# Java Rest com Ionic

Glauco Todesco

# Endereço dos repositórios GITHub

- Servidor Rest

https://github.com/glaucotodesco/Exemplo_Rest_JSON_JPA.git

- Cliente Ionic:

https://github.com/glaucotodesco/Exemplo_Rest_JSON_Client_Ionic.git

# O que instalar?

Testado com o Windows 10 64 bits

Lado do Servidor:

- Java 1.8
- Eclipse Oxygen.1a Release (4.7.1a) - J2EE
- Bando de Dados Derby (vem junto com o Java SDK)
- Tomcat 8.0

Lado do Cliente (App)

- Node.js – 8.9.1
- Ionic 3.19.0
- Cordova 7.1.0
- Visual Studio Code 1.18.0
- Chrome plugin (CROPS) - Moesif Origin & CORS Changer 0.2.4
- Chrime apps Advanced REST cliente 10.0.11-stable

# Servidor

# Construindo o servidor

- No eclipse construa um projeto maven e configure o arquivo pom.xml
- Depois que o projeto for construído incluir os Project FACETS:
  - JPA – Para criar o arquivo persistence.xml
  - JSF – Para criar a pasta WEB-INF e o arquivo web.xml

# Build do pom.xml

```
<build>
	<plugins>
		<plugin>
			<artifactId>maven-compiler-plugin</artifactId>
			<version>3.1</version>
			<configuration>
				<source>1.8</source>
				<target>1.8</target>
			</configuration>
		</plugin>
	</plugins>
</build>
```

# Dependências - pom.xml

Derby:

```
<dependency>
        <groupId>org.apache.derby</groupId>
        <artifactId>derbyclient </artifactId>
        <version>10.2.2.0</version>
</dependency>
```

# Dependências - pom.xml

JPA:

```
<dependency>
	<groupId>org.eclipse.persistence</groupId>
	<artifactId>org.eclipse.persistence.jpa</artifactId>
	<version>2.6.2</version>
</dependency>
```

# Dependências - pom.xml

JERSEY:

```
<dependency>
		<groupId>org.glassfish.jersey.core</groupId>
		<artifactId>jersey-server</artifactId>
		<version>2.6</version>
</dependency>

<dependency>
		<groupId>org.glassfish.jersey.containers</groupId>
		<artifactId>jersey-container-servlet</artifactId>
		<version>2.6</version>
</dependency>
```

# Dependências - pom.xml

JACKSON:

```
<dependency>
	<groupId>com.fasterxml.jackson.jaxrs</groupId>
	<artifactId>jackson-jaxrs-json-provider</artifactId>
	<version>2.4.1</version>
</dependency>
```

# web.xml

- Configure o web.xml do projeto para receber as chamadas REST

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name>ProjetoWSRestJSON</display-name>
    <servlet>
      <servlet-name>Jersey REST Service</servlet-name>
      <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
      <init-param>
        <param-name>jersey.config.server.provider.packages</param-name>
        <param-value>com.mysystem.webservices</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
      <servlet-name>Jersey REST Service</servlet-name>
      <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

# web.xml

- Observe essa linha no web.xml

**<param-value>com.mysystem.webservices</param-value>**

Ela define o pacote que as classes de serviços REST devem ficar.

Altere esse caminho conforme o projeto.

# web.xml

- Observe essa linha no web.xml

**<url-pattern>/rest/*</url-pattern>**

Ela define o caminho para os serviços de REST dentro da aplicação.

Altere esse caminho conforme o projeto.

# persistence.xml

Acrescente isso ao persistence.xml do JPA:

```
<class>com.mysystem.entities.Client</class>
<properties>
        <property name="javax.persistence.jdbc.url“
                        value="jdbc:derby://localhost:1527/DBClient;create=true"/>
        <property name="javax.persistence.jdbc.user" value="app"/>
        <property name="javax.persistence.jdbc.driver"
                        value="org.apache.derby.jdbc.ClientDriver"/>
        <property name="eclipselink.ddl-generation" value="create-tables"/>
</properties>
</persistence-unit>
```

# Construir a classe de entidade do JPA

- Acrescente o toString, equals e hashCode.

```
1  package com.mysystem.entities;
2  import java.io.Serializable;
3  import javax.persistence.Entity;
4  import javax.persistence.GeneratedValue;
5  import javax.persistence.GenerationType;
6  import javax.persistence.Id;
7    @Entity
8    public class Client implements Serializable {
9          private static final long serialVersionUID = -3842844978617110554L;
10         @Id
11         @GeneratedValue(strategy=GenerationType.AUTO)
12         private int id;
13         private String userName;
14         private String password;
15         private String address;
16         private String email;
17
18         public int getId() {
19               return id;
20         }
21
```

# Construindo a classe de Serviços do JPA

```java
package com.mysystem.services;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import com.mysystem.entities.Client;

public class ClientService {
    private static EntityManagerFactory emf;


    public ClientService() {
        emf = Persistence.createEntityManagerFactory("ProjetoWSRestJSON");
    }

    public Client save(Client client){

        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();
        em.persist(client);
        em.flush();
        em.getTransaction().commit();
        em.close();
        return client;
    }
```

# Construindo a classe de Serviços REST

- Lembre-se que essa classe deve ficar dentro do pacote definido do web.xml

```
1  package com.mysystem.webservices;
2  import java.util.List;
14
15 @Path("/client")
16 public class RestClient {
17
18         //Instalar o Chrome Advanced REST Client
19         //Para abrir: chrome://apps/  -> Escolha o ARC
20         //Referencia: http://o7planning.org/en/11207/simple-crud-example-with-ja
21
22
23         //http://localhost:8080/ProjetoWSRestJSON/rest/user/client/all
24         @GET
25         @Path("/all")
26         @Produces(MediaType.APPLICATION_JSON)
27         public List<Client> getAllClients(){
28             List<Client> clients;
29             ClientService service = new ClientService();
30             clients = service.getAllClients();
31             return clients;
32         }
33
34         //Exemplo: http://localhost:8080/ProjetoWSRestJSON/rest/client/findById
35         @GET
36         @Path("/findById")
37         @Produces(MediaType.APPLICATION_JSON)
```

# Construindo o arquivo index.html

- Isso não é obrigatório, somente para facilitar o acesso aos serviços ao executar a aplicação.
- Funciona somente para métodos GETS!

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="ISO-8859-1">
5  <title>Insert title here</title>
6  </head>
7  <body>
8  <h1>Instrucões</h1>
9
10 <h2>Instalar o Chrome Advanced REST Client ou equivalente</h2>
11 <h2>Comandos gets podem ser testado direto no navegador</h2>
12
13 <br/><a href="rest/client/all" >Get All Clients</a>
14 <br/><a href="rest/client/findById?id=1" >Get Client By Id - Alterar o id na url</a>
15 </body>
16 </html>
17
```

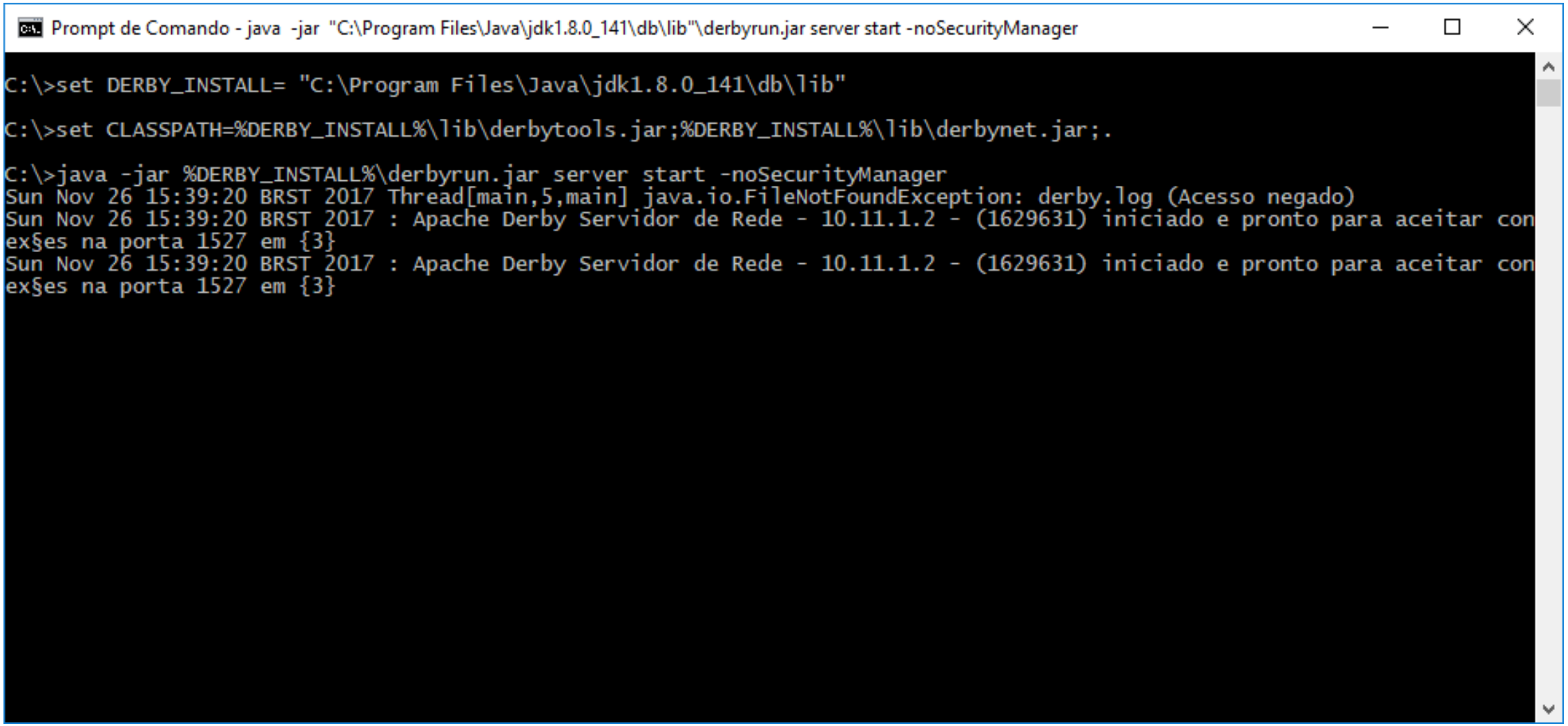
# Derby Start

- Script:

```
set DERBY_INSTALL= “C:\Program Files\Java\jdk1.8.0_141\db\lib”
set CLASSPATH=%DERBY_INSTALL%\lib\derbytools.jar;%DERBY_INSTALL%\lib\derbynet.jar;.
java -jar %DERBY_INSTALL%\derbyrun.jar server start –noSecurityManager
```

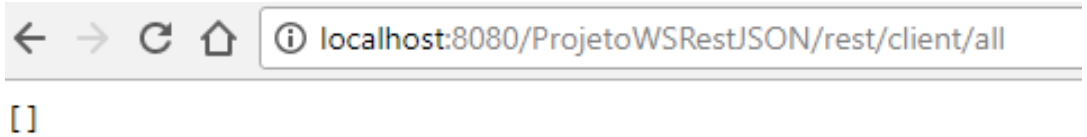
**Não fechar a janela para não finalizar a execução do Derby!**

# Derby Start


Prompt de Comando - java -jar "C:\Program Files\Java\jdk1.8.0_141\db\lib"\derbyrun.jar server start -noSecurityManager
C:\>set DERBY_INSTALL= "C:\Program Files\Java\jdk1.8.0_141\db\lib"
C:\>set CLASSPATH=%DERBY_INSTALL%\lib\derbytools.jar;%DERBY_INSTALL%\lib\derbynet.jar;.
C:\>java -jar %DERBY_INSTALL%\derbyrun.jar server start -noSecurityManager
Sun Nov 26 15:39:20 BRST 2017 Thread[main,5,main] java.io.FileNotFoundException: derby.log (Acesso negado)
Sun Nov 26 15:39:20 BRST 2017 : Apache Derby Servidor de Rede - 10.11.1.2 - (1629631) iniciado e pronto para aceitar con
ex§es na porta 1527 em {3}
Sun Nov 26 15:39:20 BRST 2017 : Apache Derby Servidor de Rede - 10.11.1.2 - (1629631) iniciado e pronto para aceitar con
ex§es na porta 1527 em {3}

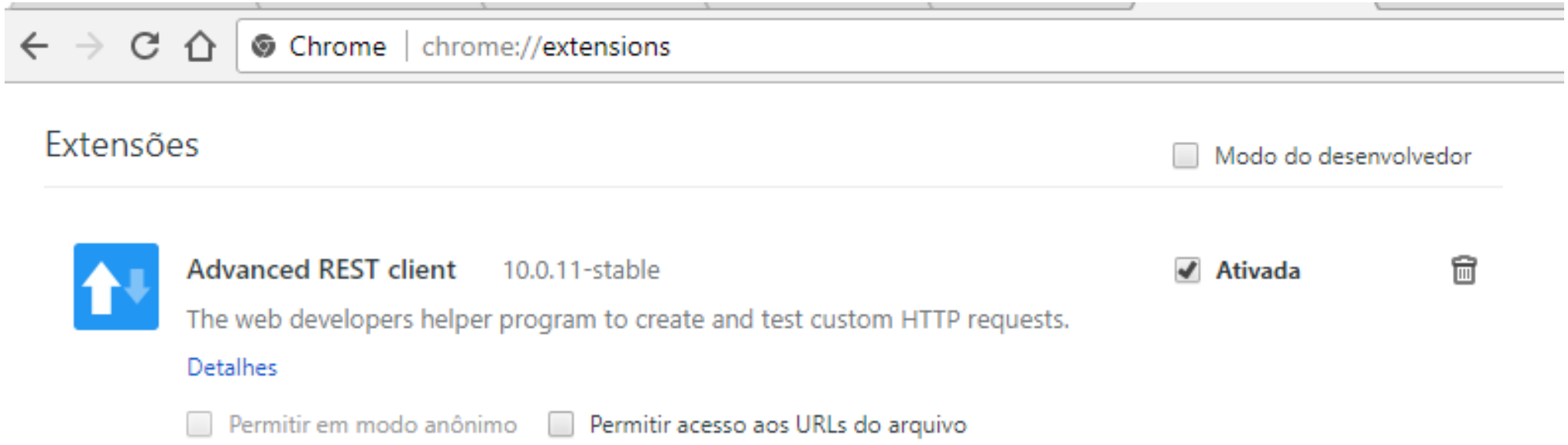
# Executando a aplicação:

- Execute o arquivo index.html

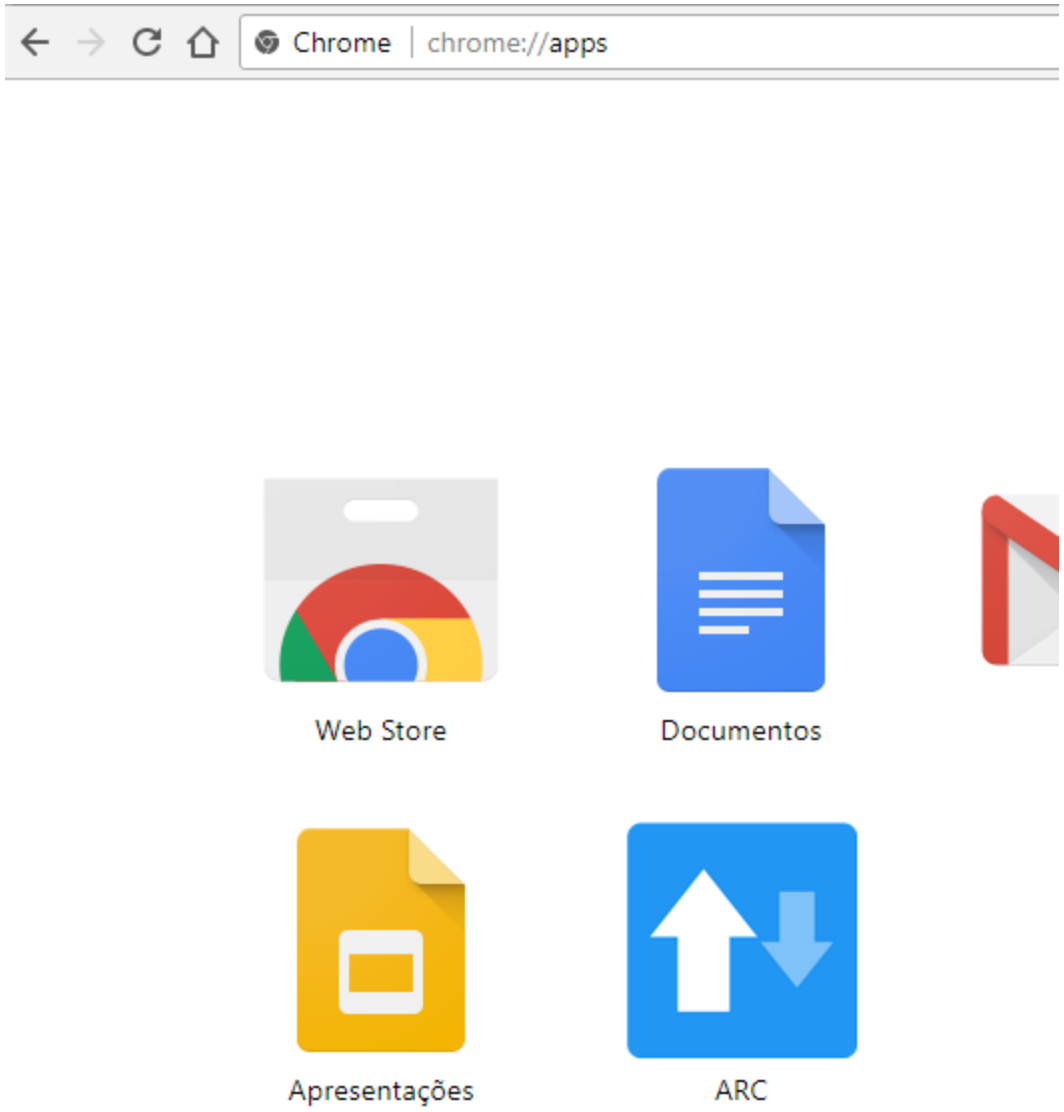
# Executando POST, DELETE e UPDATE

- Para executar os demais métodos do HTTP foi usado o seguinte app do Google Chrome. Instale para poder executar os demais métodos HTTP.



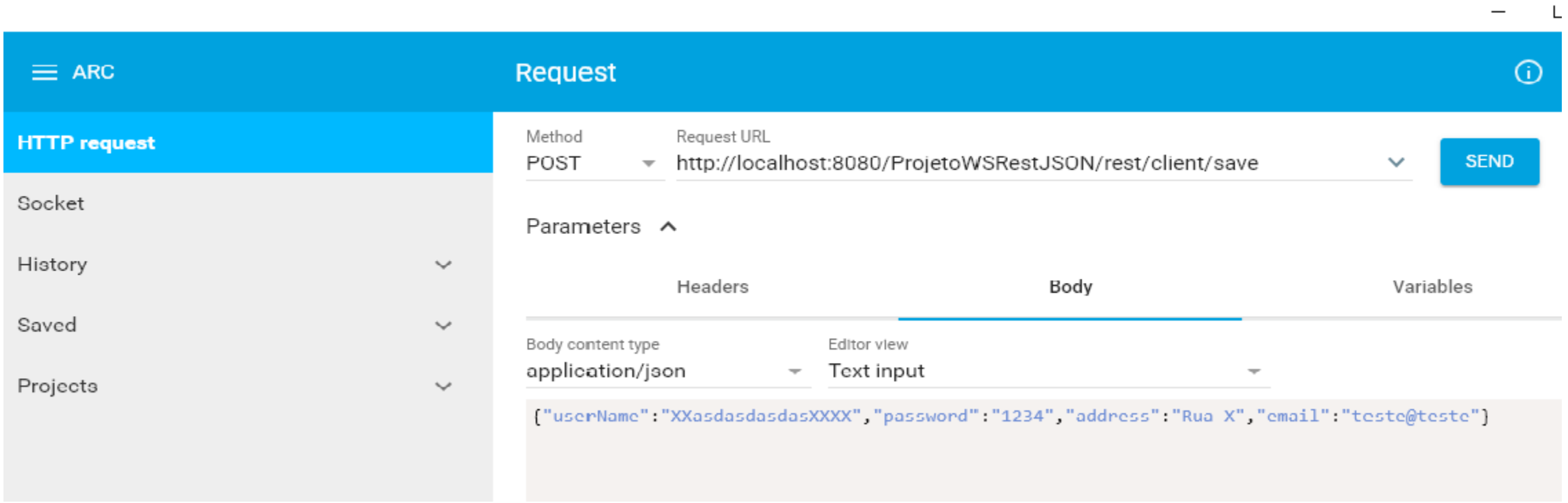
# Chrome Apps:

- Escolha a ARC

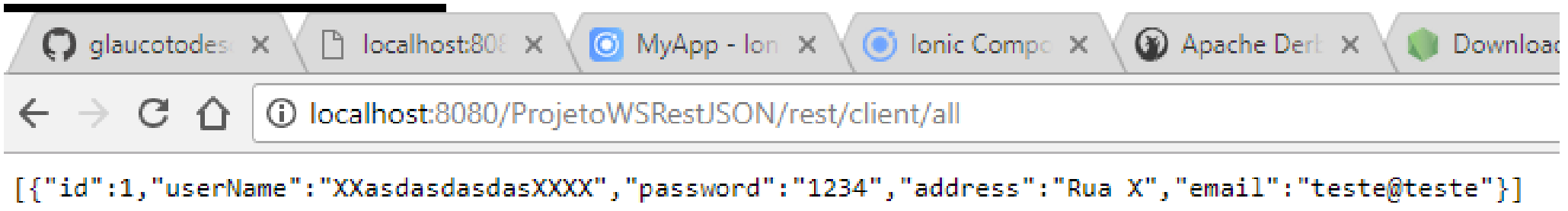


# Exemplo com POST

- Salvando um novo cliente:



# Resultado após o POST



# Cliente Ionic

# Instale o Noje.js, Ionic e Cordova

Node.js

https://nodejs.org/en/download/

Na linha de comando instalar o Ionic e o Cordova:

*npm install -g ionic cordova*

# Ionic Novo Projeto

Construindo uma nova aplicação.

Na linha de comando digite:

*ionic start RestApp tabs*

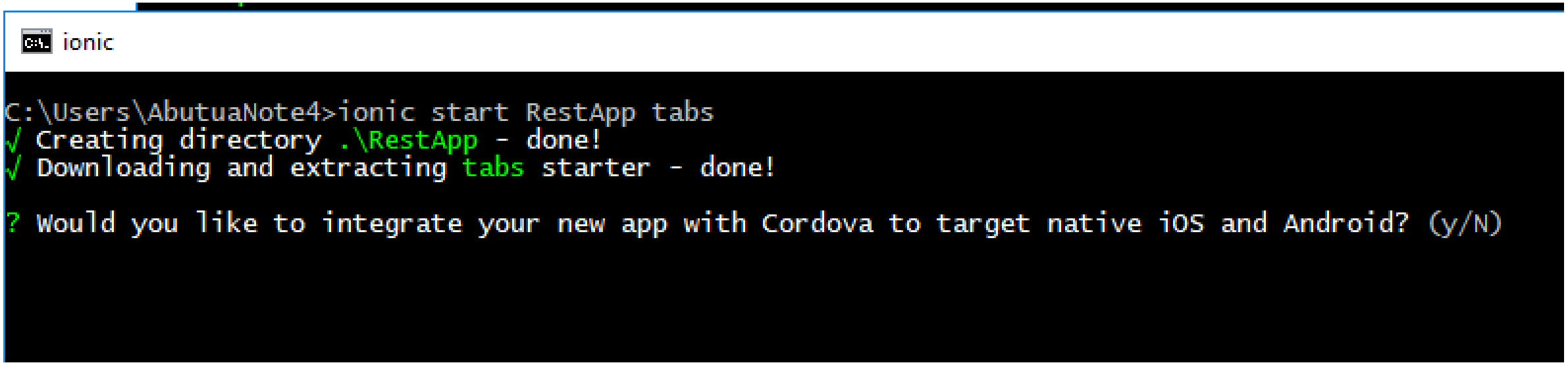
Start  -> novo projeto

RestApp -> novo do projeto

tabs -> template do novo projeto

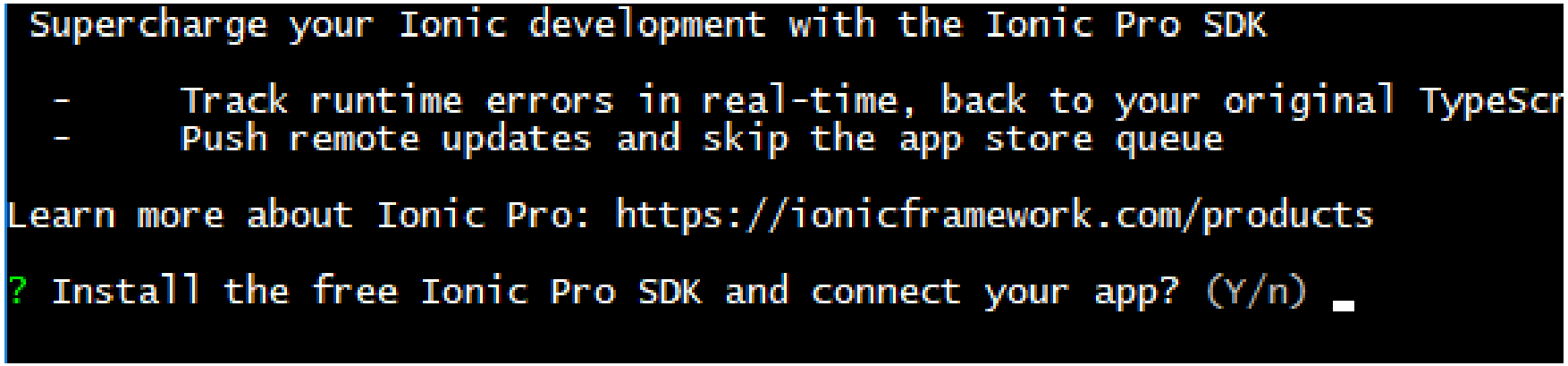
# Ionic - Novo Projeto

- No momento definir não para as plataformas iOS e Android.



# Ionic - Novo Projeto

- No momento definir não para o Ionic Pro SDK.



# Executando a aplicação

- Entre na pasta do projeto:

cd RestApp

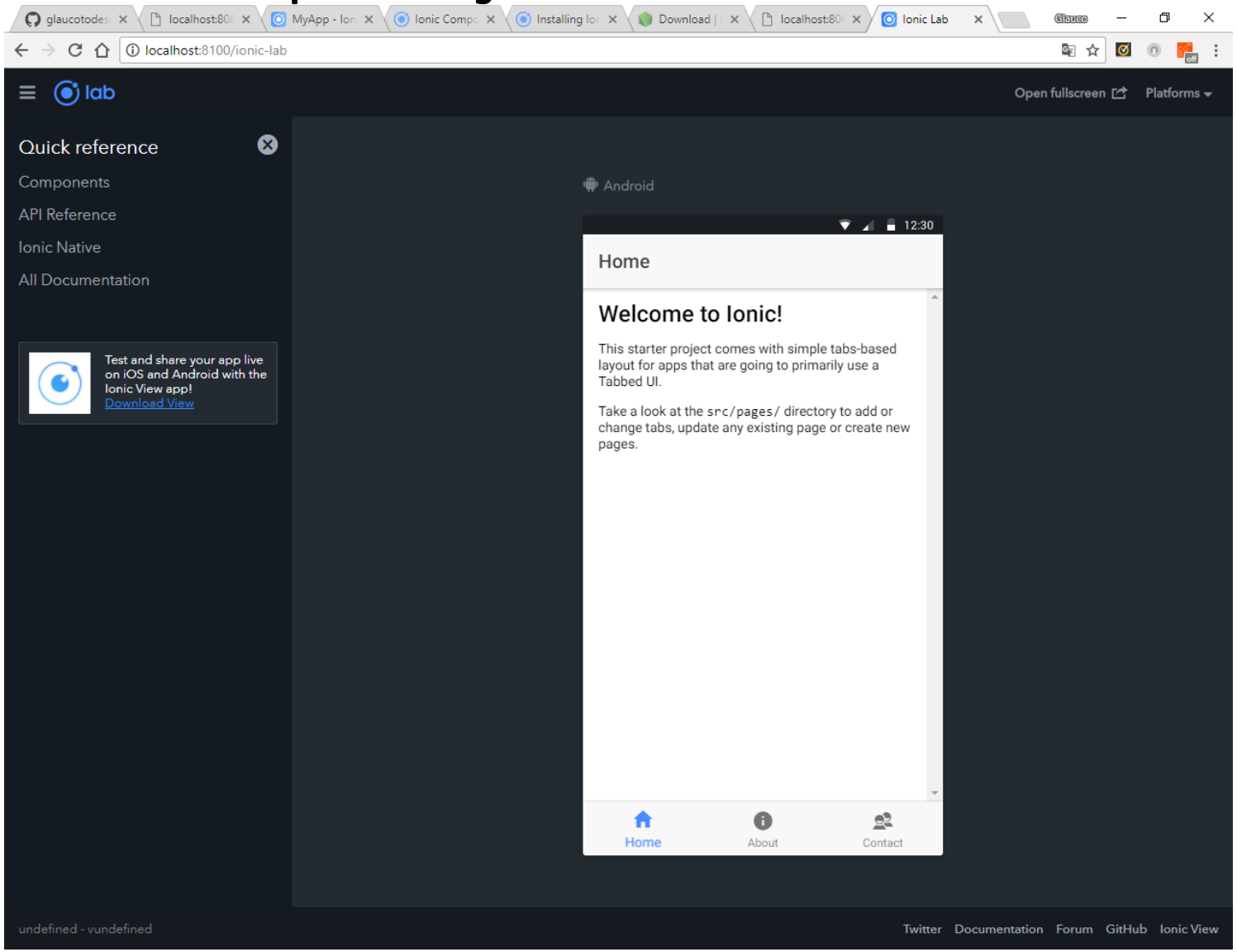
- Faça o build do projeto:

ionic build

- Execute a aplicação no modo serve:

Ionic serve --lab

# Executando a aplicação



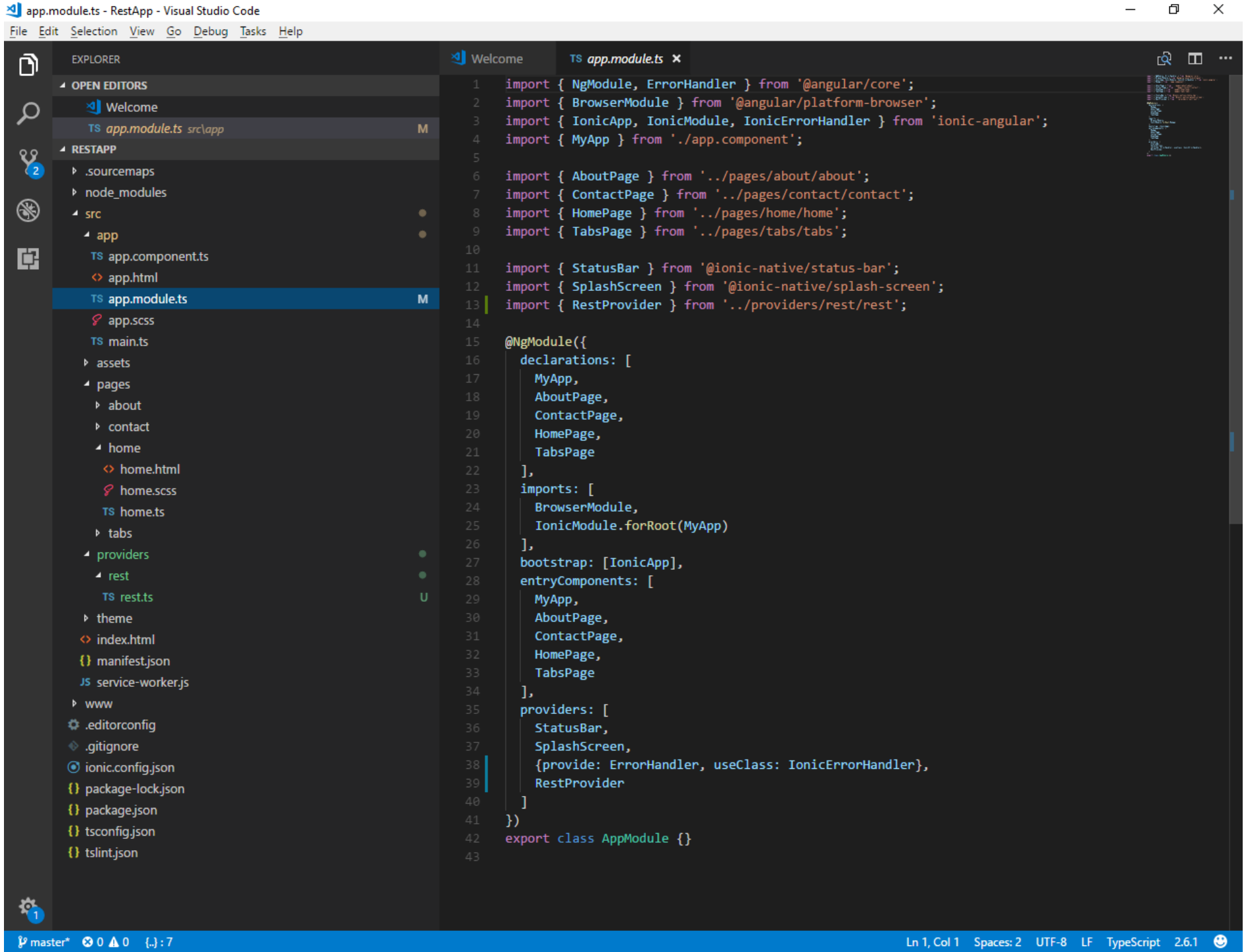
# Alterando a aplicação

- Na linha de comando adicione um novo provider para tratar das requisições REST:

*ionic g provider Rest*

# Visual Studio Code

- Para facilitar o desenvolvimento abra a pasta do projeto (opção open folder) do Visual Studio Code ou use um outro editor.


app.module.ts - RestApp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
Welcome
app.module.ts src\app
RESTAPP
.sourcemaps
node_modules
src
app
app.component.ts
app.html
app.module.ts
app.scss
main.ts
assets
pages
about
contact
home
home.html
home.scss
home.ts
tabs
providers
rest
rest.ts
theme
index.html
manifest.json
service-worker.js
www
.editorconfig
.gitignore
ionic.config.json
package-lock.json
package.json
tsconfig.json
tslint.json
Welcome
app.module.ts
import { NgModule, ErrorHandler } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';
import { AboutPage } from '../pages/about/about';
import { ContactPage } from '../pages/contact/contact';
import { HomePage } from '../pages/home/home';
import { TabsPage } from '../pages/tabs/tabs';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { RestProvider } from '../providers/rest/rest';
@NgModule({
declarations: [
MyApp,
AboutPage,
ContactPage,
HomePage,
TabsPage
],
imports: [
BrowserModule,
IonicModule.forRoot(MyApp)
],
bootstrap: [IonicApp],
entryComponents: [
MyApp,
AboutPage,
ContactPage,
HomePage,
TabsPage
],
providers: [
StatusBar,
SplashScreen,
{provide: ErrorHandler, useClass: IonicErrorHandler},
RestProvider
]
})
export class AppModule {}
master* 0 0 {..} : 7
Ln 1, Col 1 Spaces: 2 UTF-8 LF TypeScript 2.6.1

# Arquivos alterados:

- src/app/**app.module.ts**
- src/providers/rest/**rest.ts**
- src/pages/home/**home.ts**
- src/pages/home/**home.html**

# src/app/app.module.ts

Acrescente um novo import:

**import { HttpClientModule } from '@angular/common/http';**

Incluir o modulo:

```
imports: [
  BrowserModule,
  IonicModule.forRoot(MyApp),
  HttpClientModule,
],
```

app.module.ts - RestApp - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER

OPEN EDITORS
- app.module.ts src\app M
- rest.ts src\providers\rest U

RESTAPP
- .sourcemaps
- node_modules
- src
  - app
    - app.component.ts
    - app.html
    - app.module.ts M
    - app.scss
    - main.ts
  - assets
  - pages
    - about
      - about.html
      - about.scss
      - about.ts
    - contact
    - home
      - home.html
      - home.scss
      - home.ts
    - tabs
  - providers
    - rest
      - rest.ts U
  - theme
  - index.html
  - manifest.json
  - service-worker.js
- www
- .editorconfig
- .gitignore
- ionic.config.json
- package-lock.json
- package.json
- tsconfig.json
- tslint.json

app.module.ts | rest.ts

```ts
import { NgModule, ErrorHandler } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';

import { AboutPage } from '../pages/about/about';
import { ContactPage } from '../pages/contact/contact';
import { HomePage } from '../pages/home/home';
import { TabsPage } from '../pages/tabs/tabs';
import { HttpClientModule } from '@angular/common/http';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { RestProvider } from '../providers/rest/rest';

@NgModule({
  declarations: [
    MyApp,
    AboutPage,
    ContactPage,
    HomePage,
    TabsPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    HttpClientModule
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    AboutPage,
    ContactPage,
    HomePage,
    TabsPage
  ],
  providers: [
    StatusBar,
    SplashScreen,
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell

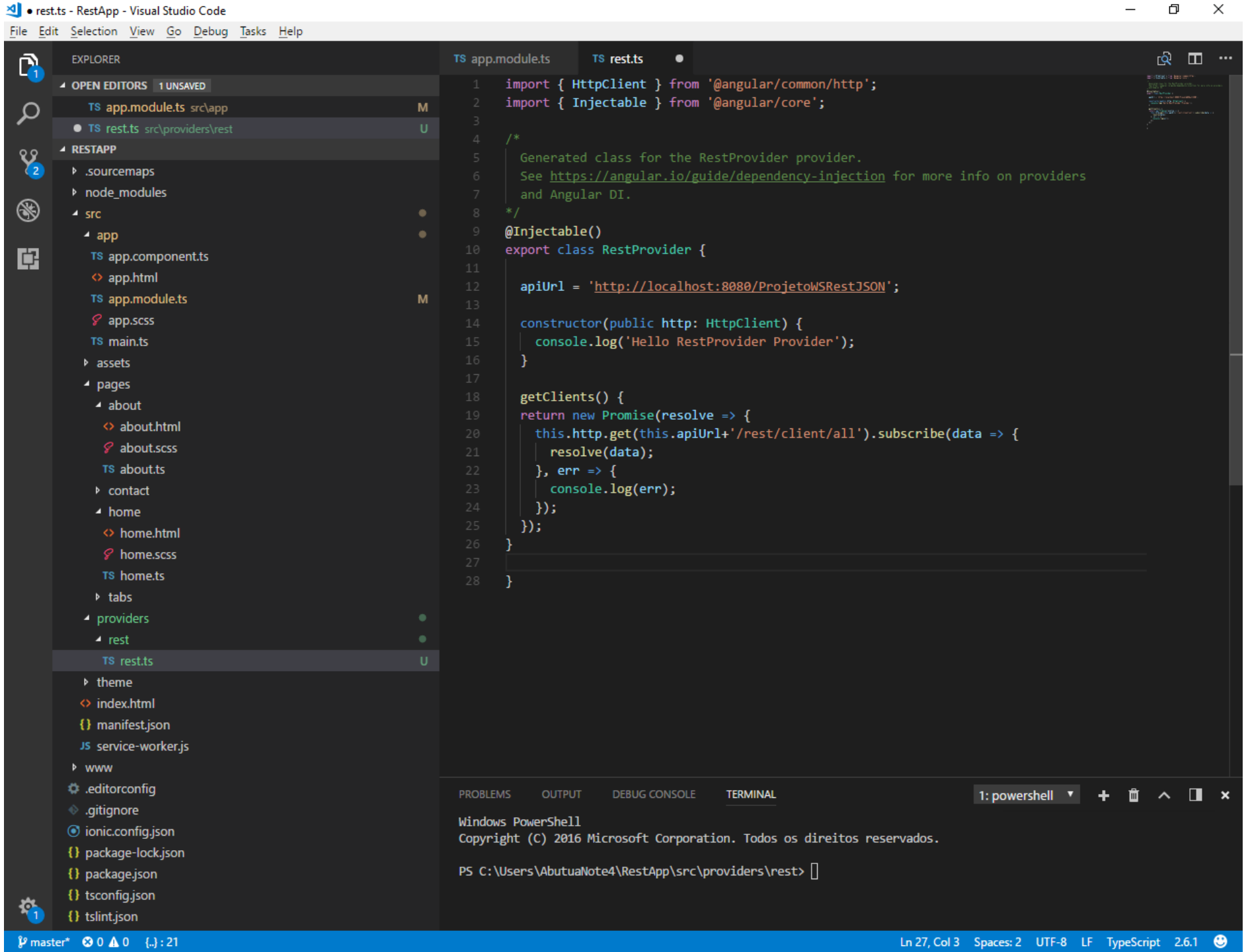
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.

PS C:\Users\AbutuaNote4\RestApp\src\providers\rest>
```

master* 0 0 {..} : 21 Ln 30, Col 11 Spaces: 2 UTF-8 LF TypeScript 2.6.1

# src/providers/rest/rest.ts

No provider definir como consumir o serviço de REST:

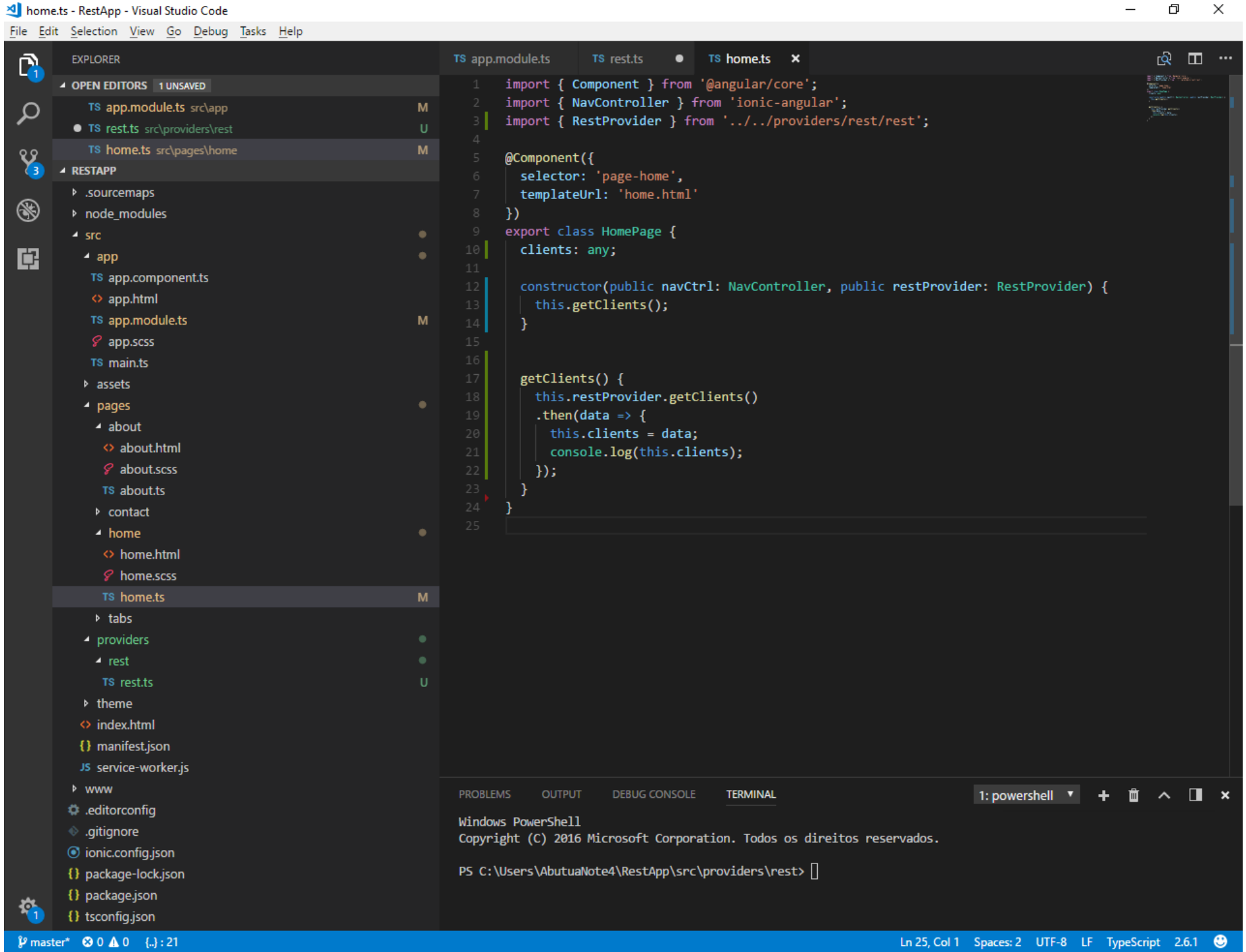
```
apiUrl = 'http://localhost:8080/ProjetoWSRestJSON';

getClients() {
        return new Promise(resolve => {
                this.http.get(this.apiUrl+'/rest/client/all').subscribe(data => {
                resolve(data);
                }, err => {
                console.log(err);
                });
        });
}
```


● rest.ts - RestApp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS 1 UNSAVED
app.module.ts src\app M
rest.ts src\providers\rest U
RESTAPP
.sourcemaps
node_modules
src
app
app.component.ts
app.html
app.module.ts M
app.scss
main.ts
assets
pages
about
about.html
about.scss
about.ts
contact
home
home.html
home.scss
home.ts
tabs
providers
rest
rest.ts U
theme
index.html
manifest.json
service-worker.js
www
.editorconfig
.gitignore
ionic.config.json
package-lock.json
package.json
tsconfig.json
tslint.json
app.module.ts
rest.ts
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
/*
Generated class for the RestProvider provider.
See https://angular.io/guide/dependency-injection for more info on providers
and Angular DI.
*/
@Injectable()
export class RestProvider {
apiUrl = 'http://localhost:8080/ProjetoWSRestJSON';
constructor(public http: HttpClient) {
console.log('Hello RestProvider Provider');
}
getClients() {
return new Promise(resolve => {
this.http.get(this.apiUrl+'/rest/client/all').subscribe(data => {
resolve(data);
}, err => {
console.log(err);
});
});
}
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.
PS C:\Users\AbutuaNote4\RestApp\src\providers\rest>
master* 0 0 {..} : 21
Ln 27, Col 3 Spaces: 2 UTF-8 LF TypeScript 2.6.1

# src/pages/home/home.ts

Definir o controle da home para usar o provider:

```
getClients() {
  this.restProvider.getClients()
  .then(data => {
    this.clients = data;
    console.log(this.clients);
  });
```


home.ts - RestApp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS 1 UNSAVED
app.module.ts src\app M
rest.ts src\providers\rest U
home.ts src\pages\home M
RESTAPP
.sourcemaps
node_modules
src
app
app.component.ts
app.html
app.module.ts M
app.scss
main.ts
assets
pages
about
about.html
about.scss
about.ts
contact
home
home.html
home.scss
home.ts M
tabs
providers
rest
rest.ts U
theme
index.html
manifest.json
service-worker.js
www
.editorconfig
.gitignore
ionic.config.json
package-lock.json
package.json
tsconfig.json
app.module.ts
rest.ts
home.ts
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { RestProvider } from '../../providers/rest/rest';
@Component({
selector: 'page-home',
templateUrl: 'home.html'
})
export class HomePage {
clients: any;
constructor(public navCtrl: NavController, public restProvider: RestProvider) {
this.getClients();
}
getClients() {
this.restProvider.getClients()
.then(data => {
this.clients = data;
console.log(this.clients);
});
}
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.
PS C:\Users\AbutuaNote4\RestApp\src\providers\rest>
master* 0 0 {..} : 21
Ln 25, Col 1 Spaces: 2 UTF-8 LF TypeScript 2.6.1

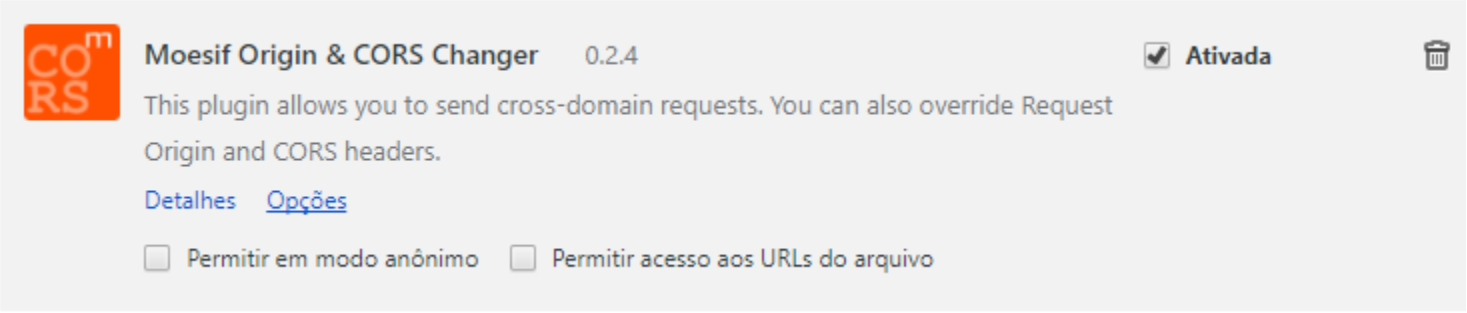
# src/pages/home/home.html

Definir o view da home para apresentar os dados do REST:

```
<ion-list inset>
  <ion-item *ngFor="let client of clients">
    <h2>{{client.userName}}</h2>
    <p>{{client.email}}</p>
  </ion-item>
 </ion-list>
```

# CORS

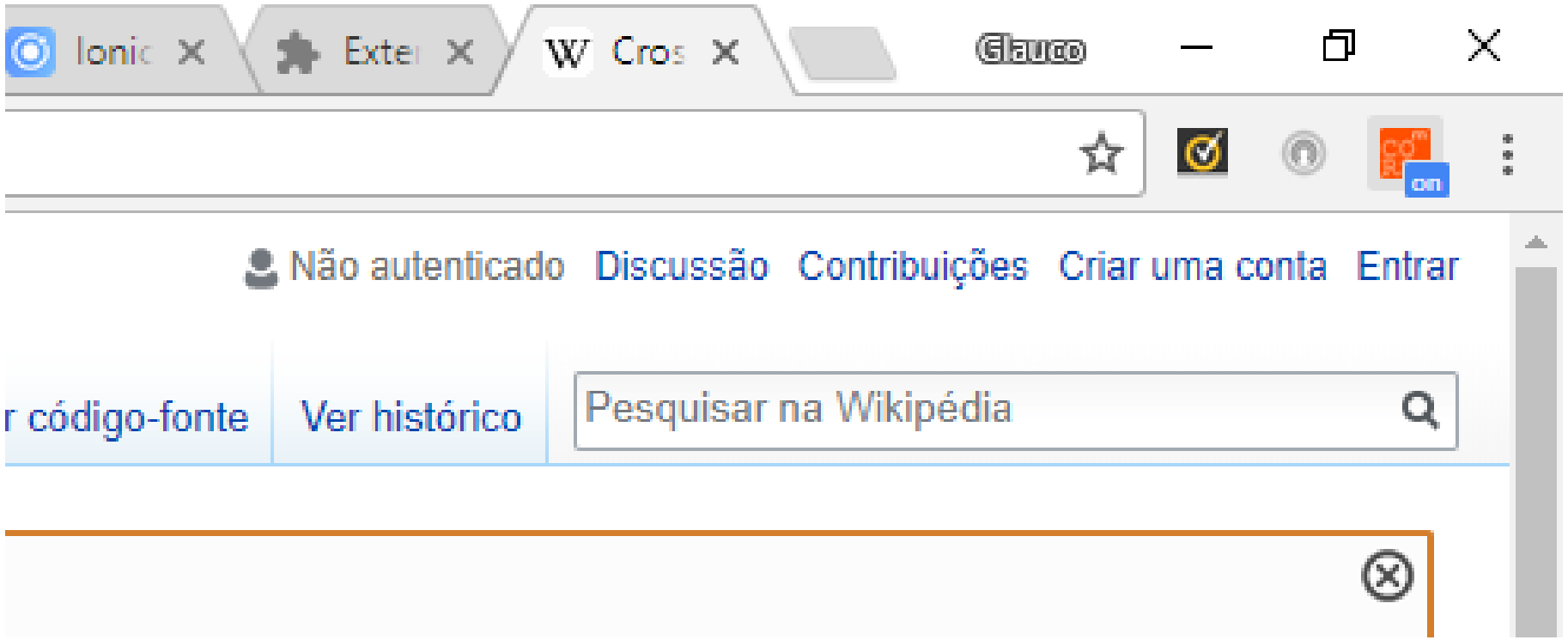
- Antes de executar é necessário tratar  CORS.
- Uma das formas é instalar o plugin  Moesif no Chrome:




https://pt.wikipedia.org/wiki/Cross-origin_resource_sharing

# CORS

- Deixe o CORS habilitado (Canto Superior Direito do Chrome):



# Executando

Certifique-se que:

- O derby esteja em execução.
- A aplicação servidor no tomcat esteja em execução.
- CORS instalado e ativado no Google Chrome.
- Pelo menos um cliente cadastrado no Derby.

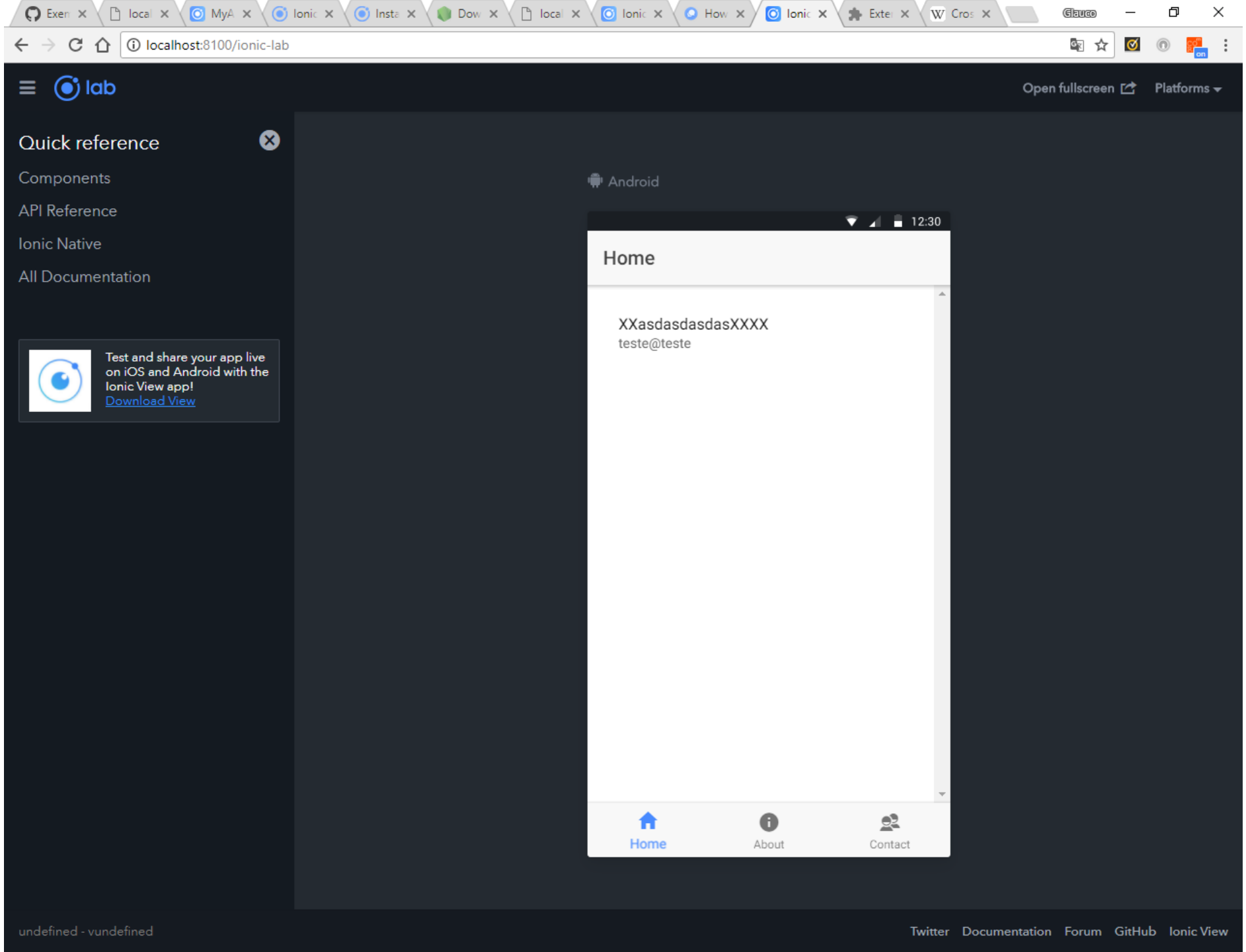
# Executando a aplicação

- Faça o build do projeto:

ionic build

- Execute a aplicação no modo serve:

Ionic serve --lab


localhost:8100/ionic-lab
lab
Open fullscreen
Platforms
Quick reference
Components
API Reference
Ionic Native
All Documentation
Test and share your app live on iOS and Android with the Ionic View app!
Download View
Android
12:30
Home
XXasdasdasdasdasXXXX
teste@teste
Home
About
Contact
undefined - vundefined
Twitter
Documentation
Forum
GitHub
Ionic View