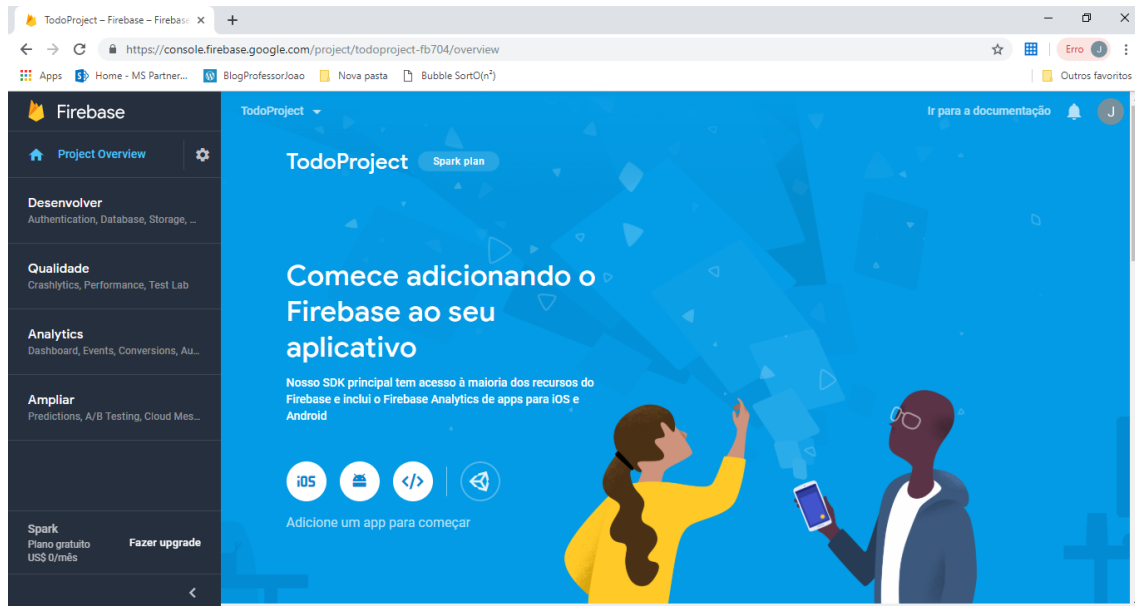


Criando Banco de Dados

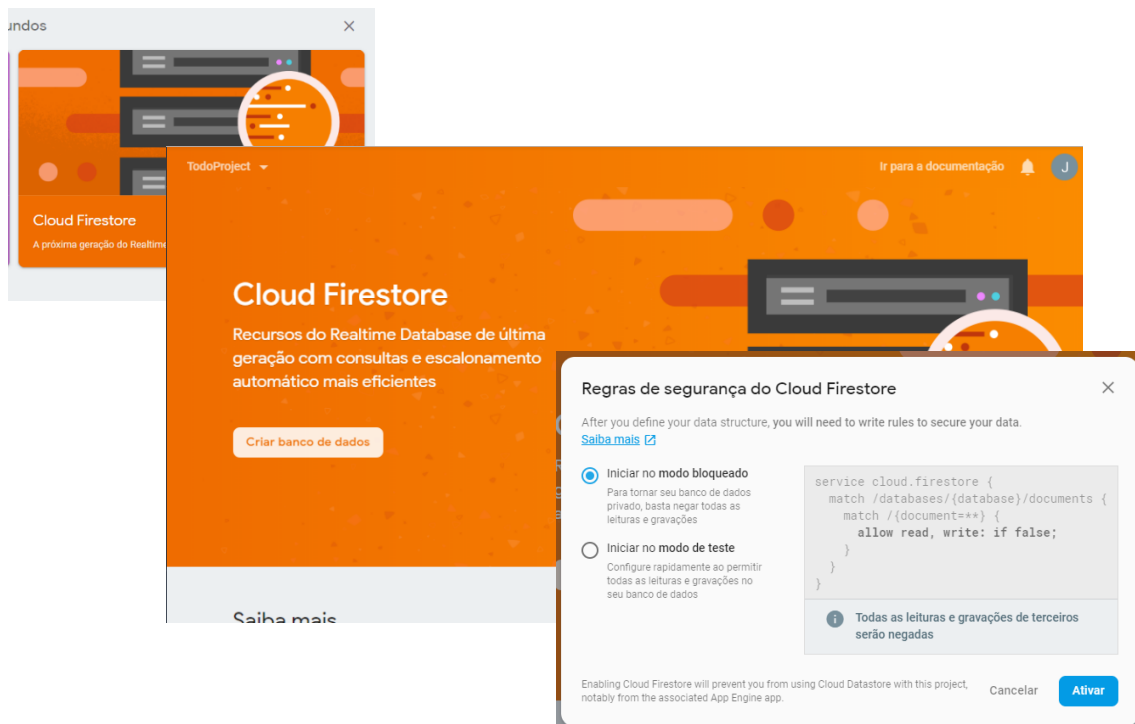
1. Acessar o Console do Firebase e efetuar login:
<https://console.firebase.google.com/?pli=1>
2. Adicionar novo projeto (eg TodoProject)



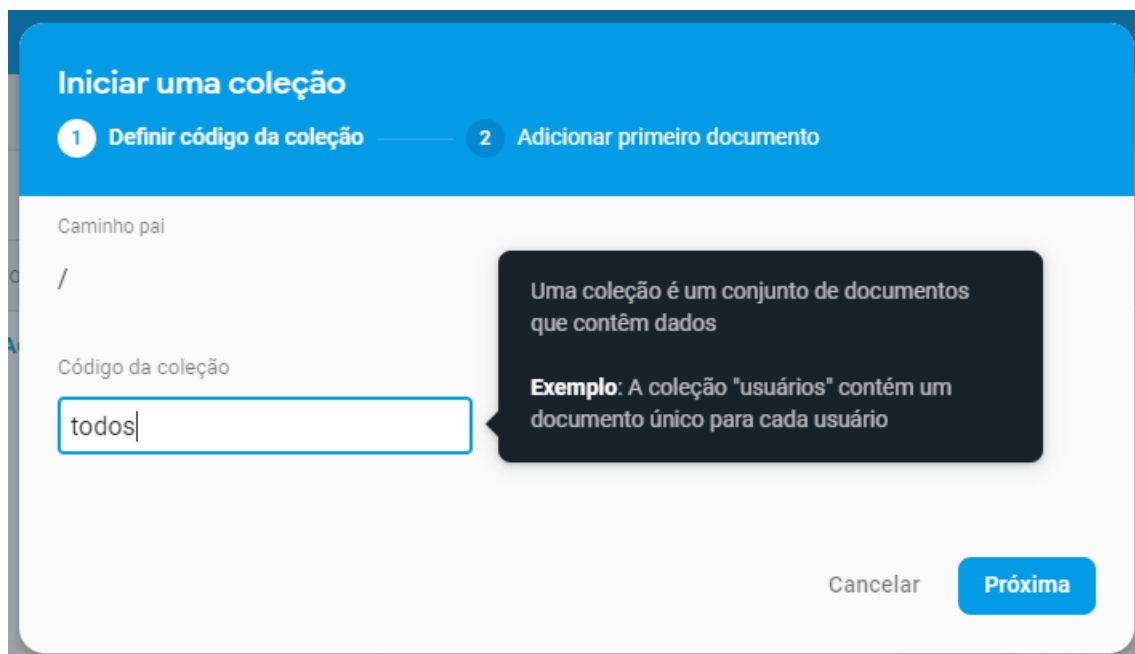
3. Selecionar a opção que indica uso do Firebase na Web e copiar o código Javascript (será usado mais tarde):



4. Criar novo banco de dados Firestore e escolher a opção modo teste:



5. Escolher regras de segurança e ativar. Elas podem ser modificadas depois no Firestore.
6. Adicionar coleção de objetos (eg. todos):



7. Especificar campos (eg. description e completed):

Iniciar uma coleção

✓ Definir código da coleção

2 Adicionar primeiro documento

Caminho pai do documento

/todos

Código do documento

Código automático

Campo	Tipo	Valor
description	string	
completed	boolean	false

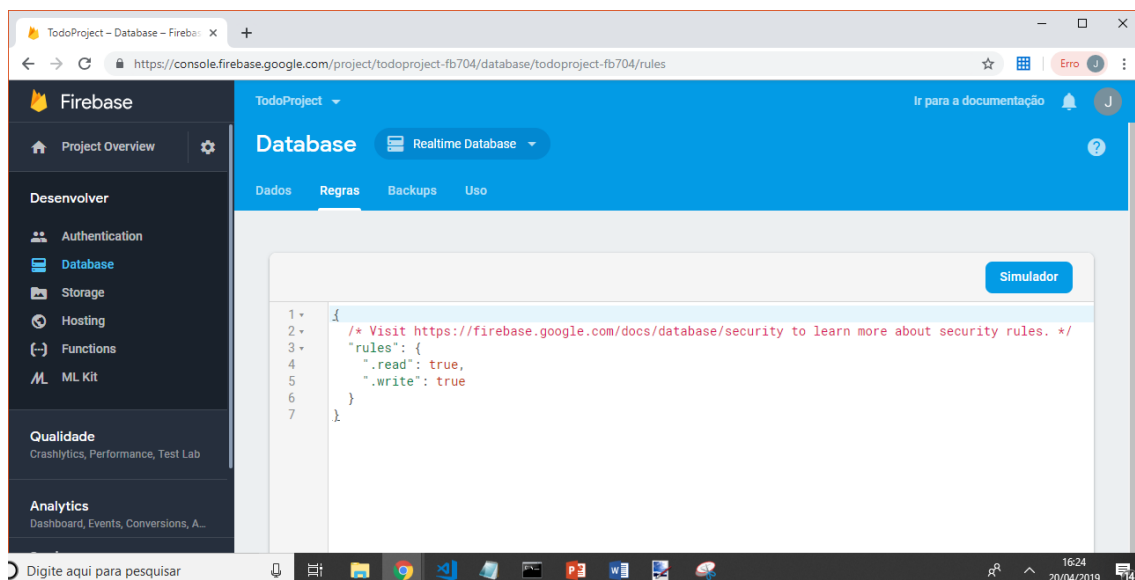
+ Adicionar campo

Cancelar

Salvar

8. Clicar em Salvar.

9. Por fim, alterar regras do “Realtime Database” para permitir leitura e escrita:



Criando Aplicação

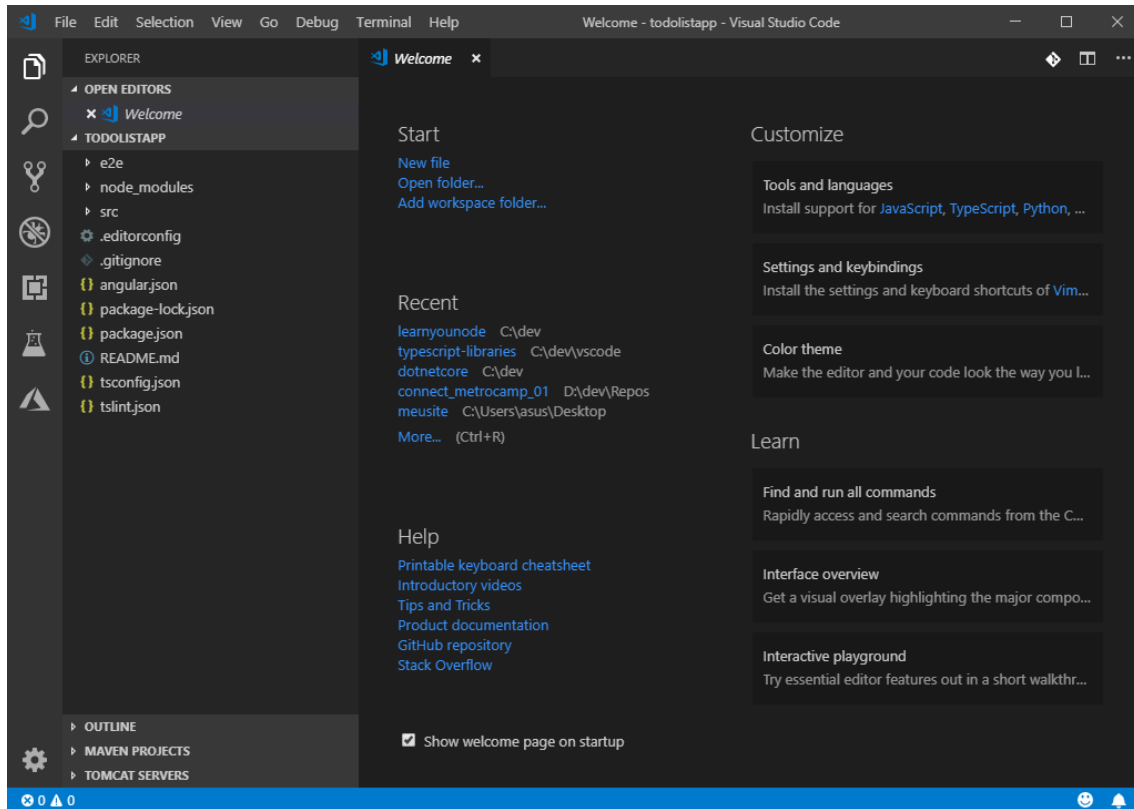
1. Acessar linha de comando.
2. Garantir que ng cli está instalado:

```
npm install -g @angular/cli
```

3. Acessar uma pasta onde irá criar a aplicação.
4. Criar a aplicação usando ng cli:

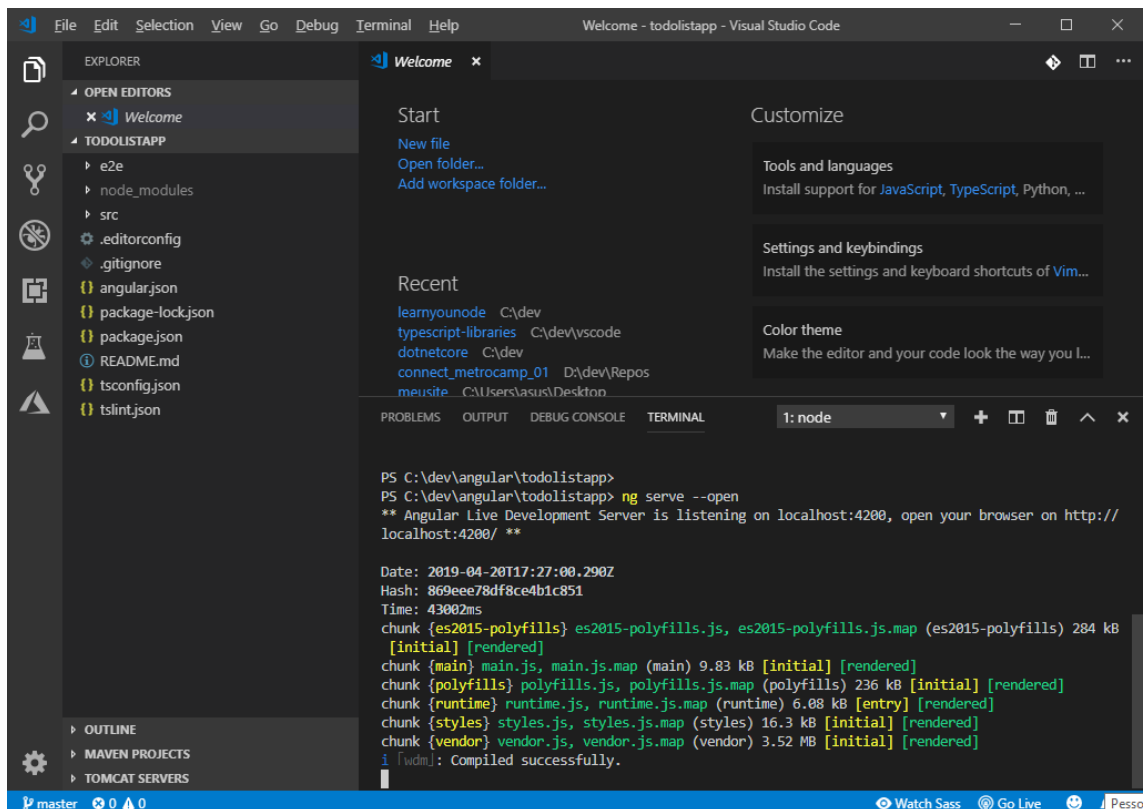
```
ng new todolistapp
```

5. Especificar as opções padrão.
6. Acessar a pasta recém criada no Visual Studio Code:



7. Para testar a aplicação, executar o seguinte comando via terminal do VSCode:

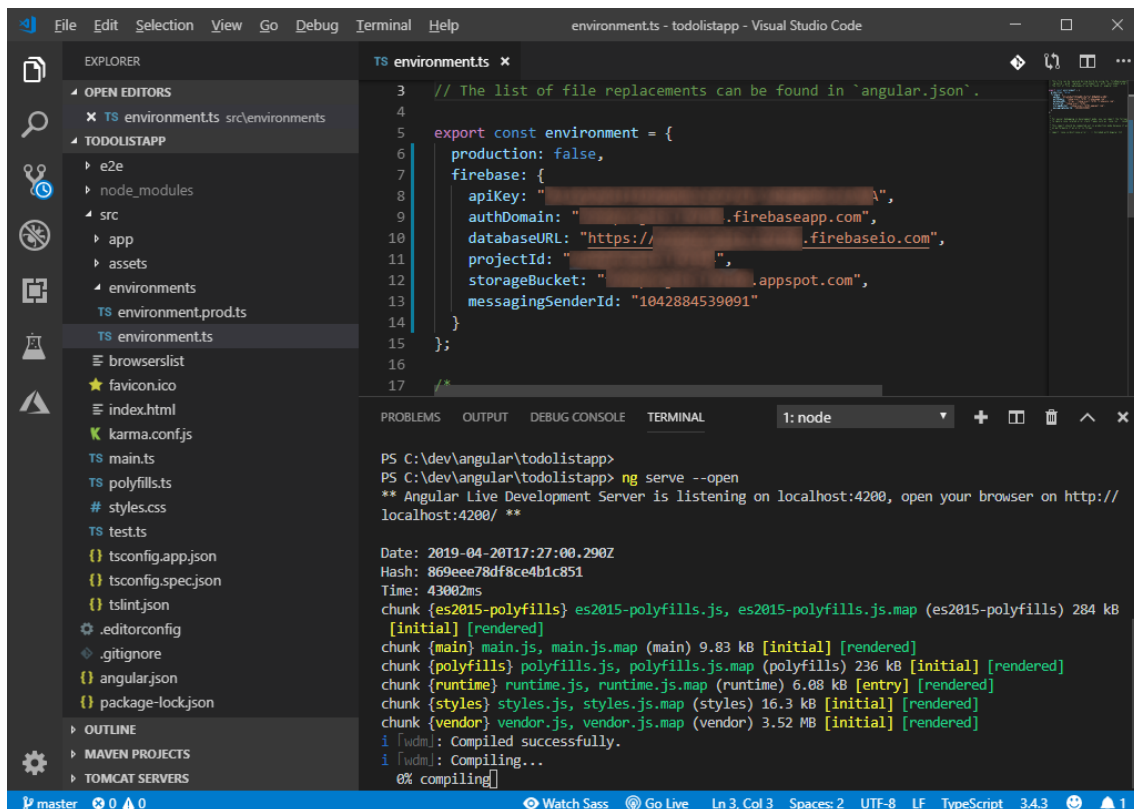
```
ng serve -open
```



8. Voltando para o Firebase, vamos copiar o código de acesso ao Banco de Dados:



9. Depois abra o arquivo src/environments/environments.ts e inclua a configuração em uma nova propriedade chamada firebase:



10. Interromper a execução do ng com Ctrl + C

11. Adicionar pacotes do firebase no projeto:

```
npm install firebase@5.10.0 @angular/fire@5.1.2 --save
```

12. Editar arquivo "src/app/app.module.ts" para incluir os dois pacotes do firebase logo após NgModule:

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireDatabaseModule } from '@angular/fire/database';
```

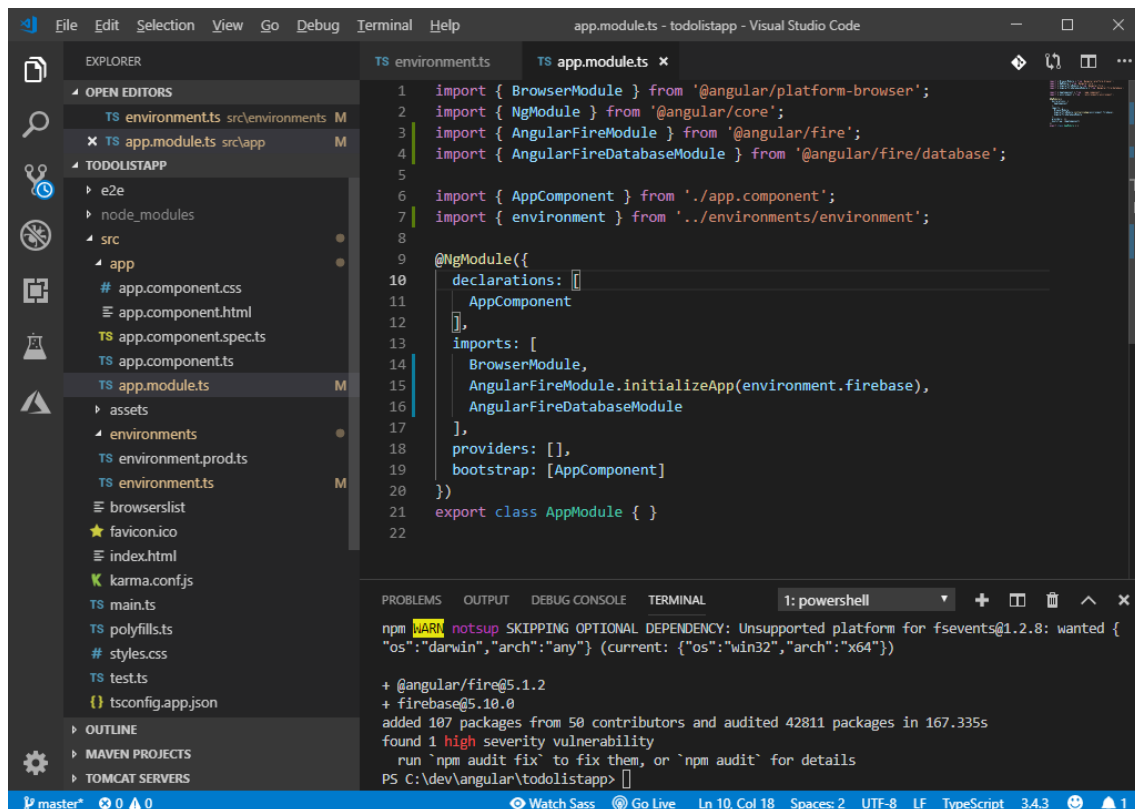
13. Depois importar o environment logo após AppComponent:

```
import { environment } from '../environments/environment';
```

14. Por fim, incluir imports dentro do NgModule, logo após BrowserModule:

```
AngularFireModule.initializeApp(environment.firebase),
AngularFireDatabaseModule
```

15. Concluído, o arquivo deverá parecer dessa forma:

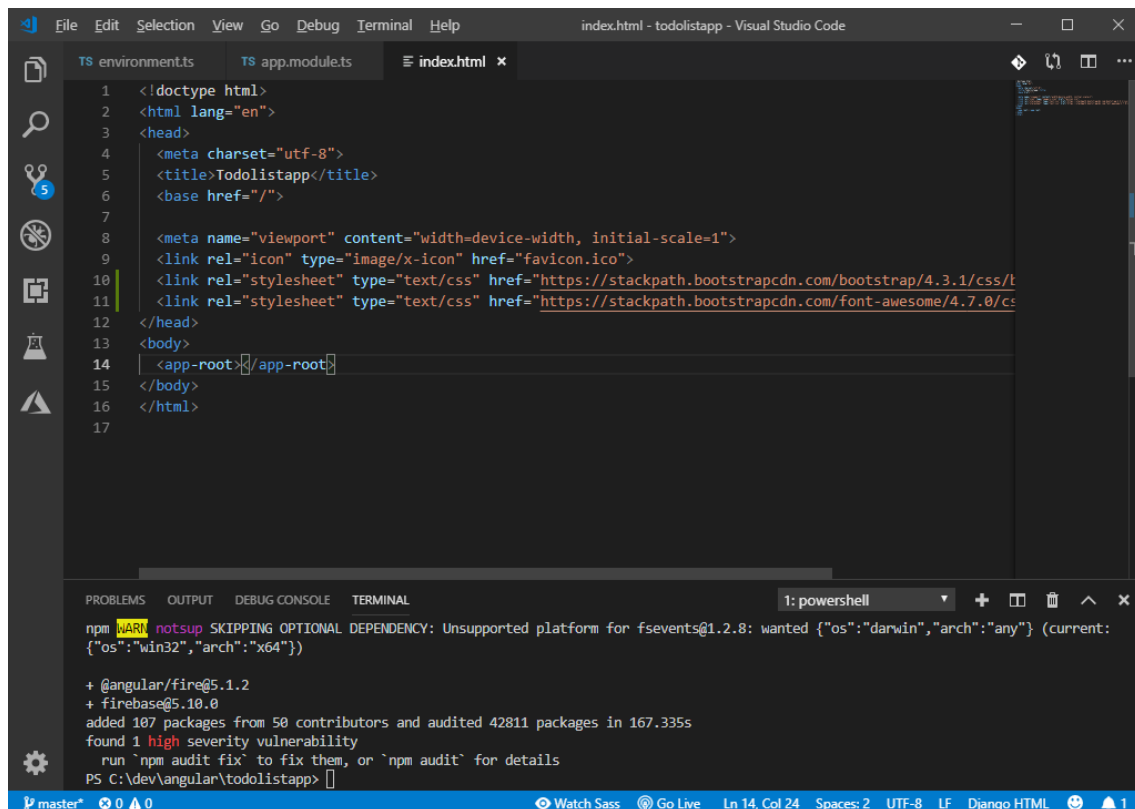


16. Agora está na hora de começar a incluir o código HTML para obter/alterar informações do banco.

17. Acesso o arquivo "src/index.html" e inclua os seguintes links CSS:

```
<link rel="stylesheet" type="text/css"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css"
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
```

18. No final, o arquivo ficará da seguinte forma:



The screenshot shows the Visual Studio Code interface. The editor window displays the `index.html` file for a project named `todolistapp`. The HTML structure includes a `<head>` section with meta tags for charset, title, base href, and viewport, as well as links for a favicon and Bootstrap CSS. The `<body>` section contains an `<app-root>` element. Below the editor, the TERMINAL panel is active, showing the output of an `npm install` command. It includes a warning about an unsupported platform for `fsevents@1.2.8` and reports that 167 packages were added from 50 contributors, with 1 high severity vulnerability found. The status bar at the bottom indicates the file is on the `master` branch, with 0 changes, and shows icons for Watch Sass, Go Live, and file encoding (UTF-8).

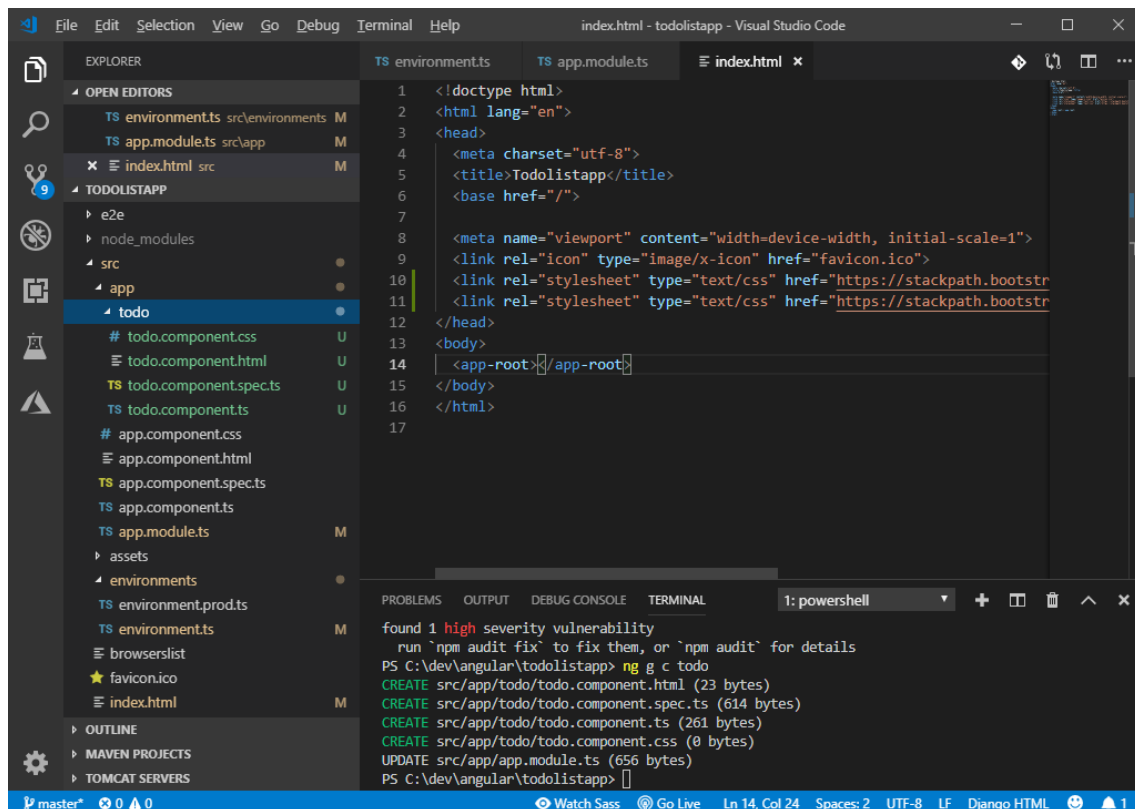
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>Todolistapp</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10  <link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/t
11  <link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/c
12 </head>
13 <body>
14   <app-root></app-root>
15 </body>
16 </html>
17
```

```
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.8: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ @angular/fire@5.1.2
+ firebase@5.10.0
added 167 packages from 50 contributors and audited 42811 packages in 167.335s
found 1 high severity vulnerability
run `npm audit fix` to fix them, or `npm audit` for details
PS C:\dev\angular\todolistapp>
```

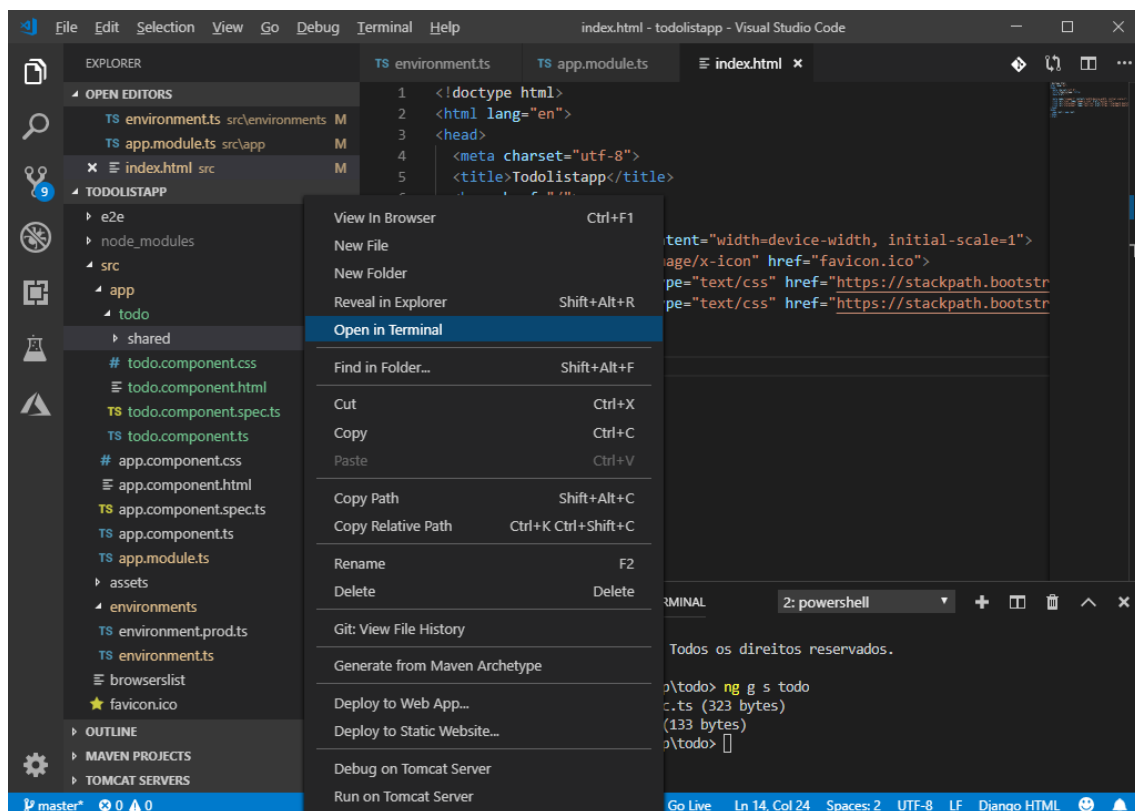
19. Agora vamos criar novo componente genérico Angular. Acesse o terminal de comando do VSCode e execute o seguinte comando:

```
ng g c todo
```

20. Ele cria o novo componente dentro da pasta "app":



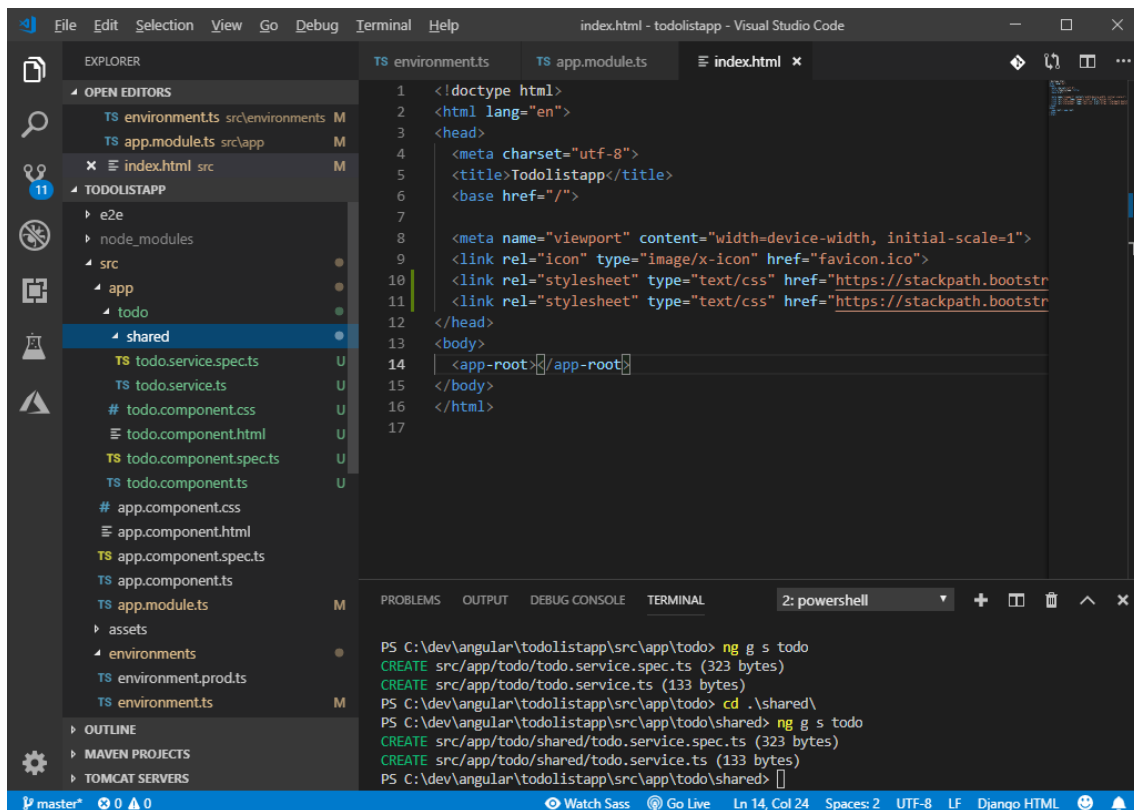
21. Dentro da pasta "todo", criar a pasta "shared" e abrir novo terminal a partir desta pasta:



22. Agora vamos criar novo serviço usando novamente linha de comando ng:

```
ng g s todo
```

23. Novos arquivos de serviço são criados:



24. Agora vamos incluir algumas regras CSS gerais em "src/styles.css" para definir o estilo do ponteiro quando o mouse estiver sobre os ícones de ação. Isto será usado depois pela aplicação:

```
.hover-cursor {  
  cursor: pointer;  
}
```

25. Incluir no arquivo "todo.service.ts" a importação dos módulos do firebase logo após Injectable:

```
import { AngularFireDatabase, AngularFireList } from  
'@angular/fire/database'
```

26. Especificar atributo todoList e, no construtor, o firebaseDb:

```
todoList: AngularFireList<any>;  
constructor(private firebaseDb: AngularFireDatabase) { }
```

27. Definir CRUD:

```
getTodoList() {  
  this.todoList = this.firebaseDb.list('todos');  
  return this.todoList;  
}
```

```

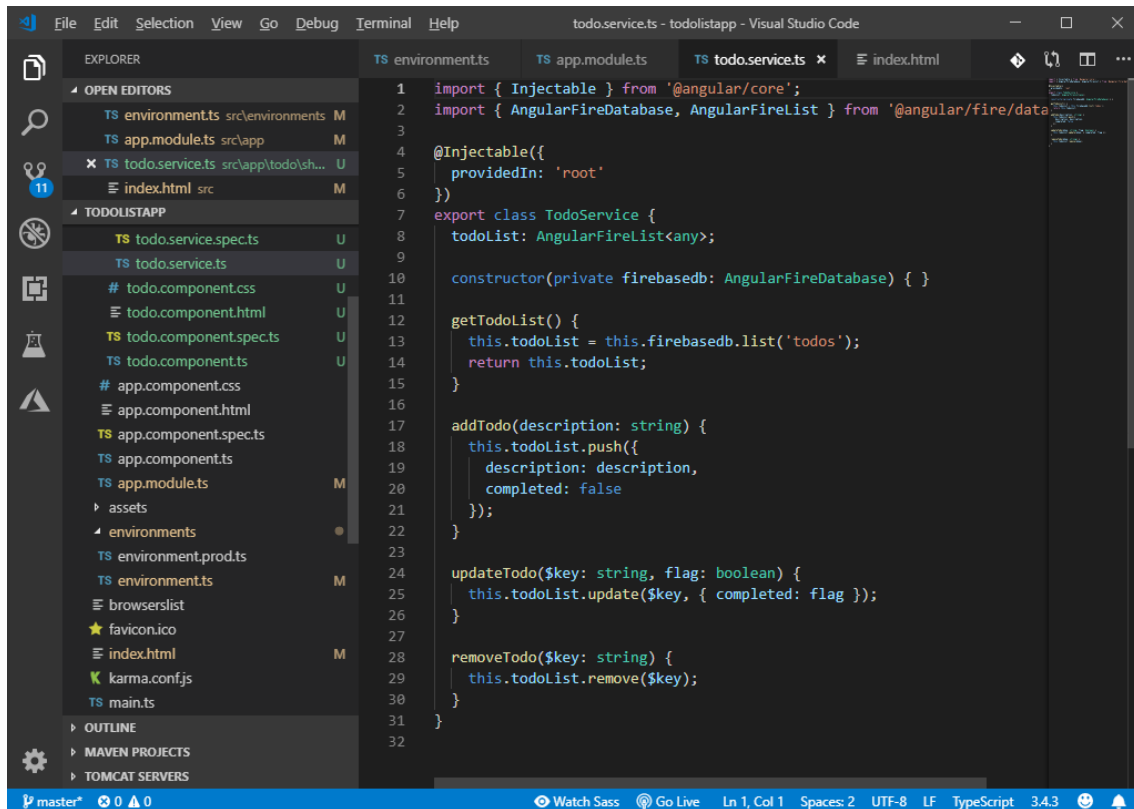
addTodo(description: string) {
  this.todoList.push({
    description: description,
    completed: false
  });
}

updateTodo($key: string, flag: boolean) {
  this.todoList.update($key, { completed: flag });
}

removeTodo($key: string) {
  this.todoList.remove($key);
}

```

28. No final o arquivo ficará dessa forma:



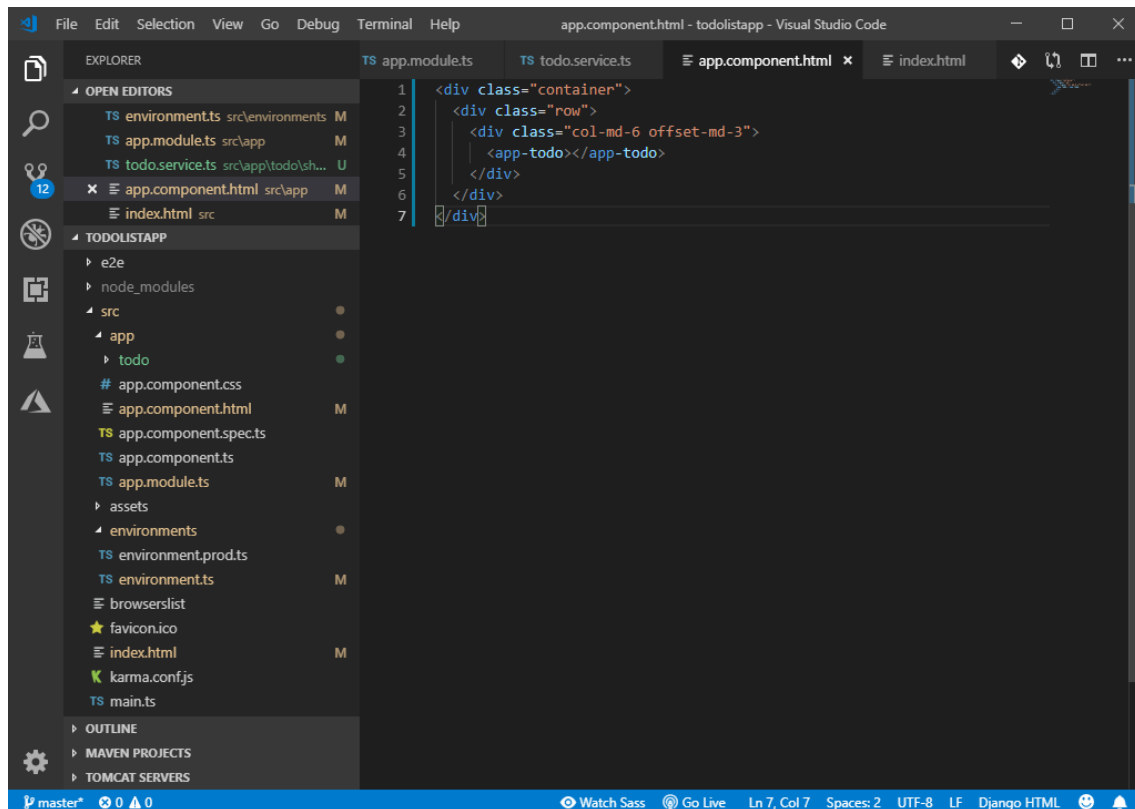
29. Agora, acessar o arquivo "src/app/app.component.html" e substituir o conteúdo incluindo no lugar o componente todo:

```

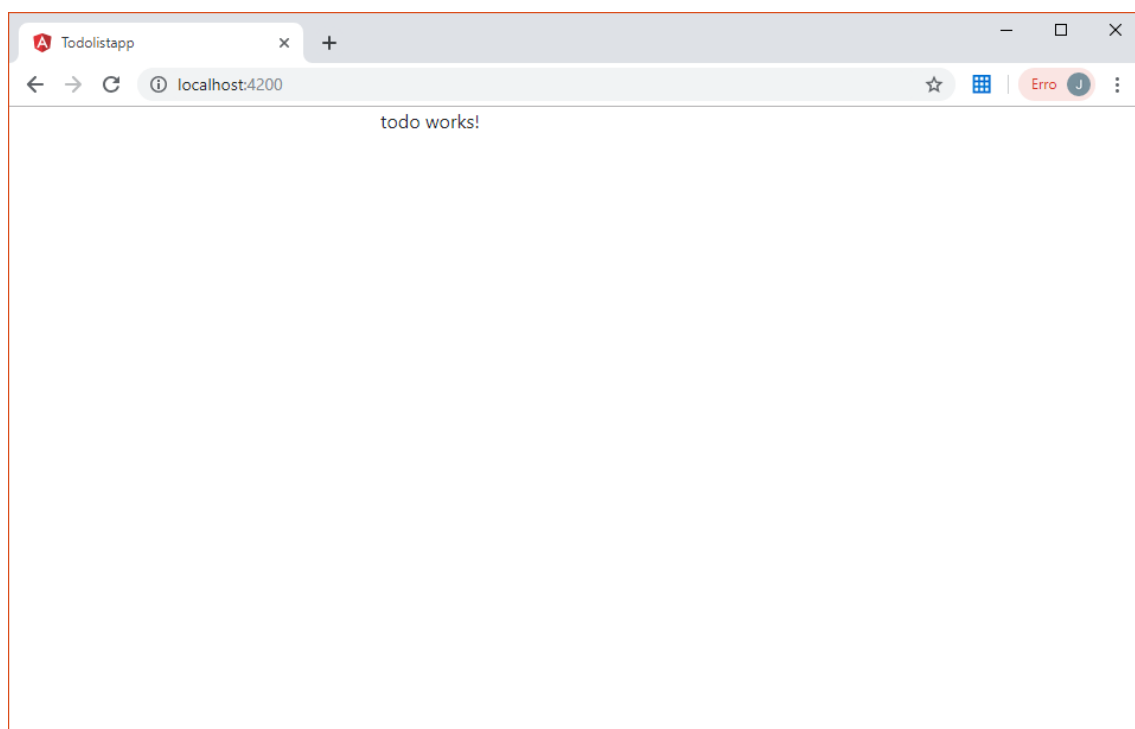
<div class="container">
  <div class="row">
    <div class="col-md-6 offset-md-3">
      <app-todo></app-todo>
    </div>
  </div>
</div>

```

```
</div>
```



30. Ao executar a aplicação, ele deve mostrar o componente:



31. Agora vamos alterar o componente "todo.component.ts":

- a. importar o serviço Todo

```
import { TodoService } from './shared/todo.service';
```

- b. Incluir o TodoService como Provider:

```
providers: [ TodoService ]
```

- c. Adicionar o TodoService (Injectable) no construtor:

```
constructor(private todoService: TodoService) { }
```

- d. Incluir o todoList como atributo:

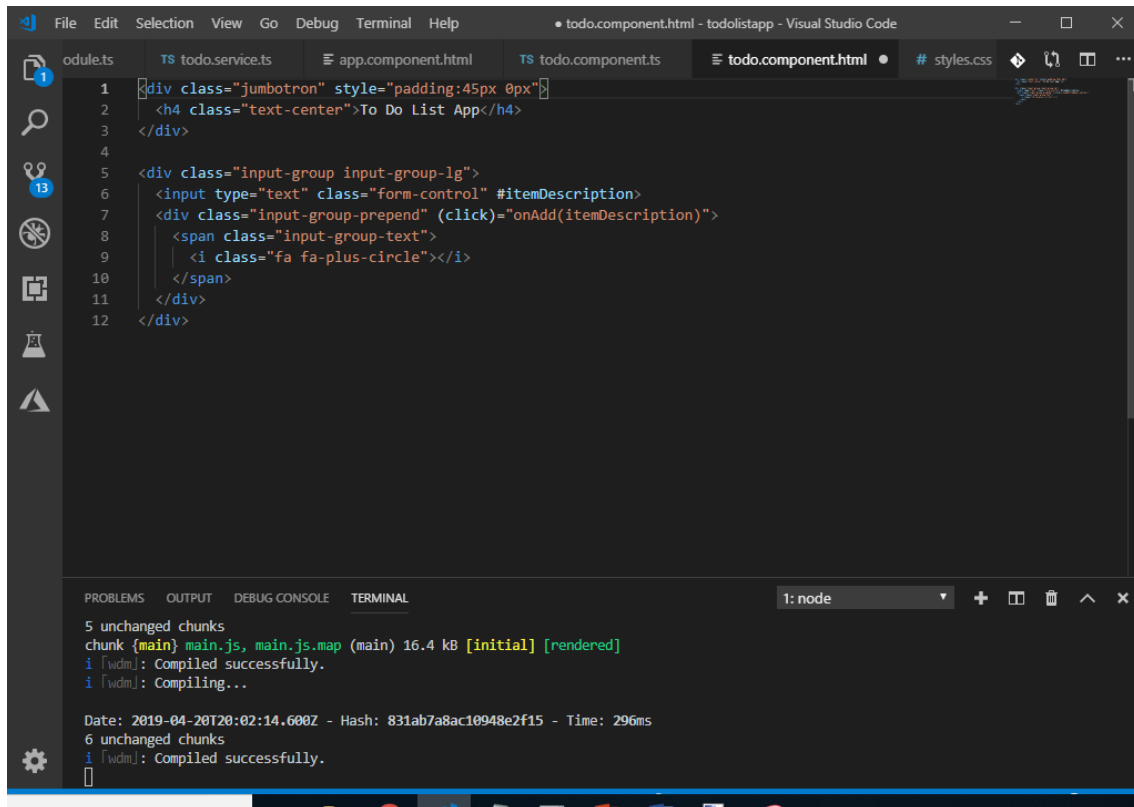
```
todoListArray: any[]
```

- e. No ngOnInit() carregar o todoList:

```
ngOnInit() {  
  this.todoService.getTodoList().snapshotChanges()  
    .subscribe(item => {  
    this.todoListArray = [];  
    item.forEach(element => {  
      var x = element.payload.toJSON();  
      x["$key"] = element.key;  
      this.todoListArray.push(x);  
    })  
  
    this.todoListArray.sort((a, b) => {  
      return a.completed - b.completed;  
    })  
  });  
}
```

32. Próximo passo é alterar o .html:

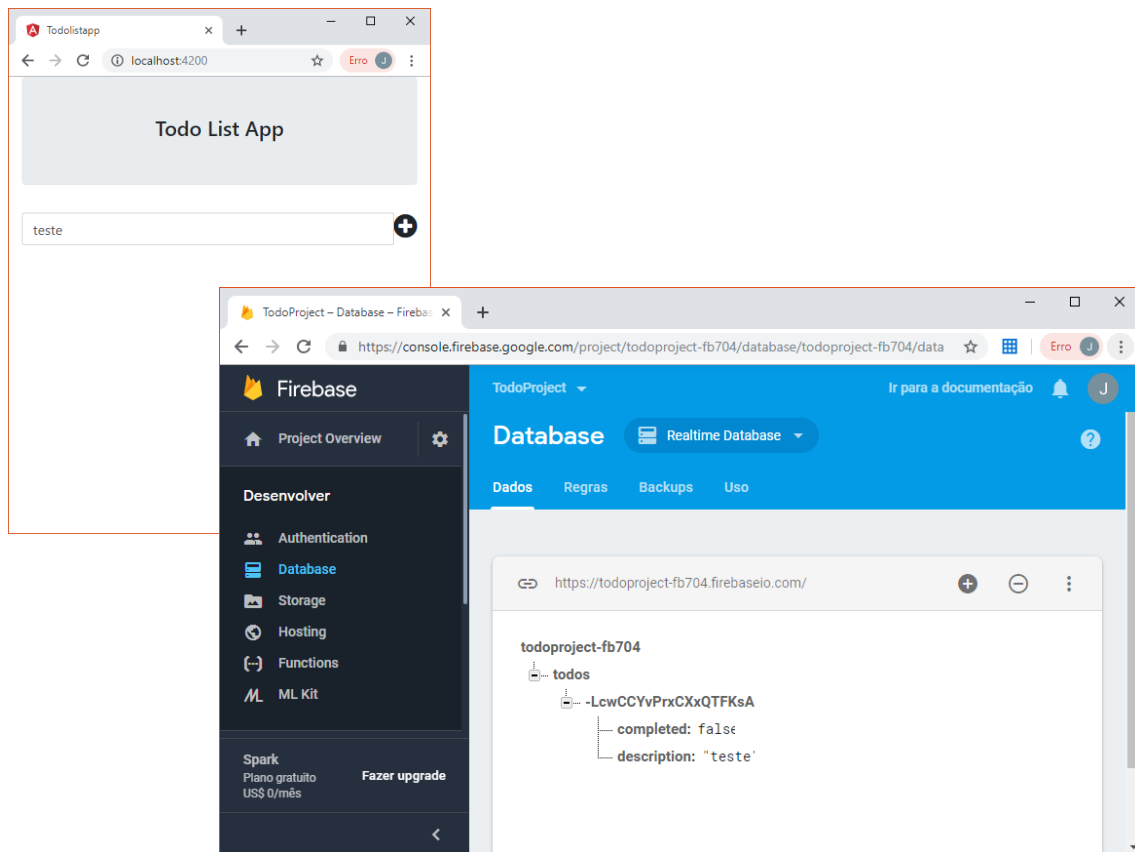
```
<div class="jumbotron" style="padding: 45px 0px">  
  <h4 class="text-center">Todo List App</h4>  
</div>  
  
<div class="input-group input-group-lg">  
  <input type="text" class="form-control" #itemDescription>  
  <div class="input-group-prepend" (click)="onAdd(itemDescription)">  
    <span class="input-group-text">  
      <i class="fa fa-plus-circle"></i>  
    </span>  
  </div>  
</div>
```



33. Note que incluímos a ação de clique para a função “onAdd”. Incluir essa função em todo.component.ts:

```
onAdd(itemDescription) {
  this.todoService.addTodo(itemDescription.value);
  itemDescription.value = null;
}
```

34. Feito isso, se testarmos agora, ele já estará inserindo novo item no Firebase:



35. Pronto! Agora vamos incluir a lista de itens Todo em todo.component.html:

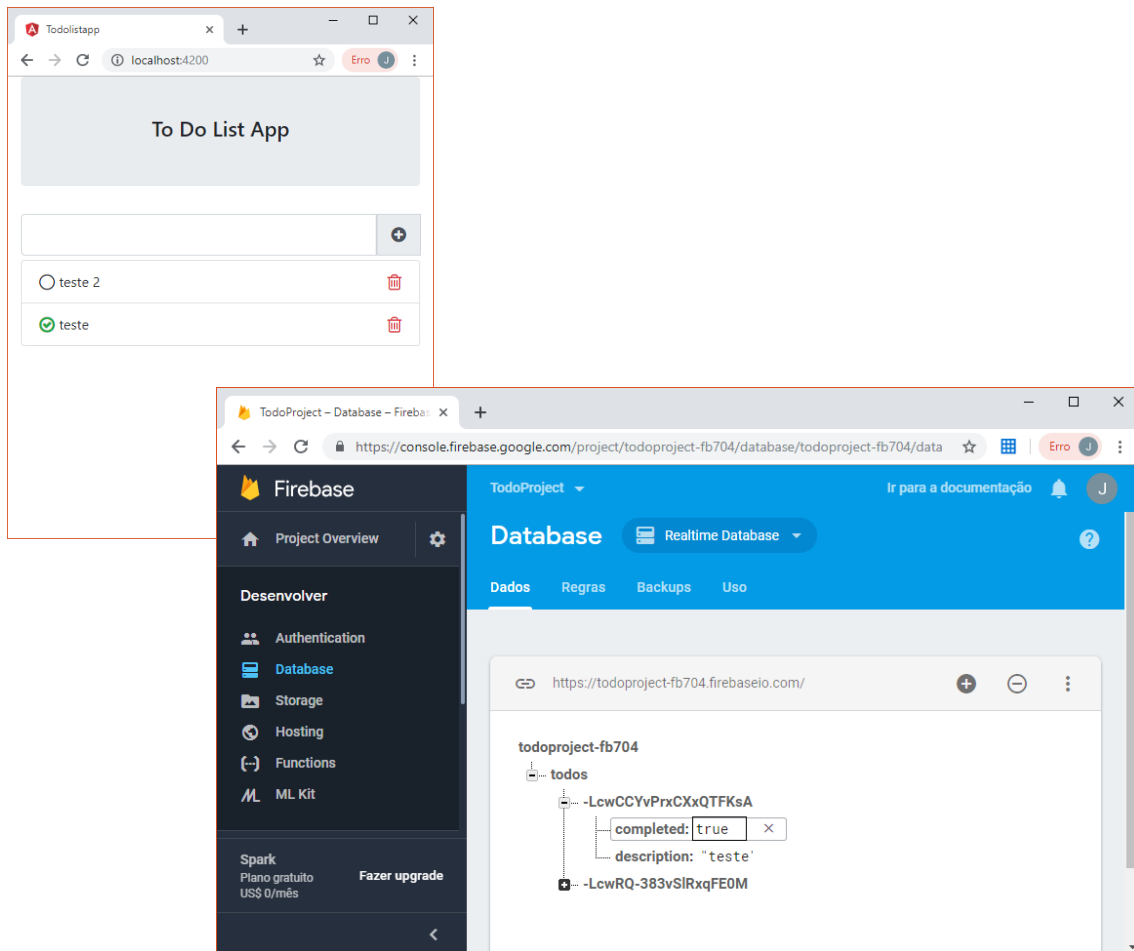
```
<div style="margin: 5px 0px">
  <ul class="list-group">
    <li class="list-group-item" *ngFor="let item of todoListArray">
      <span class="hover-cursor" [class.text-
success]="item.completed">
        <i class="fa fa-lg" [ngClass]="item.completed?'fa-check-
circle-o':'fa-circle-thin'"></i>
      </span>
      {{item.description}}
      <span class="hover-cursor text-danger pull-right">
        <i class="fa fa-trash-o fa-lg"></i>
      </span>
    </li>
  </ul>
</div>
```

36. Agora vamos tratar o check. Incluir na linha do todo list item o tratamento para a ação de click no todo.component.html:

```
<span class="hover-cursor" [class.text-success]="item.completed"
(click)="updateCompleted(item.$key, item.completed)">
```

37. Depois, incluir esta função em todo.component.ts:

```
updateCompleted($key: string, completed) {
  this.todoService.updateTodo($key, !completed);
}
```



38. Para finalizar, vamos incluir a ação de deleção. Alterar o todo.component.html para incluir a ação de deleção:

```
<span class="hover-cursor text-danger pull-right"
(click)="onDelete(item.$key)">
  <i class="fa fa-trash-o fa-lg"></i>
</span>
```

39. E incluir essa função em todo.component.ts:

```
onDelete($key: string) {
  this.todoService.removeTodo($key);
}
```

40. Agora ao clicar no botão de delete, a tarefa será excluída:

