ARTIGO: 11756
MOBLOJA setup nova loja

Nesse artigo descreveremos como fazer o setup de uma nova loja. Para fins didáticos, vamos chamar a nova loja de **Droga Zé.**

Portal WEB

1. No menu **Adminstrativo** > **Empresas** crie a nova empresa.
2. No menu **Administrativo** > **Usuários** crie o usuário de gerenciamento com permissão admin vinculado a nova empresa.
3. Dentro do cadastro de empresa, crie o usuário "**Usuário API loja web**" clicando no respectivo botão.
4. Você ainda pode configurar os dados da integração com a HOS preenchendo os dados da integração.
5. Também é importante configurar o **URL key** para ser usado no site.
6. Revise as configurações junto ao cliente

Aplicativo

Para iniciar, na pasta do projeto **mobloje/mobile/app-templates** copie as configurações (pasta) de uma loja existente criando uma pasta com o nome **drograze.**

Customização de imagens de ícones

Dentro da pasta da nova loja, precisamos customizar os ícones usado no app, os ícones usados pelo Android e iOS e as imagens usadas no Splash do app:

Icones usados no app: Customizar as imagens presentes na pasta **app-res** e na pasta das respectivas plataformas de acordo com as cores da nova loja.Para iOS ficam em **drogaze/ios/assets** a para Android ficam em **drogaze/android/assets**

Icones usados pelas plataforma: Criar os ícones que serão usados no Android e iOS. Para criação dos ícones, acesse https://icons.mobilemind.com.br e envie uma imagem do ícone na opção **Laucher Icons**. Essa imagem deve ter 1**024x1024** e não pode ter fundo transparente. O site vai gerar um zip contendo todos os ícones do Android e iOS que deve ser guaraddos na pasta das suas respectivas plataformas, a saber: iOS pasta **drogaze/ios/assets/Assets.xcassets/AppIcon.appiconset** e Android **drogaze/android/assets**.

Imagens de Splash: Customizar as imagens do Splash do app. iOS em **drogaze/ios/assets/Assets.xcassets** pastas **LaunchScreen.AspectFill.imageset** e **LaunchScreen.Center.imageset**, no Android na pasta **drogaze/android/assets**

Ajuste das configurações

Na pasta da nova loja, edite o arquivo chamado **configs.json**. Você precisa alterar todas as referências do modelo copiado para a nova loja, ou seja, em todos os lugares onde aparece o nome da loja antiga, informe o nome da nova loja.

- O appId segue um padrão: br.com.mobloja.. Par anosso exemplo ficaria **br.com.mobloja.drogaze**
- As cores hexadecimais informadas nos campos **ns*, nav* e notify*** seguem as principais cores da nova loja. Essas cores são usadas no tema padrão do app para cada plataforma, pintando a **NavBar**, fundo de notificação, etc..
- O campos **tenantId** deve ser preenhido com o **UUID** da loja no portal web.
- Os campos **ios*** são preenchidos com as informações obtidas através da conta da apple.
- Os campos **androidKey*** indicam onde estão as informações do certificado de produção do Android.
- O campos **senderId** é o número do projeto do **Firebase**
- O campo **mapsApiKey** é a chave gerada no projeto do google descrito mais adiante. Ele se refere a chave Browser Key exibido nas credenciais do projeto do Google.
- O campo **mapsSdkApiKey** é a chave gerada no projeto do google descrito mais adiante. Ele se refere a chave Android Key exibido nas credenciais do projeto do Google.
- O campos **tenants** é usando quando o app atente mais que uma empresa, com isso é exibido uma tela de seleção de empresa antes do login.

Firebase

Crie um app firebase para notificações: https://console.firebase.google.com. Na criação será solicitado o **appId** e a impressão digital do app.

Impressão digital (finger print)

Precisaremos gerar duas impressões digitais, uma para o certificado de teste para podermos usar as funcionalidades em ambiente de desenvolvimento e uma para produção. A digital de teste é gerada usando as chaves padrões do android (que é única para cada instalação). O comando abaixo gera a impressão pra versão de desenvolviment, a senha solicitada é **android.**

```
keytool -list -v -alias androiddebugkey -keystore ~/.android/debug.keystore
```

Copie e SHA1 exibido e informe na criação da conta no Firebase. Nesse momento você só consegue informar uma impressão, mas nas configurações do projeto você pode adicionar mais impressões. Siga os passos de criação do projeto. Quando terminado, gere a impressão de produção.

Para gerar a impressão de produção vamos precisar criar o **keystore** de produção do cliente. Esse **keystore** deve ser guardado no Dropbox, na pasta **certificados/google/**. Para facilitar, abra um terminal, navegue até a pasta e gere o **keystore** dentro da pasta da nova loja:

```
cd ~/home/Dropbox/certificados/google/drogaze
```

```
keytool -genkey -alias drogaze -keyalg RSA -keystore drogaze.keystore -keysize 2048 --validity 100000
```

Serão solicitadas várias informações de quem está gerando o certificado. Use uma senha segura com mais de 10 caracteres e salve os dados da senha/alias dentro da mesma pasta em um arquivo chamado **info.txt** para registro.

Após gerado com sucesso será gerado um arquivo **drogaze.keystore** dentro da pasta, que será usado para geração do APK de produção.

Feito isso, podemos gerar a impressão de produção. (use a mesma senha que você criou anteriormente)

```
keytool -list -v -alias drogaze -keystore ~/home/Dropbox/certificados/google/drogaze/drogaze.keystore
```

Copia o SHA1 e no console do Firebase acesse no ícone da engrenagem as **Configurações do projeto**. Na aba **Geral** você vai ter os dados do projeto (copie o Número do projeto para o **configs.json**) e um aplicativo android. Vá na opção "**Adicionar impressão digital**" para adicionar a impressão de produção.

Adicionando Authentication (login do google)

Na barra lateral do console do Firebase acesse "**Todos os produtos**" e selecione **Autenticação**. Selecione o **Método de login Google.** Com isso o app pode usar o login do Google.

O valor gerado em **"Configuração do SDK da Web"** > **"Id do cliente da Web"** deve ser copiado para a configuração **" ConfigGoogleClientWebId"**

Adicionando Cloud Messaging (notificações do android)

Na barra lateral do console do Firebase acesse "**Todos os produtos**" e selecione **Cloud Messaging**.

Por fim, acessando as configurações, na aba Geral > Projetos faça o download do arquivo **googe-services.json** e coloque na pasta do projeto **drogaze/android** substituindo o atual.

Configurando as APIs do Google

Quando criamos um projeto no Firebase, automaticamente é criado um projeto no google developers.

Api de localização

Essa função habilita ao usuário pesquisar um endereço através da sua **GEO Localização**.

Vá para o painel de desenvolvedor do Google: https://console.developers.google.com/apis e selecione o projeto da nova loja. Vá em **Bibliotecas**, pesquise por **Geocoding API** e ative.

Agora vá em **Credenciais** e observe que foram criadas duas chaves de API: **Android key** e **Browser key.** Copie essas chaves para o arquivo **configs.json** da loja loja nas respectivas chaves:

- mapsApiKey - Browser key
- mapsSdkApiKey - Android key

Arquivo de credenciais para notificações

Ainda em **Credenciais**, em "**Contas de serviço**" precisamos gerar a chave responsável pelo acesso ao servido de envio de notificações por parte da nossa API. Na listagem clique no item "**firebase-adminsdk-***". Na nova tela, vá na aba **Chaves** na opção **Adicionar Chave** > **Criar nova chave** > selecione **JSON** e faça o download do arquivo. Esse arquivo deve ser configurado e enviado para o servidor **pushd**.

E aqui termina as configurações no console do Googel Cloud.

Customização de cores do app/desenvolvimento

Se você chegou até aqui, você criou tudo que é necessário para podermos iniciar o trabalho de customização de cores do app. Então você pode executar

```
$ ./docker prepare android drogaze
```

Agora você pode iniciar o app e trabalhar na customização de cores.

```
$ ./docker run android
```

IMPORTANTE: Todas as vezes que você executa o comando acima, algumas informações do app são substituídas pela loja destino. Qualquer mudança feita nos seguintes itens devem ser **replicadas** para o template da loja para que não haja perda de dados:

- Imagens/Ícones
- _variables.sass

O arquivo **_variables.sass** guarda todas informações de cores do app. Durante o processo de customização, você vai editar ele em **mobile/vue/src/theme/_variables.sass.** Assim que a customização estiver concluída, você deve copiar o conteúdo desse arquivo para o template da loja (**app-templates/<loja>/styles/_variables.sass**).

Qualquer variável adicionadas nesse arquivo deve ser replicada, **OBRIGATORIAMENTE**, para todas outras lojas em seus respectivos arquivos.

Agora basta você customizar as cores do app e, ao final, replicar as cores para o template da nova loja.

Informações pertinentes a publicação do app no Google Play

Para usar saber o sha do certificado usado pelo Gooogle Play acesse "Testar e lançar" > "Integridade do app". Na opção "Assinatura de apps do Google Play" acesse "Configurações". Então você encontrará "Certificado da chave de assinatura do app"

Ou você pode gerar o sha fazendo o download do apk assinado, como explicado abaixo:

Note: When using Play App Signing, the upload key certificate will be different than the app signing key certificate.

Using Keytool on an APK or AAB To get the certificate of an application binary:

APK file

```
keytool -printcert -jarfile app.apk
```

AAB file

```
keytool -printcert -jarfile app.aab
```