

Rota Show

- Equipe:
 - Caliny Basílio de Oliveira
 - Christopher de Almeida Martins
 - Daniel Lopes Bueno Lobato
 - Douglas Gonçalves Sales
 - Pedro Henrique Figueiredo Salim
 - Henrique de Lima Barroso
- Sobre
 - Tecnologias:

Para o desenvolvimento do projeto, utilizamos as seguintes tecnologias:

 - Python/Django/Django Rest Framework no Backend.
 - Sqlite3 como banco de dados.
 - Javascript/React Native/Redux, utilizando o expo-cli no Frontend (aplicativo).
 - Estrutura de Pastas do projeto
 - accounts:

Contêm as definições e lógicas do sistema de autenticação e registro de usuários.
 - app:

Contêm os arquivos de definição do aplicativo desenvolvido com a utilização do expo. Mais informações a seguir.

 - assets:

Arquivos e imagens utilizadas no aplicativo.
 - components:

Componentes React do sistema. Importados nas telas.
 - context:

Contexto de Autenticação do sistema. Utilizamos este contexto para armazenar o token do usuário e realizar chamadas a apis.
 - routes:

Arquivos que separam as rotas privadas (que requerem autenticação) e públicas. Registramos as telas que podem ser acessadas nestes arquivos.
 - screens:

Contêm cada uma das telas do aplicativo.
 - App.js:

Arquivo raiz do aplicativo.
 - reducers.js:

Arquivo de registro de alteradores de estado do redux. Só utilizamos um único reducer (Home) no aplicativo.
 - store.js:

Configuração de armazenamento de estado do aplicativo.

- variables.js:
Definições de variáveis de ambiente (DESENVOLVIMENTO, TESTE ou PRODUÇÃO) do aplicativo.
- img:
Imagens utilizadas nesta documentação.
- legacy:
Contêm as definições e lógicas do sistema de geração de rotas.
- meuapp:
Contêm os arquivos de configuração e raiz do Sistema.
- dbsqlite:
Banco de dados Sqlite.
- templates:
Arquivos de template html.
- insomnia_export.json:
Biblioteca com exemplos e testes das APIS do sistema. Pode ser importado no Insomnia para mais informações.
- manage.py:
Script do Django, pelo qual executamos comandos no Backend.
- requirements.txt:
Listagem de pacotes e versões requeridas pelo sistema.

- Considerações

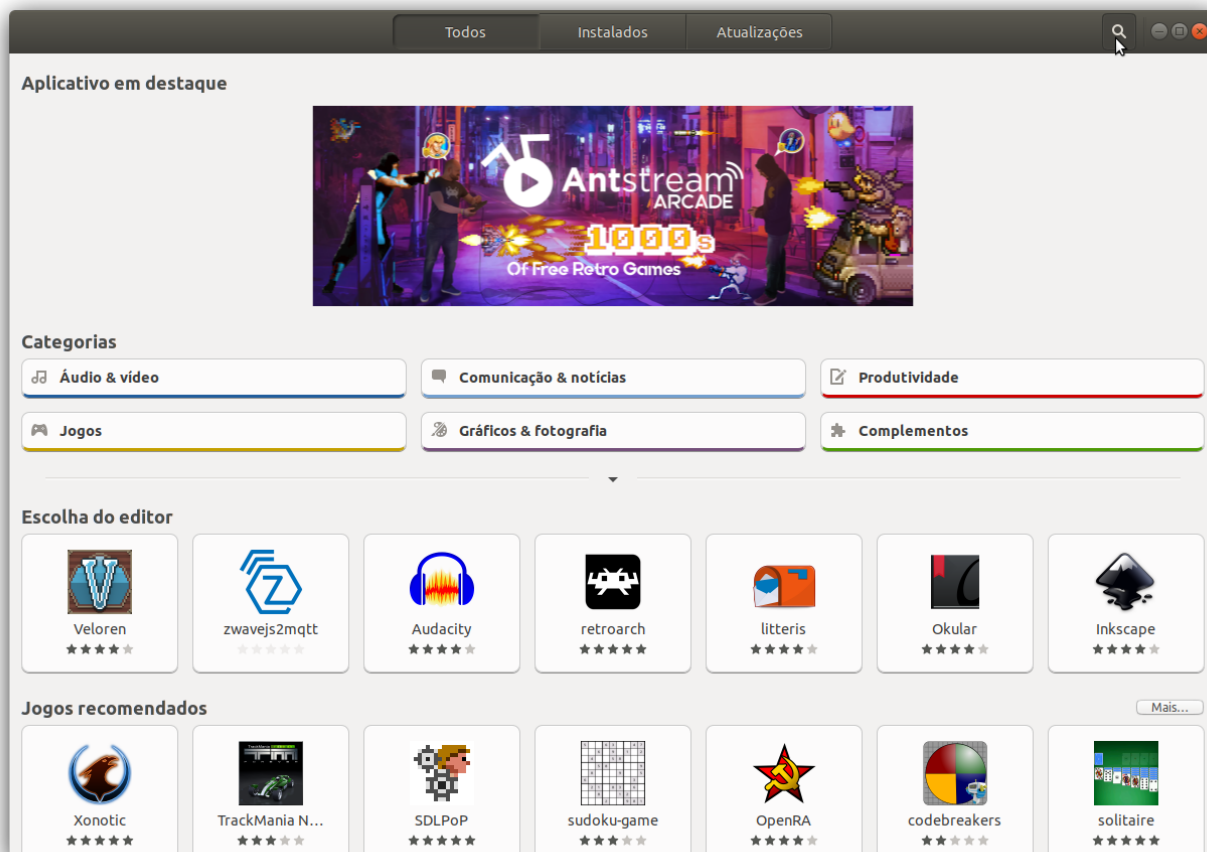
1. Instalação

Backend

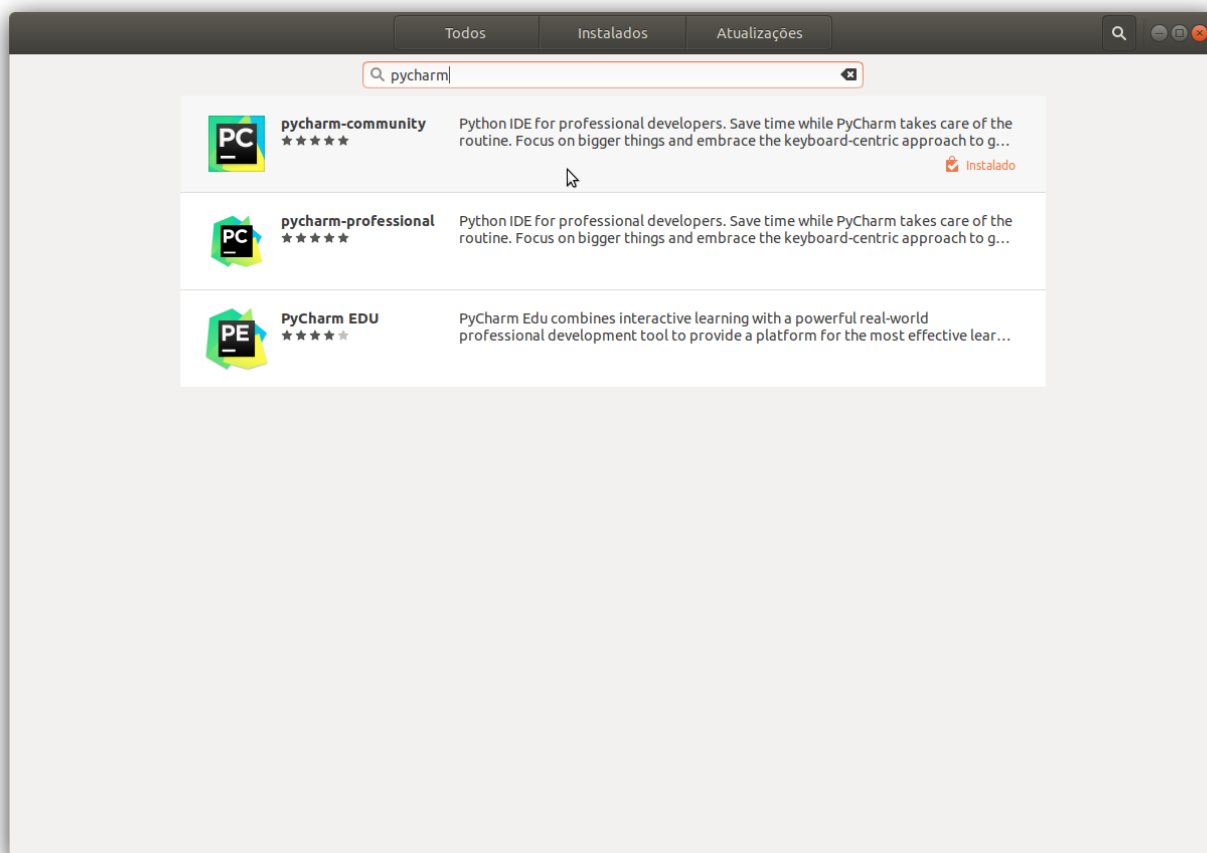
- Python 3 e pip (no terminal digite)

```
sudo apt-get install python3.6 python3-pip
```

- Pycharm Community Edition
Abra o Software Manager e utilize o botão de pesquisa para encontrar o pacote do Pycharm Community Edition.
Clique na Lupa e pesquise por pycharm.



Selecione pycharm-community.

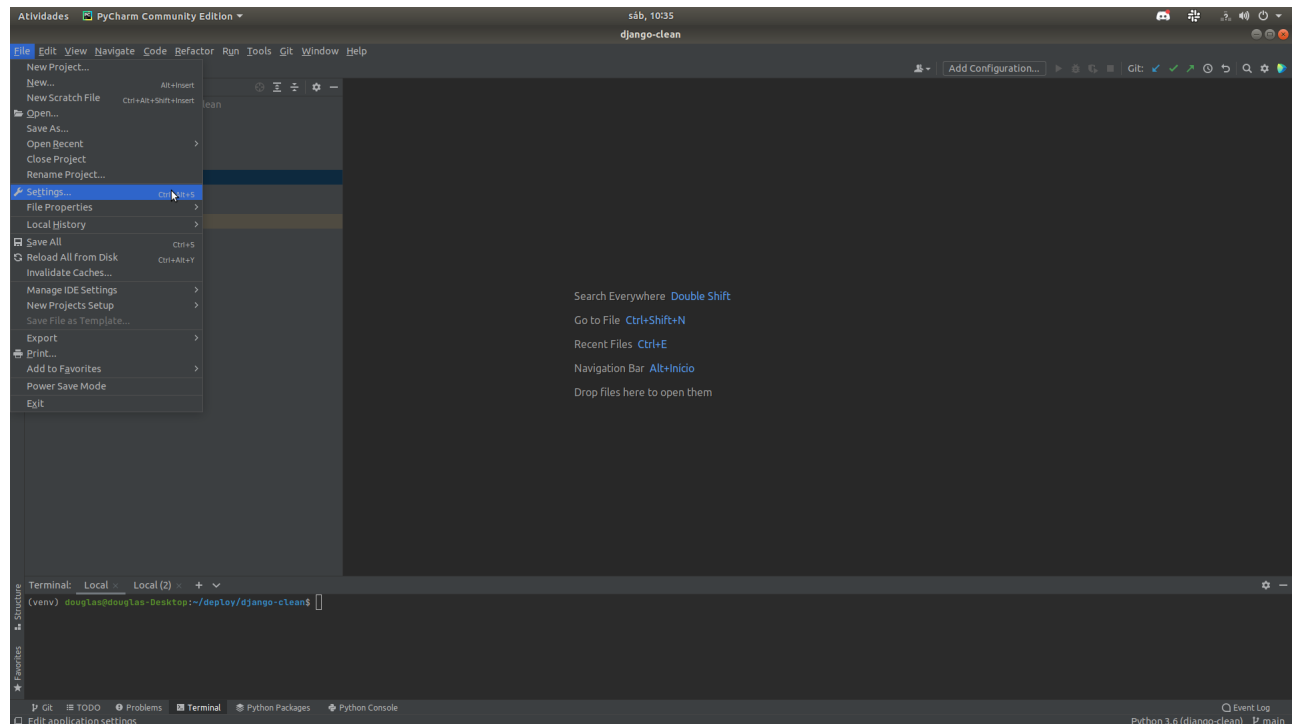


Clique no botão Instalar e aguarde a finalização.

- Criar Ambiente Virtual

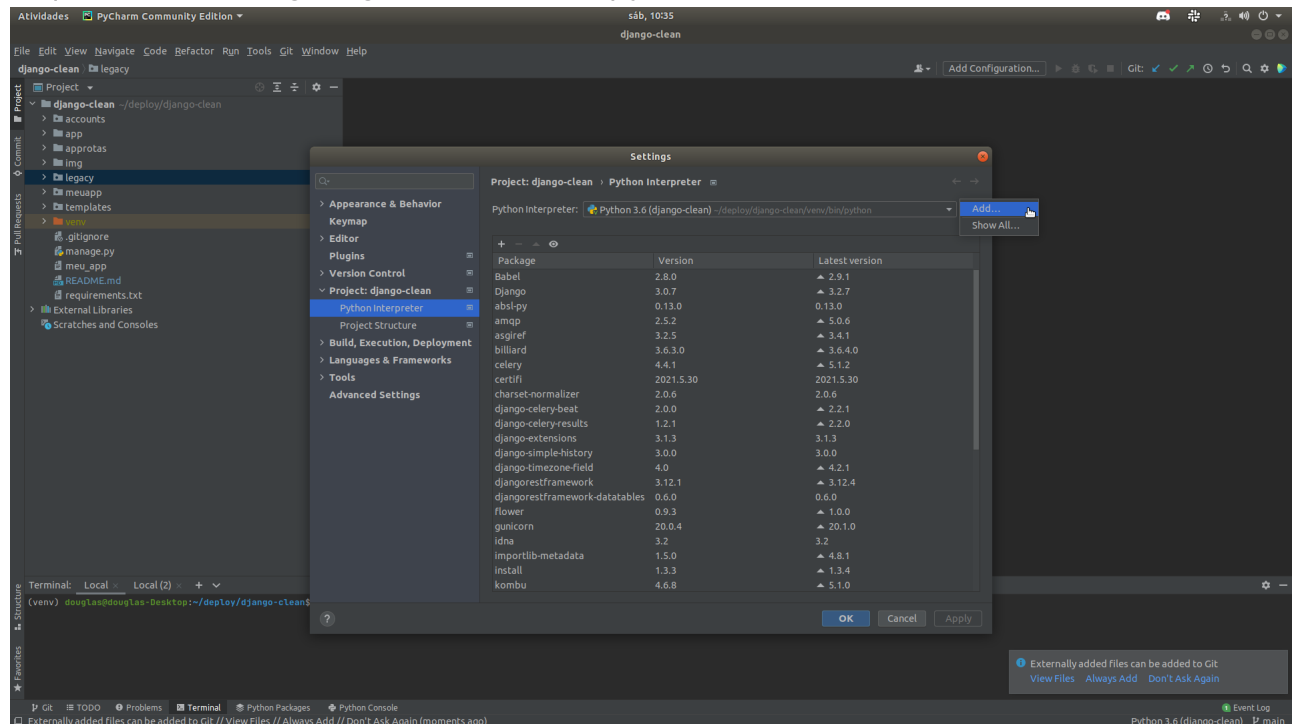
Abra o Pycharm, clique em Open e selecione a pasta do projeto.

Abra o menu FILE > SETTINGS.



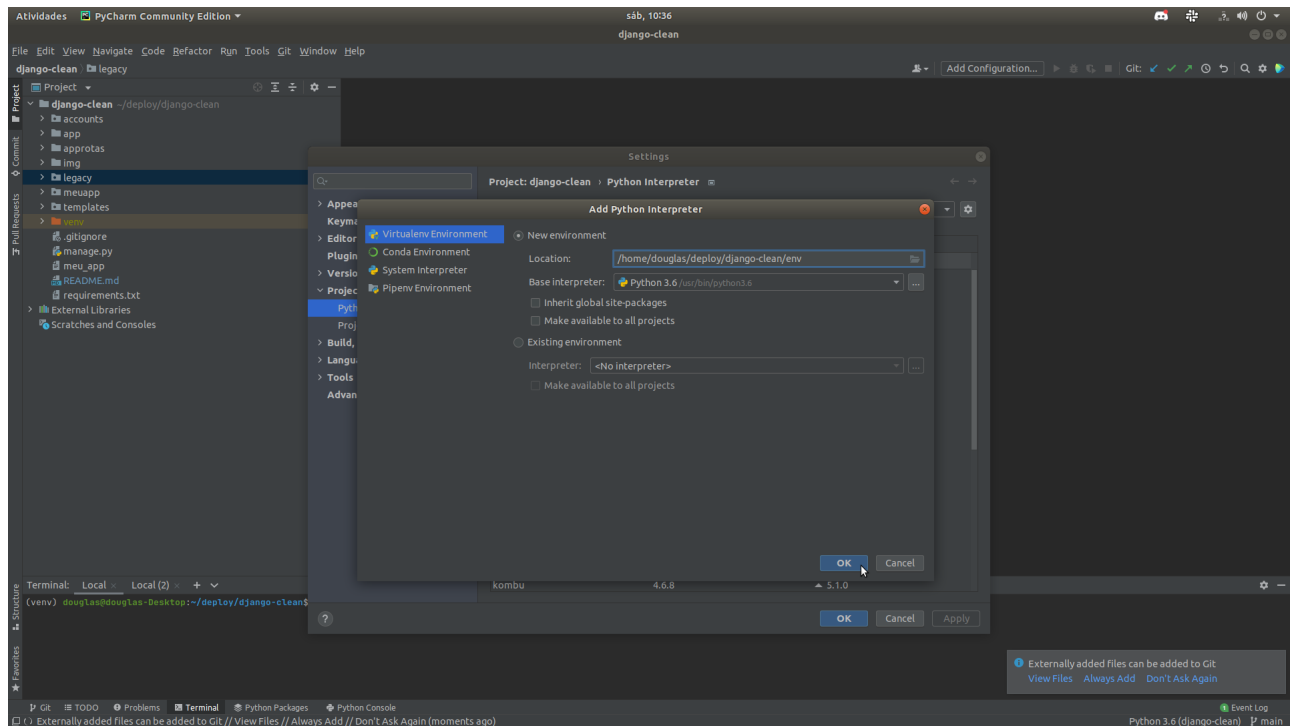
Navegue para Project > Python Interpreter.

Clique no botão da engrenagem e selecione a opção add.

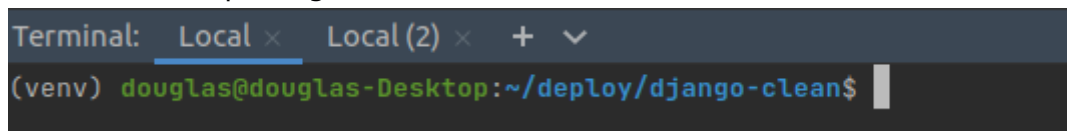


Selecione Python 3.6 no Campo Base Interpreter e Clique em OK.

Feche a janela de configurações do Pycharm.



Na parte inferior do Pycharm, clique no botão Terminal e verifique se o seu ambiente virtual está ativado antes de prosseguir.



No início do seu terminal deverá haver uma mensagem (venv), caso não haja, feche o projeto e abra-o novamente ou digite o seguinte comando no terminal:

```
. venv/bin/activate
```

- Instalar Dependências do Projeto.

No terminal do Pycharm, digite:

```
pip install -r requirements.txt
```

- Executar Migrações.

No terminal do Pycharm, digite:

```
python manage.py migrate
```

- Criar Super Usuário.

No terminal do Pycharm, digite:

```
python manage.py createsuperuser
```

- Executar o Servidor.

No terminal do Pycharm, digite:

```
python manage.py runserver 0.0.0.0:8000
```

Frontend

- Instalar Curl

No terminal do Linux, digite:

```
sudo apt-get install curl
```

- Instalar Node.js 16.9.1.

No terminal do Linux, digite:

```
cd ~  
curl -o- https://raw.githubusercontent.com/nvm-  
sh/nvm/v0.35.3/install.sh | bash  
source ~/.bashrc  
nvm install v16.9.1
```

- Instalar yarn.

No terminal do Linux, digite:

```
sudo npm install --global yarn
```

- Instalar Expo.

No terminal do Linux, digite:

```
sudo yarn global add expo-cli
```

- Instalar Dependências do Projeto.

No terminal do Linux, navegue para a raiz do projeto e digite os comandos a seguir:

```
cd app  
yarn install
```

- Configurar Variáveis.

Precisamos informar ao aplicativo o nosso número de ip para que possamos acessar o servidor de backend no ambiente de desenvolvimento. Para isso, descubra seu endereço de ip utilizando o comando ifconfig ou através de outros meios que preferir, e o digite dentro do arquivo variables.js, da seguinte forma:

```
app > JS variables.js > ...  
1 | export const BASE_URL = "http://192.168.100.32:8000/api" //MEU IP NA LAN
```

- Rodar o Projeto.

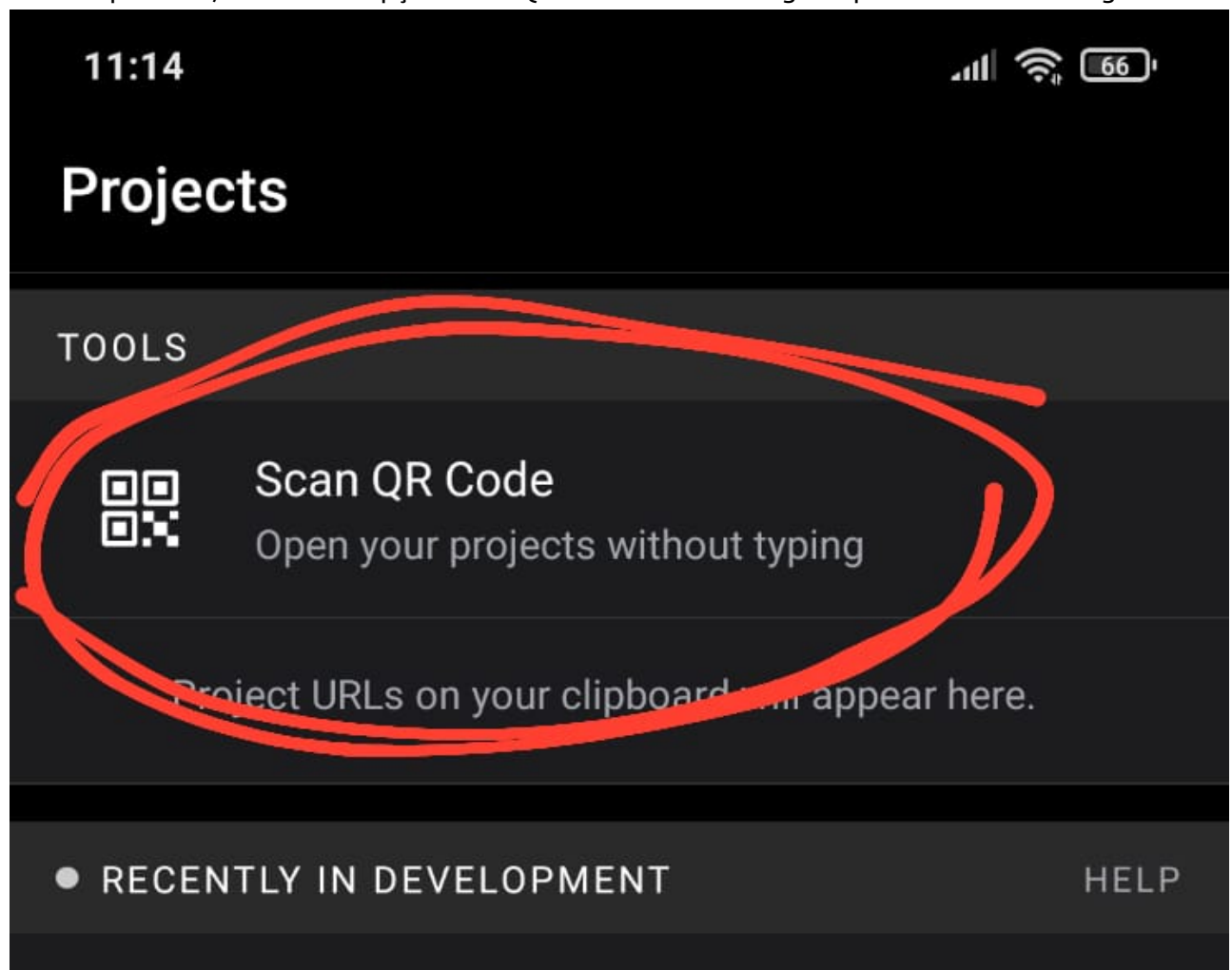
No terminal do Linux, dentro da pasta app digite:

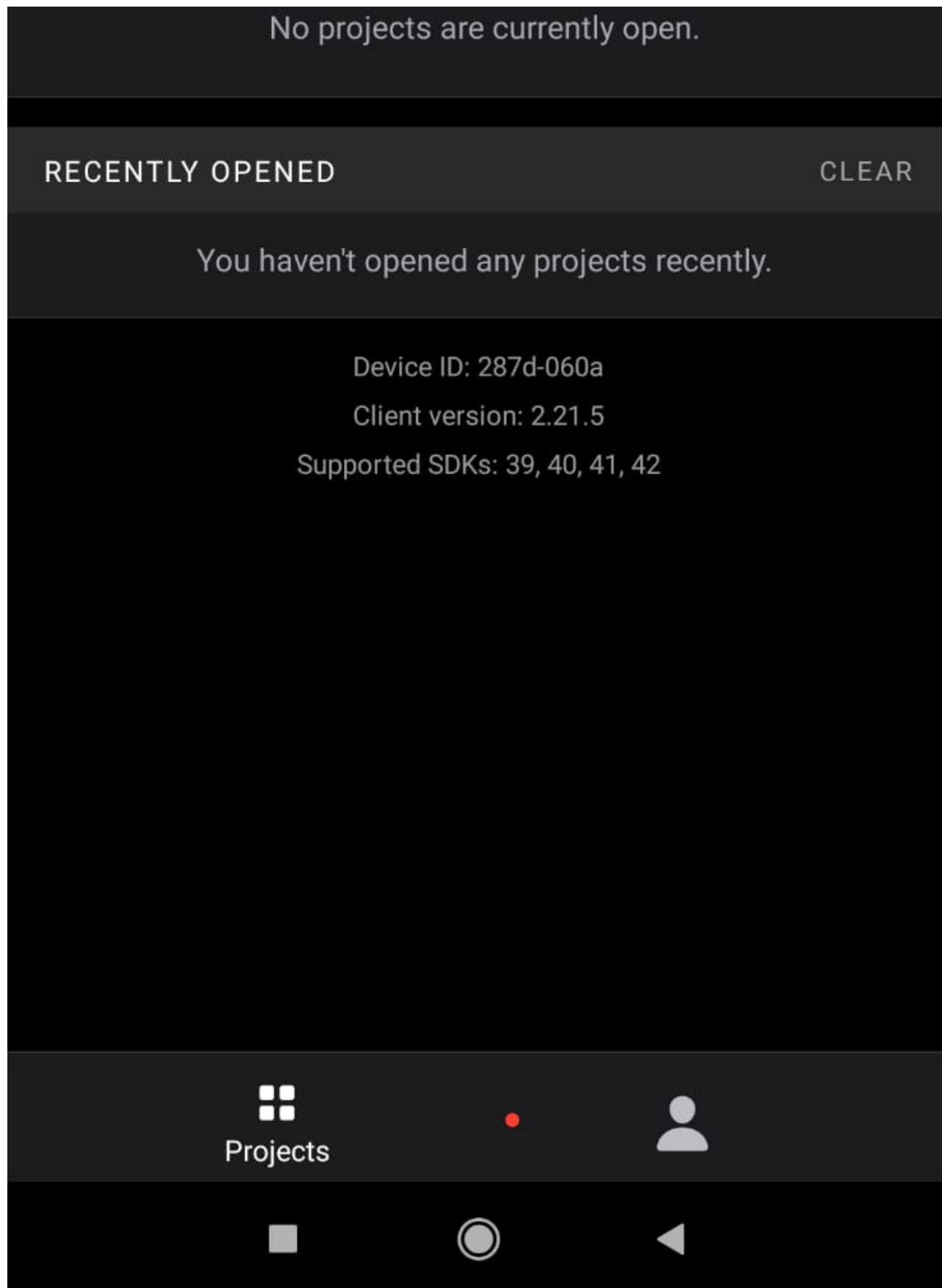
```
yarn start
```

Uma página será aberta no navegador.

Na parte esquerda, acima do código QR, clique em LAN e aguarde a iniciação.

- No seu smartphone, instale o aplicativo Expo, disponível para Android e IOS.
<https://apps.apple.com/br/app/expo-go/id982107779>
https://play.google.com/store/apps/details?id=host.exp.exponent&hl=pt_BR&gl=US
- Abra o aplicativo, selecione a opção SCAN QR CODE e leia o código disponibilizado no navegador.





Aguarde a instalação, o aplicativo será aberto no modo de desenvolvimento.

- Disponibilizamos também um usuário administrador com registros criados previamente para facilitar os testes, você pode efetuar o login em tal usuário utilizando as seguintes credenciais:

Login: admin

Senha: admin

- Divirta-se!

2. Documentação APIs

Aqui listamos as APIs disponíveis. Para mais informações e exemplos, importe o arquivo do insomnia disponível na raiz do projeto.

- Public Routes

- Login [POST]

```
/api/accounts/token/
```

- Create User [POST]

```
/api/accounts/users/
```

- Private Routes

Requer header: Authorization: "Token {USER_TOKEN}"

Também requer que o objeto em questão pertença ao usuário que efetua a requisição.

- USER

- Read User [GET]

```
/api/accounts/users/{token_user_id}/
```

- Update User [PUT]

```
/api/accounts/users/{token_user_id}/
```

- Delete User [DELETE]

```
/api/accounts/users/{token_user_id}/
```

- LOCALS

- Create Local [POST]

```
/api/locals/
```

- Read Local [GET]

```
/api/locals/{local_id}/
```

- Update Local [PUT]

```
/api/locals/{local_id}/
```

- Delete Local [DELETE]

```
/api/locals/{local_id}/
```

- List Locals [GET]

```
/api/locals/
```

- ROUTES

- Generate [POST]

```
/api/routes/generate/
```