Visão Geral

A aplicação tem como objetivo atender toda a cadeia pós-venda da VMI Security, desenvolvida para dispositivos mobiles e web em Ionic, a aplicação é um suporte que possuem chamados, ordem de serviços, relatórios, guias de ajuda, tutoriais, feed, cadastro de peças, tarefas, modelos, equipamentos, justificativas de rejeição, clientes, técnicos, distribuidores. O sistema ainda possui leitura de QR Code para controle de peças, chats e muitas outras funcionalidades. Com linguagem em três idiomas o sistema busca facilitar o contato entre cliente - técnico - distribuidor - vmi.

O desenvolvimento em Ionic, deve-se ao fato de ser uma framework em javascript com possibilidade de desenvolver para múltiplas plataformas, pela sua facilidade de manutenção e excelente documentação.

Configurações de ambiente

Para começar a usar a ferramenta Ionic, o único requisito é uma ambiente Node e NPM.

Claro, um editor de código também é necessário. Visual Studio Code é recomendado.

Terminal

Em geral, é recomendado usar um terminal para builds. A grande maioria dos terminais funcionam com Ionic, mas alguns podem não ser suportados.

Para Windows, **Prompt de Comando** e **PowerShell** são suportados.
Para macOS, o **Terminal** é suportado.

Git Bash (de git-scm.com) **não é suportado** pelo Ionic.

Node & npm

Quase todas as ferramentas utilizadas em projetos JavaScripts são baseadas em Node.js. A página de download possui pacotes de instalação pré-construídos para todas as plataformas. É recomendado a versão LTS para garantir melhor compatibilidade.

Node é fornecido com npm, o pacote de gerenciamento para JavaScript.

Para verificar a instalação, abra um novo terminal e execute:

```
$ node --version
$ npm --version
```

Erros de permissão são comuns no macOS na instalação de pacotes globais com **npm**. Caso possua o **EACCES** erro, veja Resolvendo problemas de permissões.

Git

O sistema de controle de versão Git é bastante recomendado.

O Git geralmente é acompanhado por Host Git, como o GitHub, no caso uma configuração adicional é recomendada. Siga o tutorial para a documentação dos Host Git abaixo:

GitHub: Configure o Git
GitLab: Instalando o Git
Bitbucket: Instalando o Git

Caso o contrário, siga as instruções de instalação oficial. O utilitário de linha de comando pode ser baixado na página de download.

Para verificar a instalação, abra um novo terminal e execute:

```
$ git --version
```

Instalando o Ionic

Os aplicativos Ionic são criados e desenvolvidos principalmente por meio do utilitário de linha de comando Ionic. A CLI Ionic é o método preferido de instalação, pois oferece uma ampla gama de ferramentas de desenvolvimento e opções de ajuda ao longo do caminho. É também a principal ferramenta através da qual o aplicativo é executado e conectado a outros serviços, como o Ionic Appflow.

Instalando o Ionic CLI

Antes de prosseguir, verifique se o seu computador possui o Node.js instalado. Consulte estas instruções para configurar um ambiente para o Ionic.

Instale a CLI Ionic com o npm:

```
$ npm install -g @ionic/cli
```

Se houve uma instalação anterior da CLI Ionic, ela precisará ser desinstalada devido a uma alteração no nome do pacote.

```
$ npm uninstall -g ionic
$ npm install -g @ionic/cli
```

Download do projeto Ionic

Para fazer o download do projeto Ionic, você precisa entrar no terminal e executar o comando Git :

```
$ git clone https://gitlab.com/vmi_mobile/vmi.git
```

Será necessário realizar o login vinculado ao projeto

Alterações do projeto Ionic

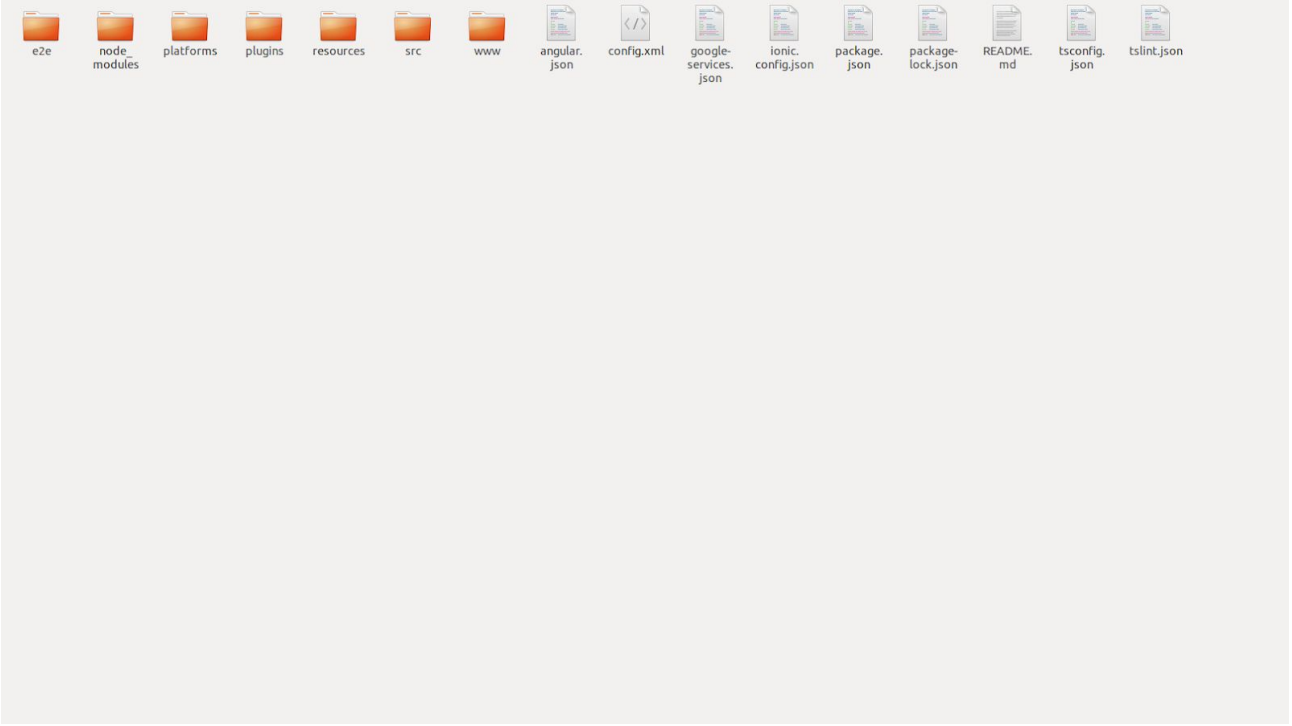
Para instalar os módulos da aplicação, abra um novo terminal no diretório da aplicação e execute:

$ npm install

Para adicionar os dados do servidor da API substitua os dados em **src>environments** e selecione o arquivo **enviroment.prod.ts** (caso o build seja em produção)ou **enviroment.prod.ts** (build local). Substitua `SERVER_URL` pelo atual endereço do servidor da API.

```
export const SERVER_URL = "http://191.190.127.43:2350"
```

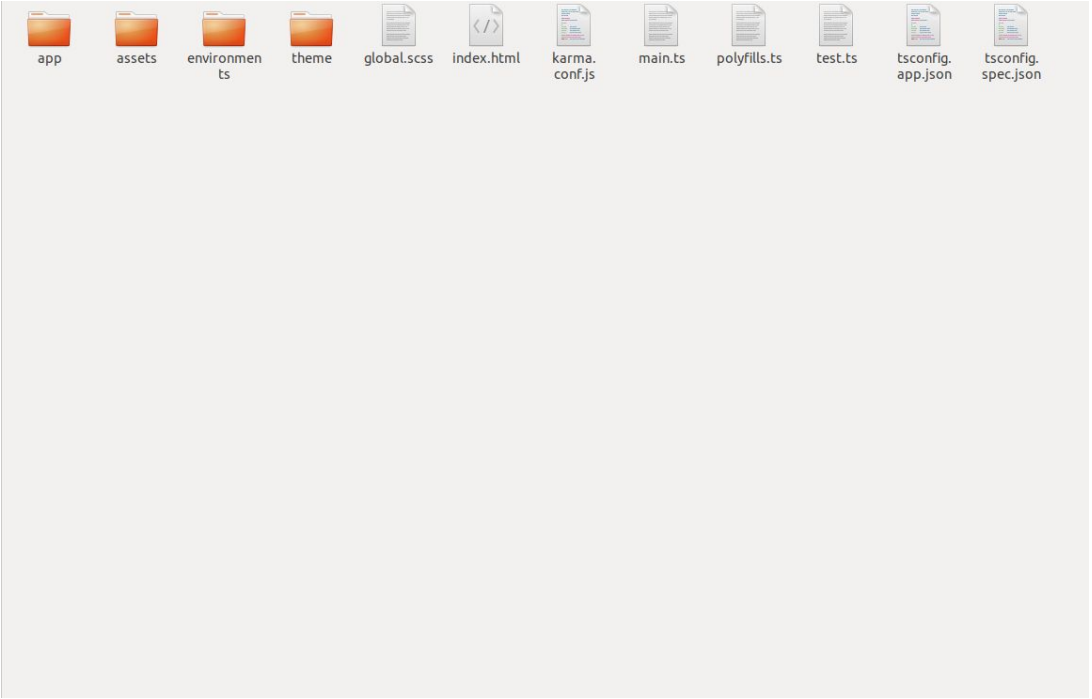
Estrutura do projeto Ionic



A pasta mais importante para o desenvolvimento do projeto é a src, onde está todas as páginas, componentes e serviços da aplicação;

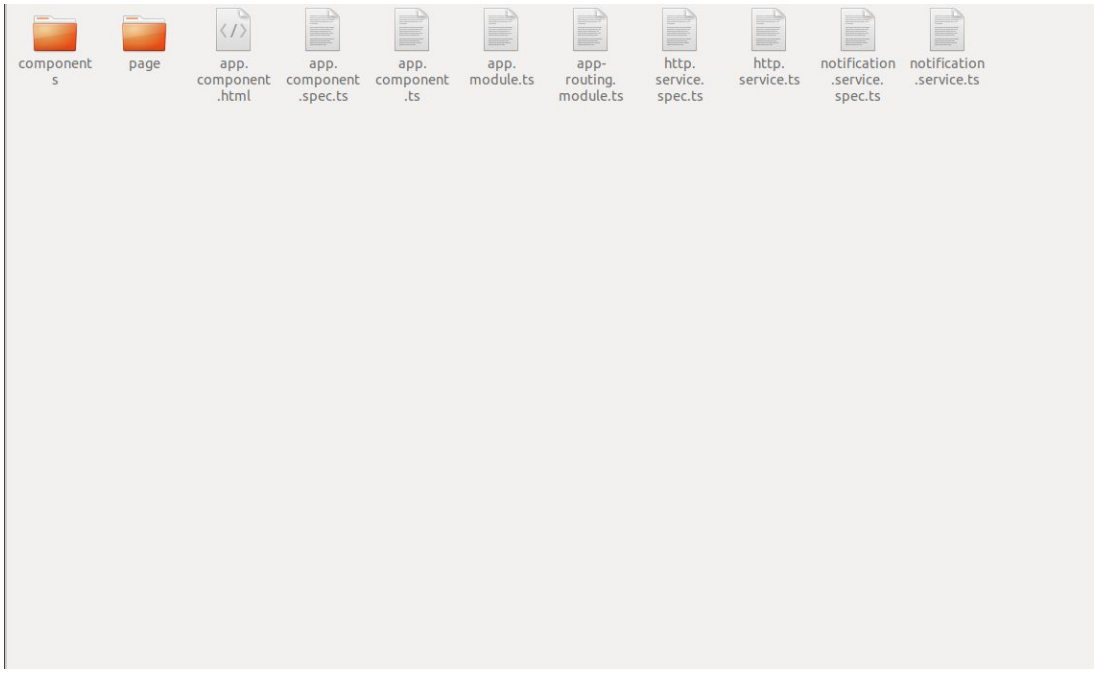
A pasta node_modules carrega todos os módulos utilizados na aplicação (essa pasta só será carregada após a instalação de módulos com o npm install visto na seção anterior);

A pasta platforms é onde estará os builds que você realizou, mais abaixo você irá a ver como é feita a geração dos builds.



Dentro da pasta src você irá encontrar os arquivos da aplicação:

- Na pasta app é onde estará pages, components e services;
- Na pasta assets é o local onde tem ícones, idiomas e imagens utilizadas na aplicação;
- Na pasta environments é onde é feita a configuração com o Server e onde tem todas as variáveis globais;
- Na pasta theme é onde estão os temas utilizados na aplicação;

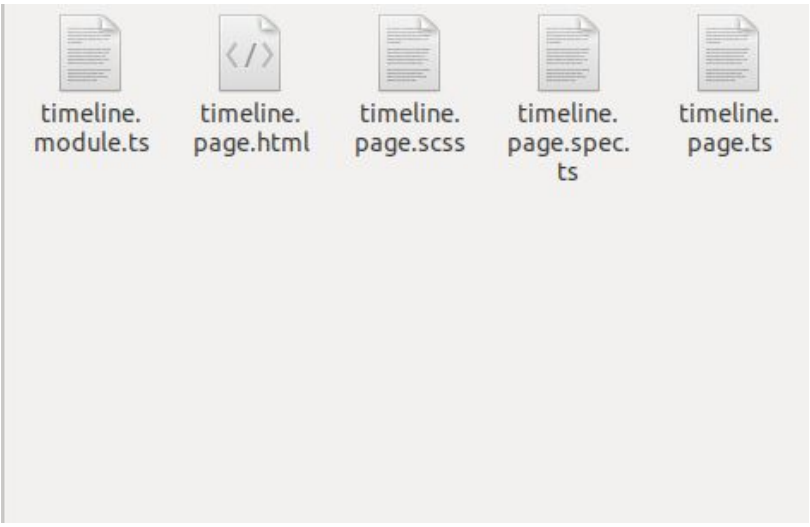


Dentro da pasta **src>app** você vai encontrar toda a distribuição de componentes e páginas da aplicação.

```
async createTimelinePost(title, description, keywords, formData) {

  var root = "";


  this.presentLoading((data) => {

    this.getHeader(),

      this.i = 0,
      this.resultTroubleshooting = [],
      root = "/customer/timeline?title=" + title + "&content=" + description
    var numberpages = [];
    if (keywords != "") {
      for (var i = 0; i < keywords.length; i++) {
        root = root.concat("&keywords=" + keywords[i]);
      }
    }
    this.http.post(SERVER_URL + root, formData, this.httpOptions).subscribe(data => {
      this.completeToast('bottom');
      this.router.navigate(['timeline']);
      this.toastController.dismiss();
    }, error => {

      this.toastController.dismiss();
      if (error.error.msg == "Token has expired") {
        this.refreshToken((j) => {

        });
        this.inativeToast('bottom');
      } else {
        this.requestToast('bottom');
      }
    })

  });
}
```

o arquivo **http.service.ts** é onde estão todas as chamadas realizadas ao servidor, como por exemplo da imagem acima temos uma chamada de criação de um post na timeline(feed) é chamado o carregamento e estabelecido a rota do server para fazer a

requisição, no exemplo acima estamos usando a rota **/customer/timeline** e enviando os dados title,content e keywords para assim a criação de um post na timeline.



Dentro da pasta **src>app>pages>timeline** você vai encontrar um exemplo de como é dividido uma página da aplicação.

- timeline.modules.ts : carrega os módulos utilizado naquela página;
- timeline.page.html : é a parte visual (html) da página;
- timeline.page.csss : é a parte de estilo(css) da página;
- timeline.page.ts: é a programação(typescript) da página;

```
getData(keyword, page, perpage) {

  this.timelineService.getTimeline(keyword, page, perpage, (j) => {


    if (this.result.length == 0) {
      this.result = j.timelines
    } else {
      var count = this.result.length;
      for (var i = 0; i < j.timelines.length; i++) {
        this.result[count + i] = j.timelines[i];
      }
    }

    this.page = j.n_pages
  });

}
```

o arquivo **timeline.page.ts** é onde está toda a programação da página, como por exemplo na imagem acima temos uma função de chamada que busca junto ao **http.service.ts** os posts da timeline para ser exibidos na tela, ele captura os dados da requisição;

Android Development

Android Studio

Android Studio é o IDE para criar aplicativos Android nativos. Ele inclui o Android SDK, que precisará ser configurado para uso na linha de comando.

O Android Studio também é usado para criar dispositivos virtuais Android, necessários para o emulador do Android. Aplicativos ionic também podem ser iniciadas em um dispositivo.

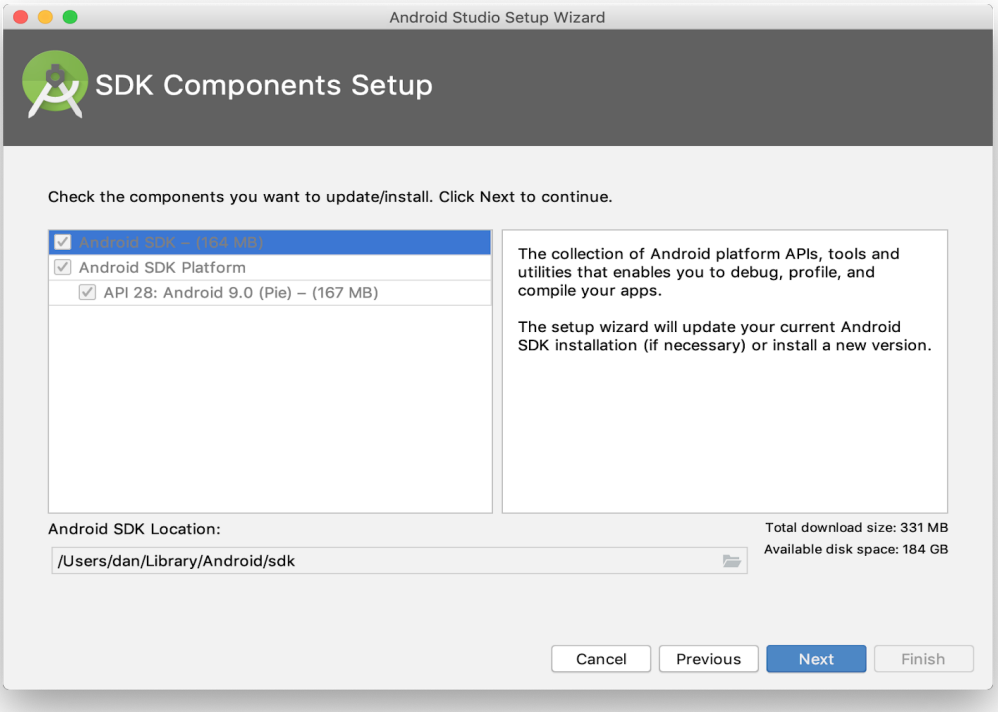
Não recomendamos o uso do Android Studio para o desenvolvimento de aplicativos iônicos. Em vez disso, ele só deve realmente ser usado para criar e executar seus aplicativos para a plataforma Android nativa e gerenciar o SDK do Android e dispositivos virtuais.

Instalando o Android Studio

Baixe o Android Studio no site do Android. Instruções de instalação mais detalhadas podem ser encontradas no Guia do Usuário.

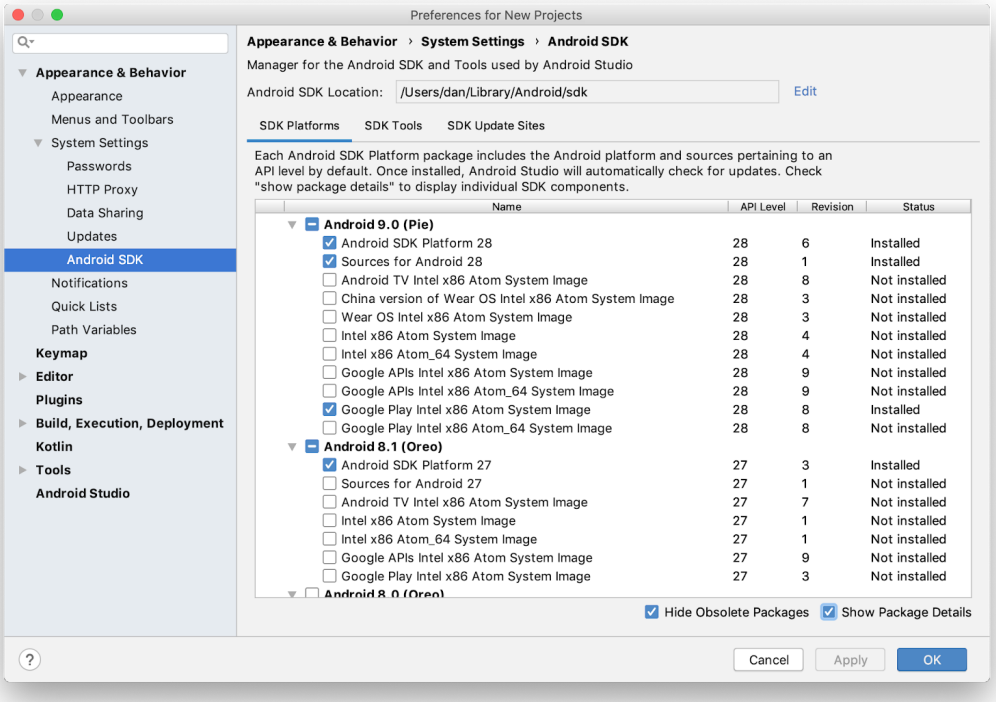
Instalando o Android SDK

Depois de instalado, abra o Android Studio. O IDE deve detectar que o SDK do Android precisa ser instalado. Na tela **Instalação dos componentes do SDK**, termine de instalar o SDK. Anote o local do **SDK do Android**.



Por padrão, a Plataforma SDK estável mais recente é instalada, o que inclui uma coleção de pacotes necessários para atingir essa versão do Android.

Para instalar imagens do sistema e outros pacotes menores da plataforma SDK, pode ser necessário garantir que a opção **Mostrar detalhes do pacote** esteja marcada na parte inferior do SDK Manager.



Para referência futura, o Android SDK pode ser gerenciado com o Android Studio no menu **Configure** » **SDK Manager** da tela de boas-vindas do Android Studio ou **Tools** » **SDK Manager** em projetos do Android.

Configurando ferramentas de linhas de comando

O SDK do Android é fornecido com ferramentas úteis de linha de comando. Antes de poderem ser usadas, algumas variáveis de ambiente devem ser definidas. As instruções a seguir são para macOS e Linux. No Windows, verifique a documentação sobre configuração e persistência de variáveis de ambiente nas sessões do terminal.

No **~/.bashrc**, **~/.bash_profile**, ou scripts de inicialização do shell semelhantes, faça as seguintes modificações:

1. Defina **ANDROID_SDK_ROOT** variável de ambiente. Esse caminho deve ser o **local do Android SDK** usado na seção anterior.

   ```
   $ export ANDROID_SDK_ROOT=$HOME/Library/Android/sdk
   ```

2. Adicione os diretórios da linha de comando do Android SDK ao **PATH**. Cada diretório corresponde à categoria da ferramenta de linha de comando.

> Para **apksigner** e **zipalign, $ ANDROID_SDK_ROOT / build-tools** também devem ser adicionados ao **PATH**.

Cordova Configuração

É necessária uma configuração adicional para o Cordova suportar compilações programáticas.

Java

Aplicativos Android nativos são compilados com a linguagem de programação Java . Faça o download do JDK8 na página de download.

Infelizmente, o Cordova não é compatível com a versão mais recente do Java.

Gradle

Gradle é a ferramenta de compilação usada em aplicativos Android e deve ser instalada separadamente. Veja a página de instalação para detalhes.

iOS Development

Este guia descreve como executar e depurar aplicativos Ionic em simuladores e dispositivos iOS usando o Cordova. Os aplicativos iOS só podem ser desenvolvidos no macOS com o Xcode instalado.

Existem dois fluxos de trabalho para executar aplicativos Ionic no iOS:

- Executando com o Xcode

- Executando com a CLI Ionic

A abordagem do Xcode é geralmente mais estável, mas a abordagem da CLI Ionic oferece a funcionalidade de recarga ao vivo.

Configurando Xcode

Xcode é o IDE para criar aplicativos iOS nativos. Inclui as ferramentas de linha de comando do iOS SDK e Xcode. O Xcode pode ser baixado gratuitamente com uma conta Apple ou pode ser instalado através da App Store.

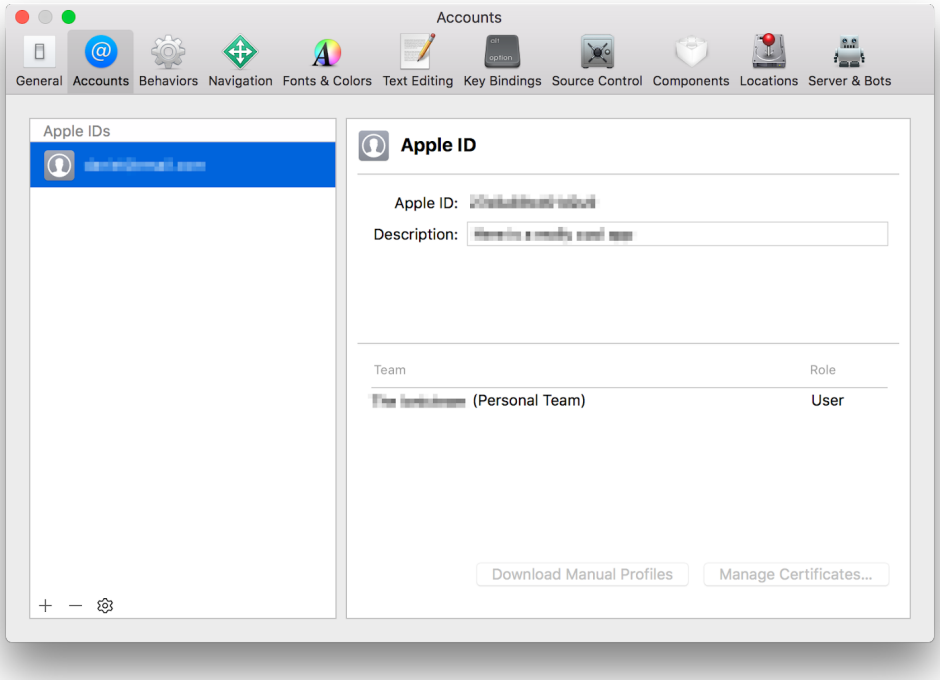
Depois que o Xcode estiver instalado, verifique se as ferramentas de linha de comando estão selecionadas para uso:

```
$ xcode-select --install
```

Configurando uma equipe de desenvolvimento

Todos os aplicativos iOS devem ter código assinado, mesmo para desenvolvimento. Felizmente, o Xcode facilita isso com a assinatura automática de código. O único pré-requisito é um ID da Apple.

Abra o Xcode e navegue até **Xcode** » **Preferences** » **Accounts**. Adicione um ID Apple, se nenhum estiver listado. Uma vez logado, uma Equipe Pessoal aparecerá na lista de equipes do Apple ID.



Build da Aplicação

Para adicionar uma plataforma a aplicação, abra um novo terminal no diretório da aplicação e execute:

$ ionic cordova platform add (platform)

Substituir platform por android/browser/ios

IMPORTANTE!

Após adicionar a plataforma **android** vá ao diretório onde está o projeto depois **platforms>android>project.properties** e substitua a versão da cordova.system.library.5

e substitua por:

```
cordova.system.library.5=com.google.firebase:firebase-messaging:20.+
```

Feito isso para buildar a aplicação, abra o terminal no diretório da aplicação e execute:

$ ionic cordova run (platform) —prod

Onde --**prod** é o parâmetro para build em produção. Substituir platform por android/browser/ios

Se tudo ocorrer bem, o apk da aplicação será encontrado em :

platforms>android>app>build>outputs>apk