Firebase está instalado em São Francisco e em Mountain View, na Califórnia. A companhia foi fundada em 2011 por Andrew Lee James Tamplim. O produto inicial que 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 transformaram 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.

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

* **Seja rápido**: Uma premissa da computação é “não reinvente a roda”. Isto significa que se temos soluções à nossa disposição, elas certamente devem ser utilizadas. O Firebase é um BaaS e possui operações de backend básico, então porque criar um novo? Além disso ele é composto de diversos outros recursos que se complementam e você pode utilizá-los conforme suas necessidades.
* **Esqueça a infraestrutura**: Utilizar o SDK do Firebase é fácil. Possuindo APIs intuitivas, podemos nos concentrar em problemas do nosso produto ao invés de perder tempo com a criação de uma infraestrutura de backend que no final fará exatamente o que o Firebase faz.
* **Tome decisões inteligentes baseadas em dados**: Recentemente o famoso Google Analytics bastante utilizado em sites e em apps foi integrado ao Firebase. Com ele temos uma solução de análise completa, gratuita e ilimitada, integrada diretamente ao Firebase. O Analytics pode obter dados de uso do seu aplicativo e ajuda-lo a tomar decisões rápidas e precisas, mostrando para você informações como taxas de cliques e falhas do app.
* **Trabalhe em qualquer plataforma**: Com o Firebase podemos unificar as informações de diversas plataformas em um único lugar. Com suas APIs reunidas em SDKs de diversas plataformas como iOS, Android, JavaScript e C++. Veja um exemplo de uso do SDK de iOS na linguagem Swift (existe também a versão Objective-C para iOS).

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:** Os recursos mais importantes e utilizados do Firebase são totalmente gratuitos para qualquer tamanho de projeto. Os recursos pagos possuem suas versões gratuitas (que são bem generosas) e outros dois planos pagos que possuem preços bem abaixo da média do mercado de BaaS.
* **Receba suporte gratuitamente:** Além de tudo isso de bom, o Firebase oferece suporte de atendimento ao usuário gratuitamente.

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

## Firebase Database

Dentre todos os serviços oferecidos pelo Firebase o Database é o mais atraente. Com ele podemos armazenar e sincronizar qualquer tipo de dado da nossa aplicação na nuvem em um formato NoSQL (ou banco de dado não relacional). Podemos utilizar os SDKs do Firebase Realtime Database para que os dados permaneçam no dispositivo quando desconectado, assim quando houver uma conexão com a internet o Database sincroniza automaticamente com a núvem.

DICA: Como não abordamos o assunto Banco de Dados neste curso, pois é tema para um outro curso bem grande, você pode não estar familiarizado com os termos NoSQL, Banco Relacional ou Não Relacional. Não se preocupe, já utilizávamos banco não relacional e não sabíamos. O JSON é uma forma de banco de dados não relacional, pois ele possui uma estrutura de dados que não precisam estar necessariamente relacionados entre si. Então tome banco de dados não relacional como sendo um banco de JSONs para este projeto. Indicamos que você pesquise um pouco sobre Banco de Dados em geral, mas isto não afetará esta aula.

### Principais funcionalidades

* **Tempo real:** O Firebase Realtime Data base não utiliza soluções HTTP típicas. Com seu SDK instalado no aplicativo, sempre que os dados são alterados, qualquer dispositivo que compartilha destes mesmos dados são atualizados em milissegundos. É uma alternativa essencial se você está desenvolvendo uma aplicação multiplataforma, por exemplo.
* **Off-line:** Não se preocupe com a conexão de internet. Os aplicativos integrados com este SDK permanecem responsivos mesmo off-line. Isto quer dizer que o Firebase irá gerenciar um cache local no seu smartphone, possibilitando que você continue utilizando seus recursos. Quando a conectividade é reestabelecida, você receberá imediatamente as alterações que outro usuário pode ter feito na base de dados e também irá mandar instantaneamente as alterações que você fez enquanto off-line.
* **Acessível a todos dispositivos clientes:** Não é necessário um servidor para acessar o Firebase Realtime Database, ele pode ser acessado diretamente de um smartphone ou um navegador web. Você pode gerenciar permissões dos usuários, ou seja, definir quem pode acessar/alterar o que, através de regras de seguranças do próprio Firebase. Tudo isso configurável online no Firebase Console (falaremos mais sobre ele).

### Como funciona?

Os dados gerenciados pelo Firebase Realtime Database (FRD) são mantidos localmente através do código do lado do cliente (frontend) utilizando o SDK, e mesmo off-line, você pode utilizar seu aplicativo, proporcionando uma experiência responsiva ao usuário. Quando dados são mantidos em cache, a velocidade das operações são muito mais rápidas do que se tivéssemos que fazer uma requisição HTTP sempre que precisássemos de uma informação, mesmo que esta seja imutável.

O FRD possui algoritmos de checagem e indexação que avaliam se o conteúdo mantido localmente está desatualizado com o conteúdo da núvem, se sim, os dados são baixados e atualizados, se não você usa os mesmos dados mantidos no seu smarphone economizando tempo da requisição e consumo do seu pacote de dados.

Para evitar que usuários mexam em dados que não devem, o FRD fornece uma linguagem de expressões flexíveis para determinar as regras de acesso aos dados, chamadas regras de segurança do Firebase Realtime Database. Estas regras definem como os dados da sua base devem ser estruturados e quando podem ser lidos, gravados ou alterados. Em complemento a isto, temos a integração do Firebase Authentication (que estudaremos mais à frente), que é uma ferramenta de autenticação de usuário que podemos colocar nos nossos apps e definir quem tem acesso a quais dados e como estes dados podem ser acessados, separando níveis de usuários como “Comum, “Administrador”, “Moderador”, “Dono” e etc.

Isto permite que você desenvolva uma ótima experiência em tempo real que pode atender milhões de usuários, por isso é importante definirmos restrições de acesso aos usuários.

## 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 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 o OAuth 2.0 e o OpenID Connect, portanto, pode ser facilmente integrado com seu backend personalizado.

### Principais funcionalidades

Dentre diversas utilidades que o Firebase Authentication tem, iremos utilizar a principal: Autenticação. Com ele temos alguns tipos de autenticação:

* **Autenticação baseada em e-mail e senha:** Integrando o SDK podemos criar uma autenticação clássica baseada em e-mail e senha. As pessoas tem se adequado cada dia mais às autenticações por redes sociais, mas infelizmente a Apple geralmente impede a publicação apps cujo a tela de login não forneça a opção clássica de e-mail e senha.
* **Integração de provedores de identidades federadas:** Autentique usuários ao integrar com provedores de identidades federadas. O SDK Firebase Authentication (ou Auth) oferece métodos que permitem que os usuários façam login com contas do Google, Facebook, Twitter e GitHub.
* **Integração de sistemas de autenticação personalizados:** Além de todas as opções acima, podemos personalizar nosso sistema de autenticação. Caso não queiramos deixar exposto todos os dados de nossos usuários para o Firebase, temos a alternativa de integrá-lo com nosso próprio sistema de autenticação.
* **Autenticação anônima:** Recursos que exigem autenticação para serem consumidos podem ser utilizados de forma anônima com contas temporárias. Se posteriormente o usuário anônimo optar por fazer o login utilizando uma das opções acima, a conta anônima temporária pode ser convertida facilmente para uma conta regular e o usuário poderá continuar de onde parou.

### Como funciona?

Como mencionado acima, existem diversas formas de credenciamento que o usuário pode optar. Estas credenciais podem ser o e-mail e a senha ou um token (código de segurança) OAuth de um provedor personalizado.

Para que o usuário se conecte, ele deve enviar sua credencial de autenticação, em seguida passamos estas para o Firebase Authentication SDK. Os serviços do Firebase verificarão estas credenciais e retornarão a resposta se o usuário tem permissão ou não de acesso a determinada informação.

Para que o usuário tenha uma credencial ele terá que se cadastrar e o Firebase Authentication SDK disponibiliza métodos para isso. Veremos que é bem fácil.

Após um login bem-sucedido o usuário poderá acessar informações básicas do seu próprio perfil, armazenar dados no Realtime Database e ter acesso a outros serviços do Firebase que você implementar no app e que precisem de autenticação de usuário.

## 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 fazer isto então.

### Criando um novo projeto no Firebase Console

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



Figura 1. - Tela inicial do Firebase

1. Faça *login* com sua conta do Google:

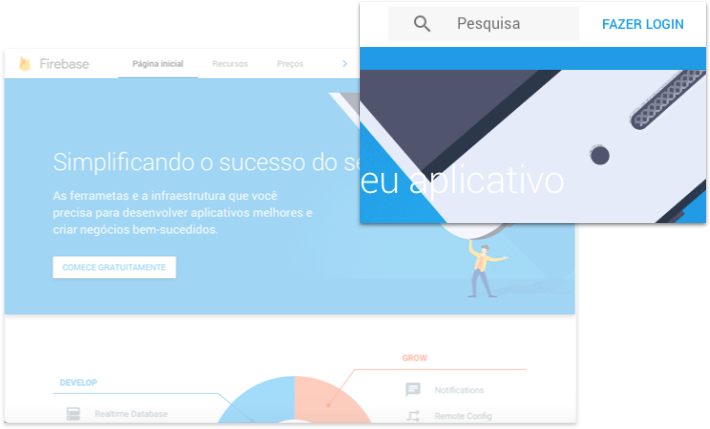


Figura 1. - Localização do botão de Login

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**:



Figura 1. - Localização da opção de criação de um novo projeto

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

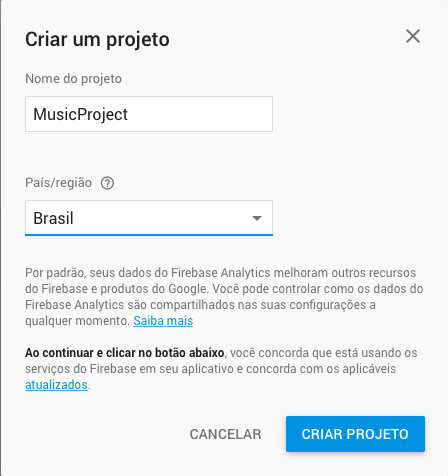


Figura 1. - Criando um novo projeto

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

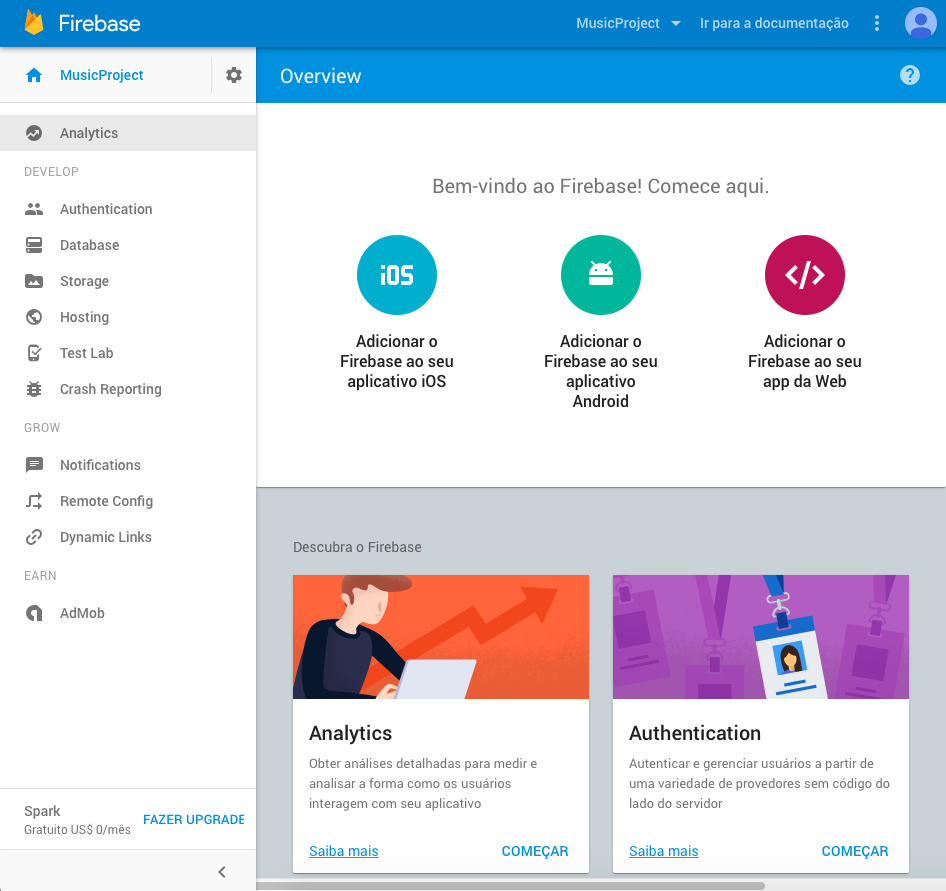


Figura 1. - Tela principal do Firebase Console

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**.



Figura 1. - Localização do botão de adição de 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.

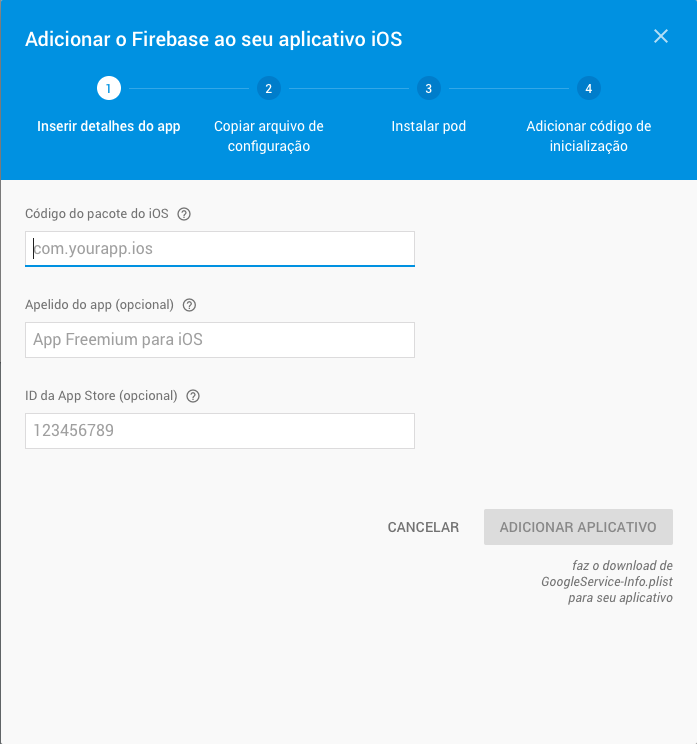


Figura 1. - Inserindo dados necessários para aplicativo iOS

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:

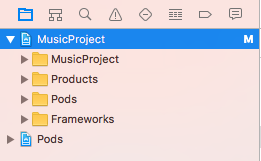


Figura 1. - Localização do arquivo de configuração do projeto Xcode

* 1. O arquivo terá esta aparência:

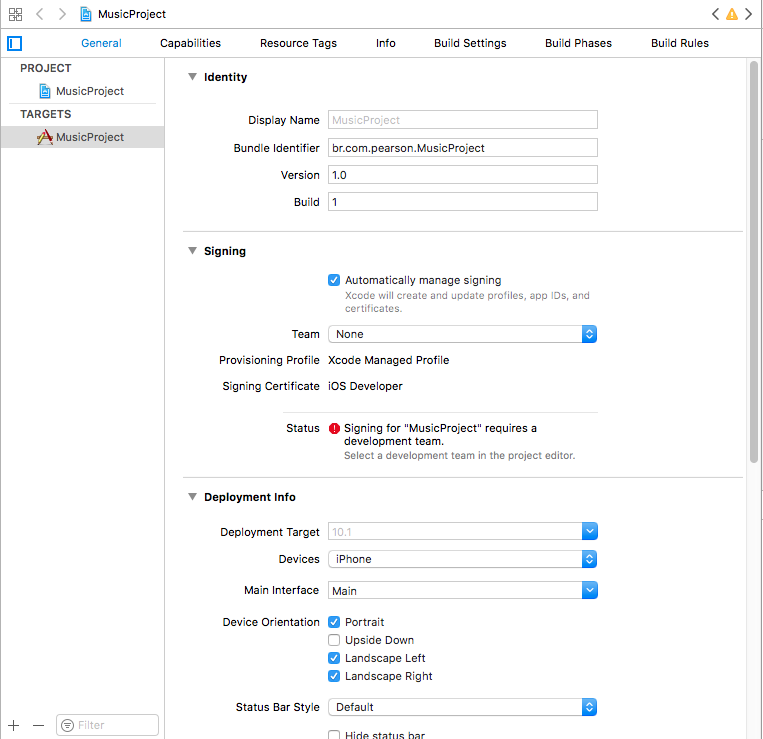


Figura 1. - Tela de configurações do projeto

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

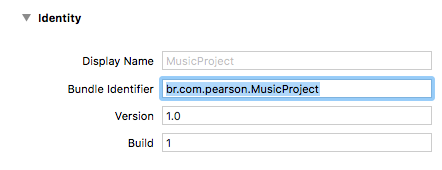


Figura 1. - Obtendo o Bundle ID

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:

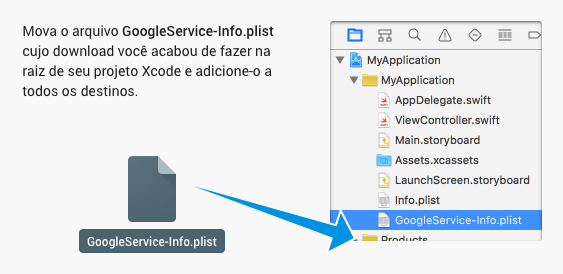


Figura 1. - Adicionando o .plist ao projeto Xcode

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**.

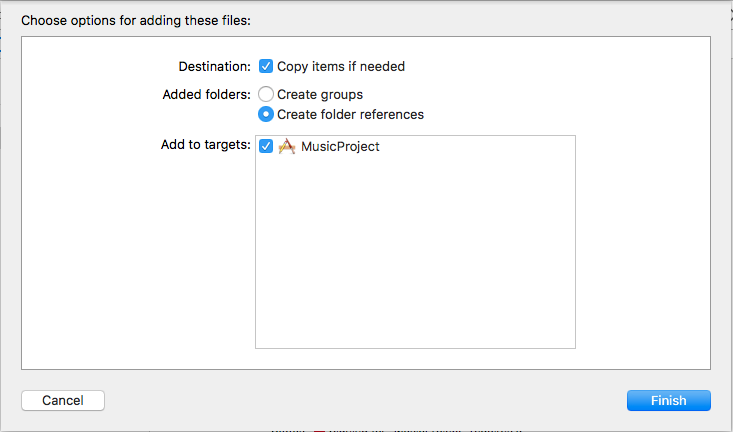


Figura 1. - Garantindo uma cópia do .plist no diretório do projeto

1. O novo arquivo estará adicionado:

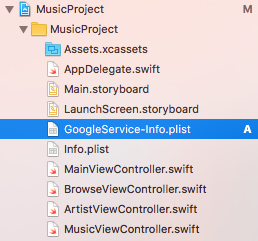


Figura 1. - Localização do arquivo **GoogleService-Info.plist**

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.

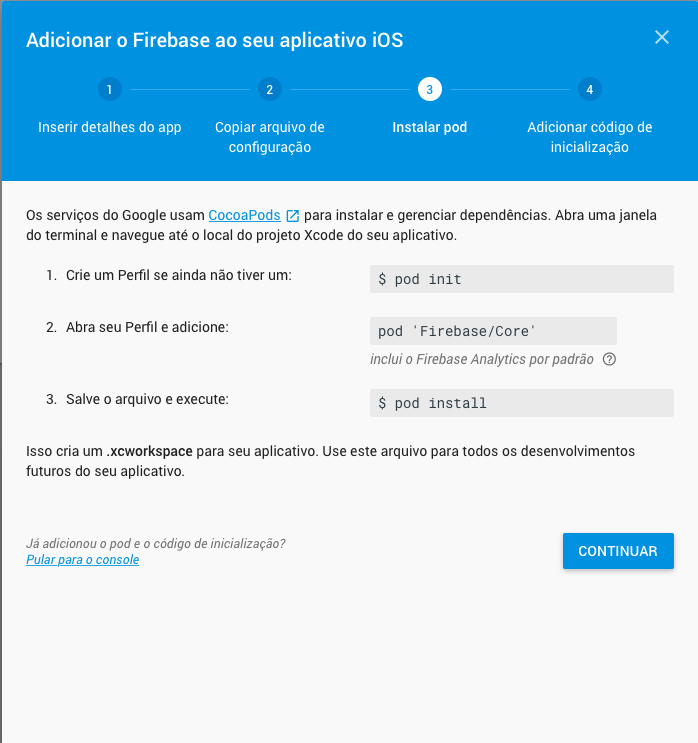


Figura 1. - Passo de instalação do SDK mostrado no Firebase

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."*

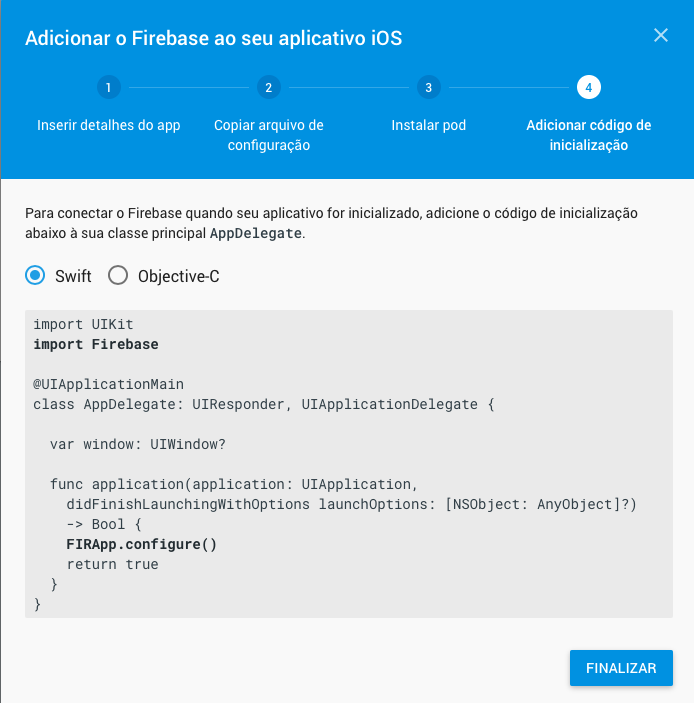


Figura 1. - Ultimo passo para adição do SDK sendo mostrado no Firebase

Com estas configurações estamos prontos para consumir os serviços do Firebase. Como dito anteriormente, 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 *backend* 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 *frontend* quanto do *backend* 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

Depois de um breve resumo na aula anterior iremos agora implementar um sistema de login na nossa aplicação utilizando o Firebase authentication.

## Firebase Authentication

O Firebase Authentication é um sistema que fornece serviços de credenciamento de usuários e um SDK fácil de utilizar. Com ele poderemos oferecer opções ao usuário de autenticação via e-mail e senha, utilizando a conta do Facebook , Google, Twitter e diversas outras. Também com o auxílio dele podemos criar regras de restrições de acesso para determinados tipos de usuários, mantendo a consistência e integridade dos nossos dados.

Um Firebase User representa a conta do usuário que se cadastrou no nosso aplicativo. Geralmente em nossos produtos teremos muitos usuários registrados, mesmo sendo um produto multiplataforma (iOS, Android e web) estes usuários compartilharão do mesmo banco de dados e farão parte do mesmo banco de usuários.

.

### Provedores de login

Uma das coisas que mais desagradam os usuários no uso de aplicativos é quando precisam preencher formulários e cadastros extensos.

Utilizando provedores de login como Facebook e Google pode diminuir a carga de formulários, pois quando um usuário faz login utilizando um destes, informações básicas do perfil dele já são obtidas. O Google por exemplo pode fornecer o Nome, e-mail e a rede de amigos do usuário, que são dados mais do que úteis para possível marketing de outros produtos.

### Usuário atual

Quando um usuário faz login ou se inscreve no seu aplicativo utilizando o Firebase Auth SDK ele fica salvo como estado “Autenticado”. Este estado persiste durante um determinado tempo e assim ele pode continuar utilizando o serviço sem ter que se autenticar novamente sempre ao abrir o aplicativo.

Quando o usuário faz logout o Firebase Auth perde as informações do mesmo, deixando de persistir seus dados. Sendo assim, o usuário precisará passar pelo processo de autenticação novamente.

### Ciclo de vida do usuário

A maneira correta de acompanhar o estado atual do usuário é por meio de **observers** (padrão de projeto aprendido no curso de Android). Um observer é notificado sempre quando algo relevante acontecer com o usuário.

Um observer do Auth é notificado diante das seguintes situações:

* Um usuário renova seu login, pois já autenticou em um momento anterior.
* 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 renovado, ou seja, o usuário precisa de alguma forma se reautenticar. 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 usuário é reautenticado, ou seja, algumas de suas credenciais mudaram e o usuário precisa ser atualizado.

## 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.

}

}

### 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.

}

}

### 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.

### 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 temos 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 e-mail/senha no Firebase console. Siga os passos:

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

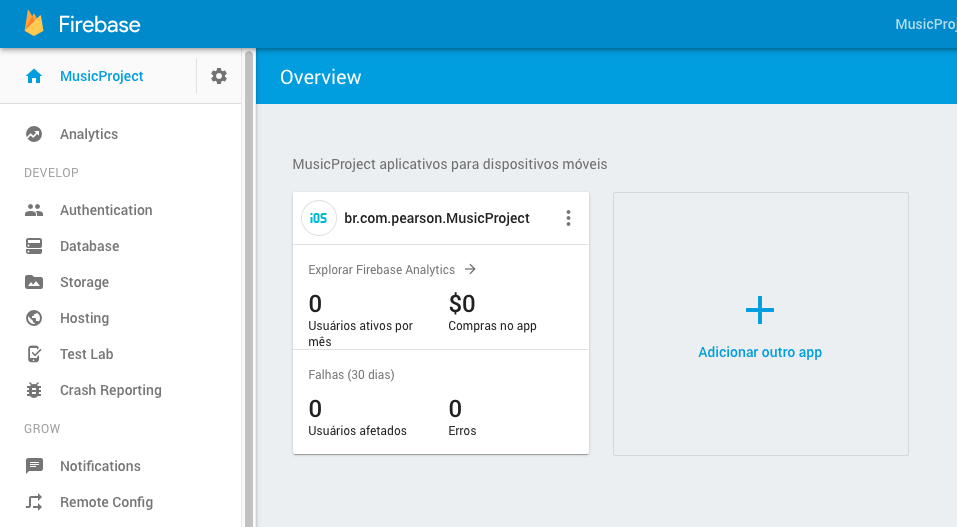


Figura 2.1 - Tela inicial do Firebase Console (com app iOS adicionado sendo mostrado no Overview)

1. Clique na opção Athentication no menu lateral esquerdo.



Figura 2.2 - Localizaçãod da opção Authentication do menu

1. Clique em CONFIGURAR MÉTODO DE LOGIN



Figura 2.3 - Botão Configurar método de login

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

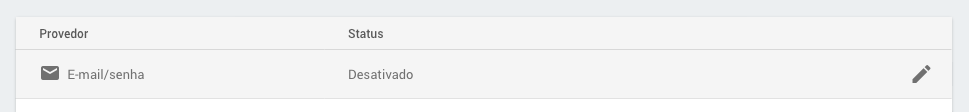


Figura 2.4 - Provedor de e-mail adicionado

1. Ative-o e finalize



Figura .5 - Opção Ativar provedor de e-mail

### 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:

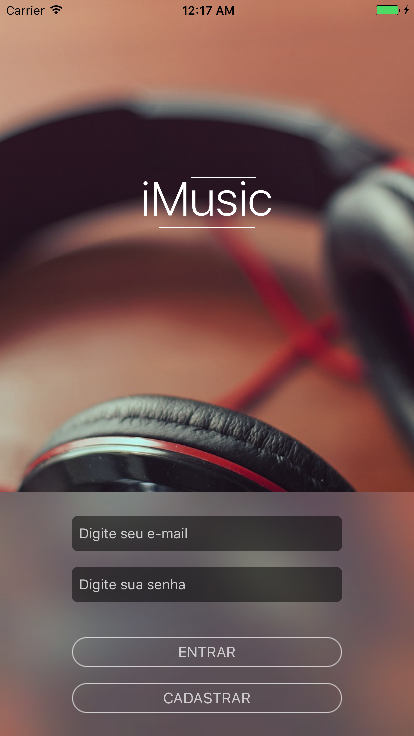


Figura .6 - Tela de login criada

Nela temos o campo *e-mail* e senha. 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 separemos ela em um método. Este método mostra um Alert quando os campos se mostram inválidos (vazios)

1. No método **registerButtonTouchUpInside()** 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:

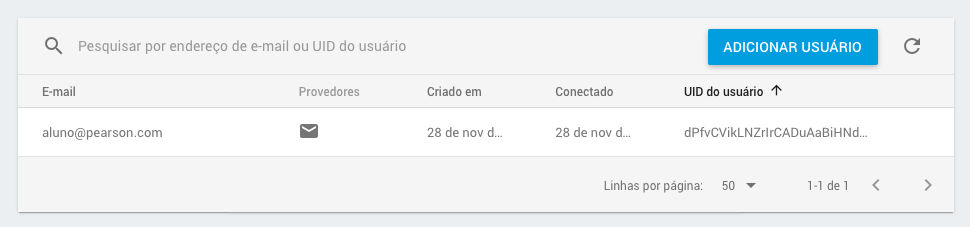


Figura .7 - Firebase mostrando um novo usuário registrado

1. O método de *login* será similar a:

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

if isValid() {

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

if error != nil {

var alertMessage = ""

switch (error as! NSError).code {

case FIRAuthErrorCode.errorCodeWrongPassword.rawValue:

alertMessage = "Senha incorreta."

break

case FIRAuthErrorCode.errorCodeUserDisabled.rawValue:

alertMessage = "Sua conta está desativada. Crie uma nova conta."

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{

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

}

}

}

}

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.

## Resumo

Esta aula aprendemos a criar uma funcionalidade presente na maioria dos aplicativos: o login. Utilizando recursos prontos do Firebase Authentication, esta tarefa se tornou mais fácil do que geralmente seria sem ele. Utilizar login via redes sociais é importante mas muitos ainda utilizam login via e-mail, por questão de privacidade. A Apple também encoraja que seus aplicativos tenham login via e-mail por questão de acessibilidade. Existem países que seu app pode ser lançado que não tem acesso livre ao Google ou Facebook, por exemplo, e o login por e-mail se torna importante neste momento. Nesta aula aprendemos a utilizar o SDK do Firebase Auth para criar um sistema de login básico no nosso app. Além disso aprendemos a configurar o Firebase para tal feito.