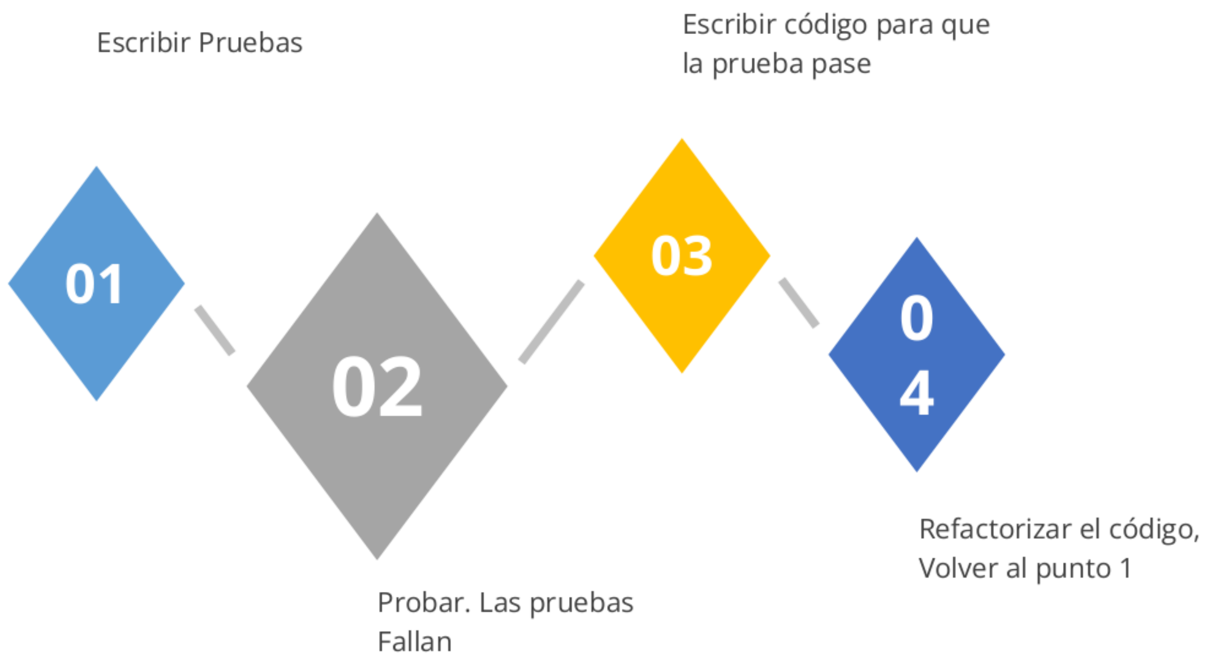
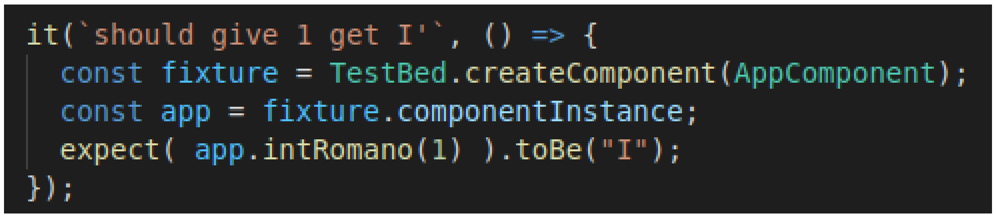
ESTUDIANTE: KADYHA PAZ GUTIERREZ
CC: 1020482429

METODOLOGÍA USADA


Escribir Pruebas
Escribir código para que la prueba pase
01
02
03
0 4
Refactorizar el código, Volver al punto 1
Probar. Las pruebas Fallan

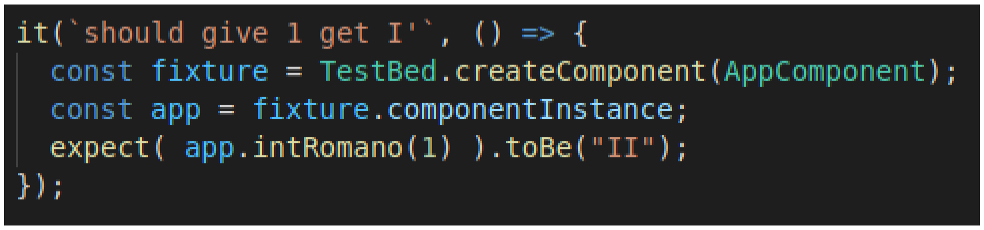
Repositorio con la aplicación: Agregar git

ESCRIBIR PRUEBA

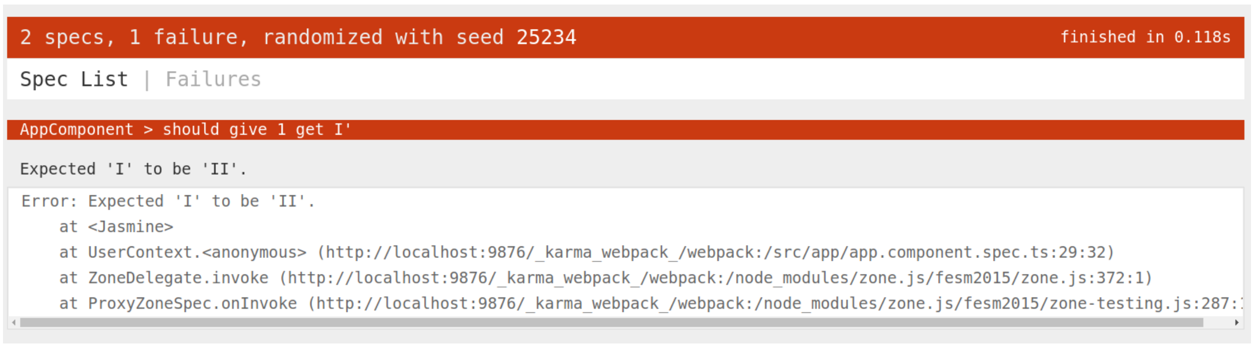


```
it(`should give 1 get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(1) ).toBe("I");
});
```

PROBAR LAS QUE FALLAN:

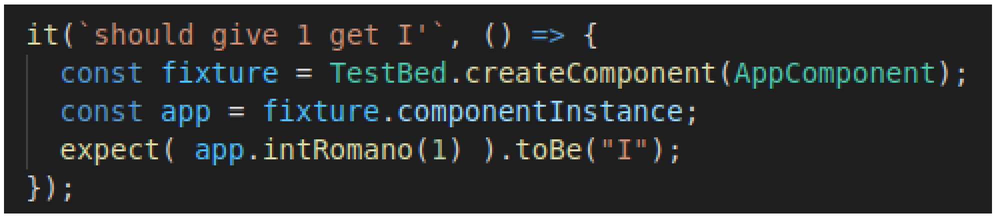


```
it(`should give 1 get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(1) ).toBe("II");
});
```

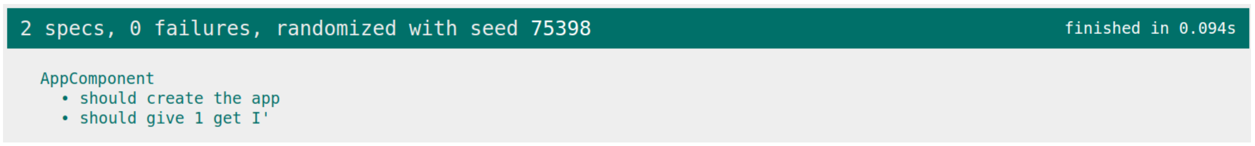


```
2 specs, 1 failure, randomized with seed 25234                                        finished in 0.118s
Spec List | Failures

AppComponent > should give 1 get I'

Expected 'I' to be 'II'.
Error: Expected 'I' to be 'II'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:29:32)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1)
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testing.js:287:
```

CÓDIGO PARA QUE PASE

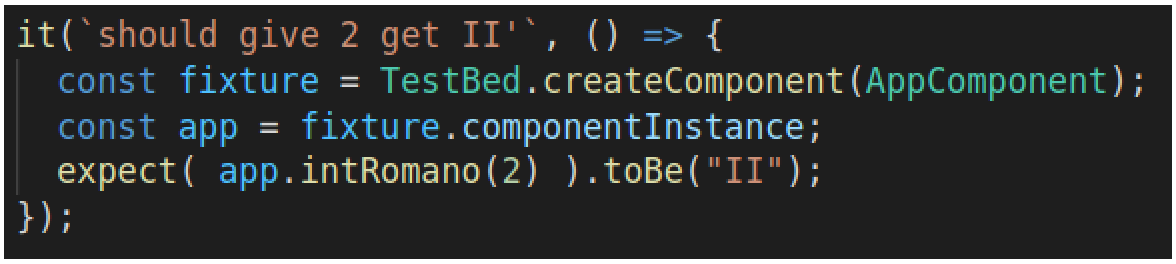


```
it(`should give 1 get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(1) ).toBe("I");
});
```



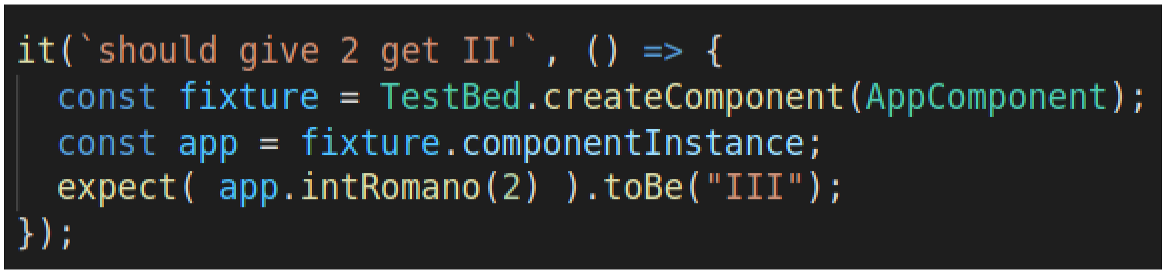
```
2 specs, 0 failures, randomized with seed 75398                                       finished in 0.094s

  AppComponent
    • should create the app
    • should give 1 get I'
```

ESCRIBIR LA PRUEBA

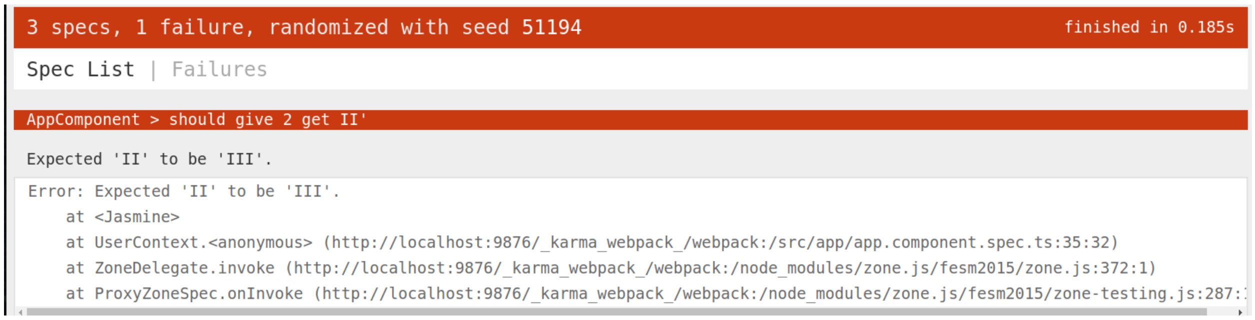


```
it(`should give 2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(2) ).toBe("II");
});
```

PROBAR LAS QUE FALLAN

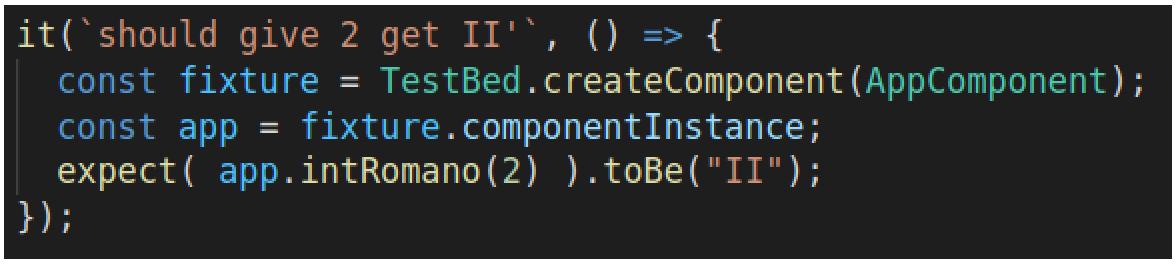


```
it(`should give 2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(2) ).toBe("III");
});
```

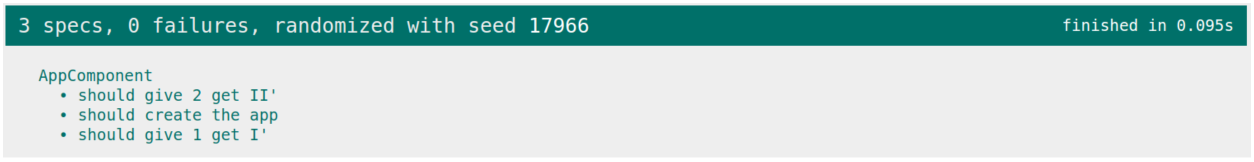


```
3 specs, 1 failure, randomized with seed 51194                    finished in 0.185s
Spec List | Failures

AppComponent > should give 2 get II'

Expected 'II' to be 'III'.

Error: Expected 'II' to be 'III'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:35:32)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1)
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testing.js:287:
```

CÓDIGO PARA QUE PASE:

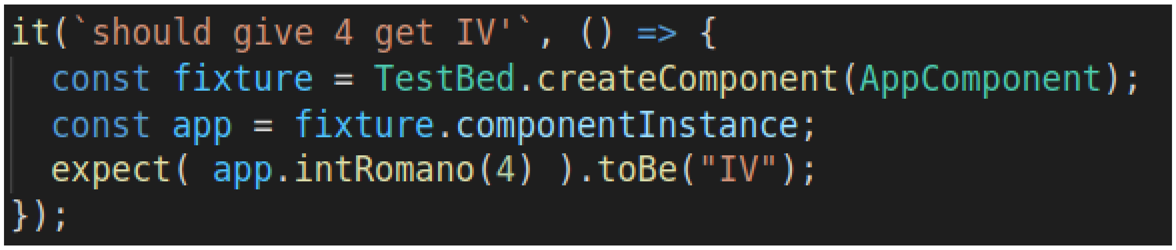


```
it(`should give 2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(2) ).toBe("II");
});
```



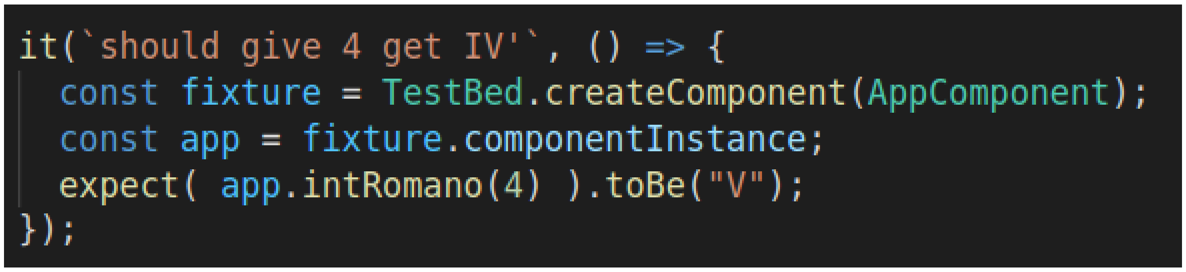
```
3 specs, 0 failures, randomized with seed 17966                   finished in 0.095s

  AppComponent
    • should give 2 get II'
    • should create the app
    • should give 1 get I'
```

ESCRIBIR LA PRUEBA

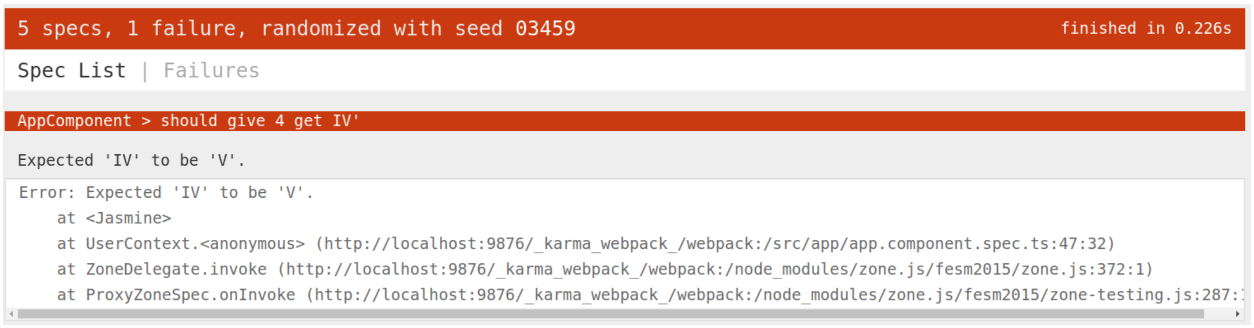


```
it(`should give 4 get IV'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(4) ).toBe("IV");
});
```

PROBAR LAS QUE FALLAN

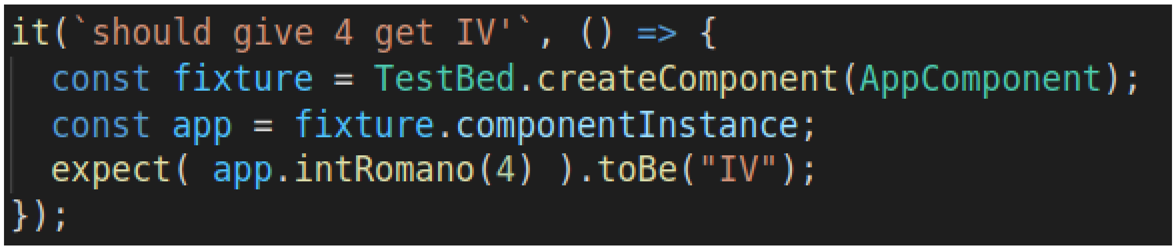


```
it(`should give 4 get IV'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(4) ).toBe("V");
});
```

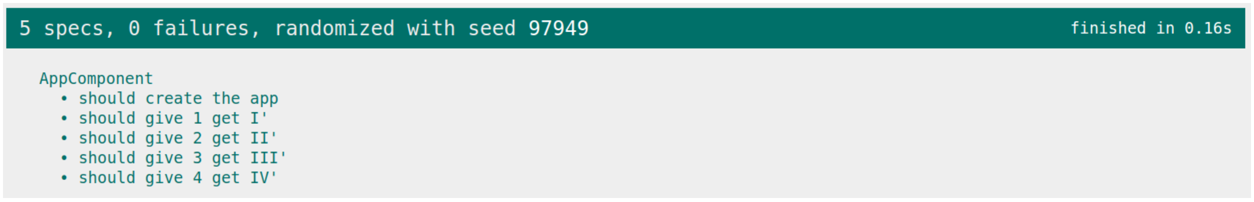


```
5 specs, 1 failure, randomized with seed 03459                    finished in 0.226s
Spec List | Failures
AppComponent > should give 4 get IV'
Expected 'IV' to be 'V'.
Error: Expected 'IV' to be 'V'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:47:32)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1)
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testing.js:287:
```

CÓDIGO PARA QUE PASE:



```
it(`should give 4 get IV'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect( app.intRomano(4) ).toBe("IV");
});
```



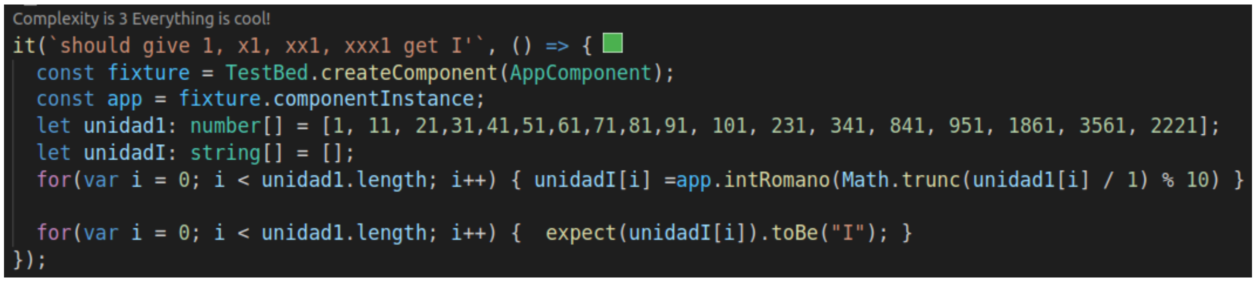
```
5 specs, 0 failures, randomized with seed 97949                    finished in 0.16s

  AppComponent
    • should create the app
    • should give 1 get I'
    • should give 2 get II'
    • should give 3 get III'
    • should give 4 get IV'
```

Este proceso se siguió realizando de forma iterativa.

ESCRIBIR LA PRUEBA

Luego de realizar varias pruebas unitarias se pensó en reducir las pruebas dividiendo el número ingresado, y evaluar las unidades, decenas, centenas y millar.

De esta forma en vez que hacer mil pruebas se deben hacer 33 pruebas, 9 para unidades, 10 para decenas, 10 para centenas y únicamente 3 para el millar, ya que no se pueden escribir números mayores del 4000. de esta forma se pueden evaluar los 3999 números posibles usando 33 pruebas con la siguiente estructura:

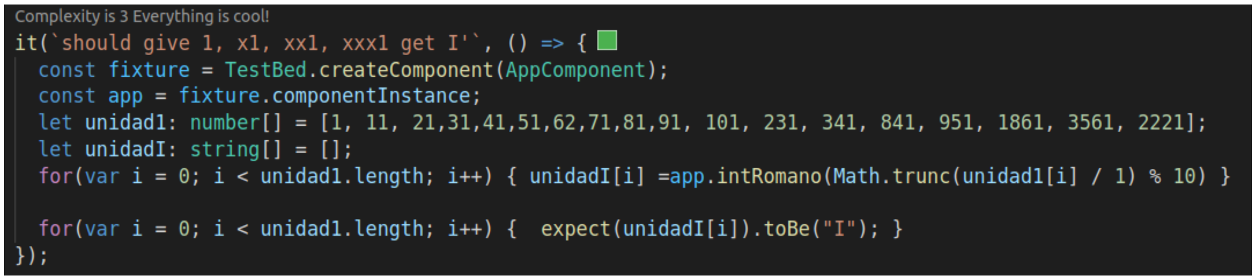


```
Complexity is 3 Everything is cool!
it(`should give 1, x1, xx1, xxx1 get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidad1: number[] = [1, 11, 21,31,41,51,61,71,81,91, 101, 231, 341, 841, 951, 1861, 3561, 2221];
  let unidadI: string[] = [];
  for(var i = 0; i < unidad1.length; i++) { unidadI[i] =app.intRomano(Math.trunc(unidad1[i] / 1) % 10) }

  for(var i = 0; i < unidad1.length; i++) {  expect(unidadI[i]).toBe("I"); }
});
```

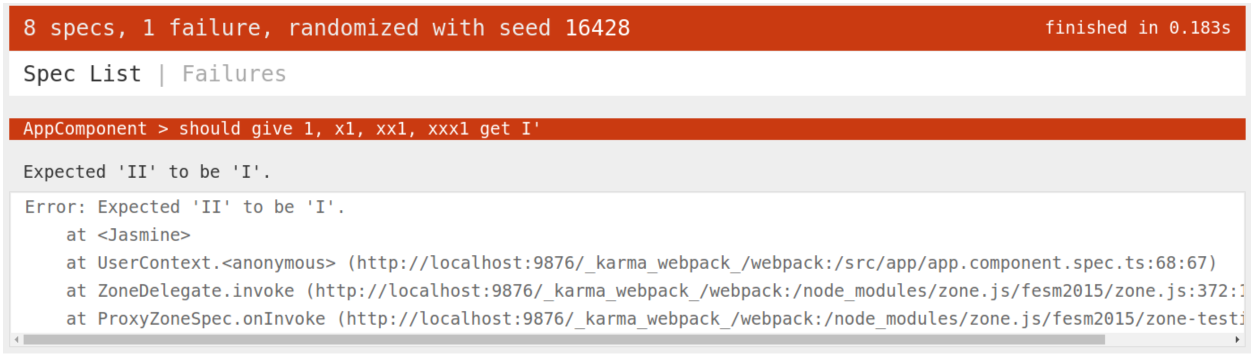
El código anterior sirve para probar los números que tienen un 1 en las unidades.

PROBAR LAS QUE FALLAN

Si se cambia el valor de la unidad la prueba debe fallar, ya que este es el único valor que se está tomando en cuenta. En este caso se cambió el 61 por 62.

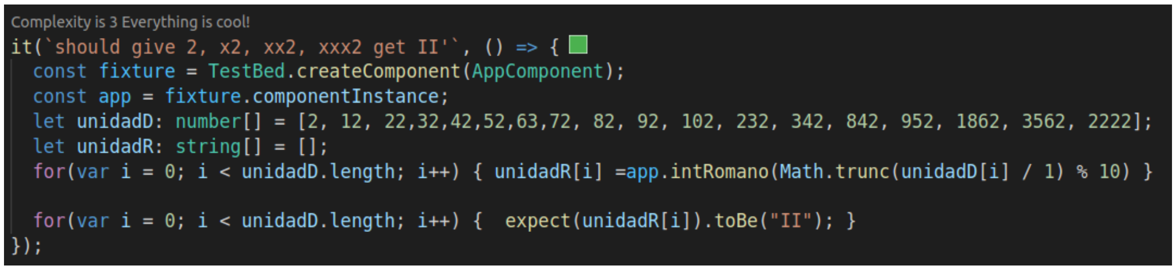


```
Complexity is 3 Everything is cool!
it(`should give 1, x1, xx1, xxx1 get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidad1: number[] = [1, 11, 21,31,41,51,62,71,81,91, 101, 231, 341, 841, 951, 1861, 3561, 2221];
  let unidadI: string[] = [];
  for(var i = 0; i < unidad1.length; i++) { unidadI[i] =app.intRomano(Math.trunc(unidad1[i] / 1) % 10) }

  for(var i = 0; i < unidad1.length; i++) {  expect(unidadI[i]).toBe("I"); }
});
```

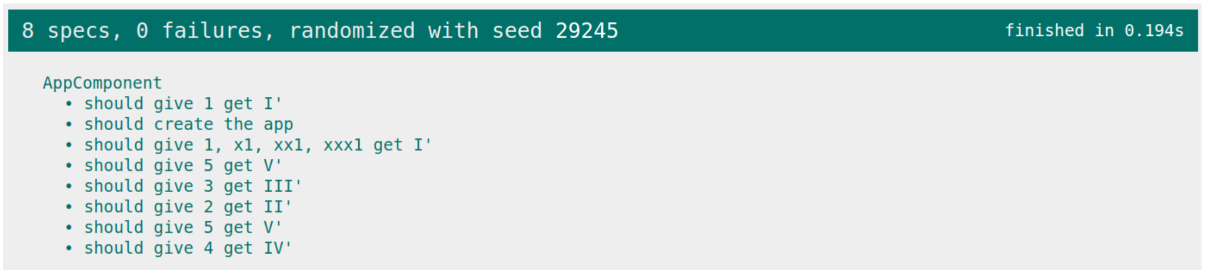


```
8 specs, 1 failure, randomized with seed 16428                                   finished in 0.183s
Spec List | Failures

AppComponent > should give 1, x1, xx1, xxx1 get I'

Expected 'II' to be 'I'.
Error: Expected 'II' to be 'I'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:68:67)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testi
```

CÓDIGO PARA QUE PASE:



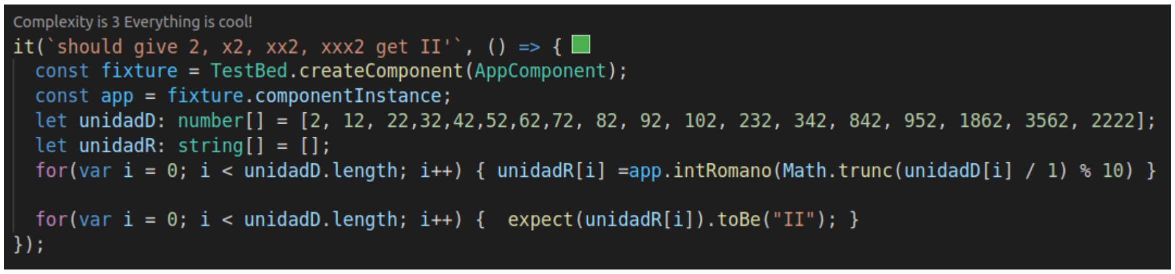
```
Complexity is 3 Everything is cool!
it(`should give 2, x2, xx2, xxx2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [2, 12, 22,32,42,52,63,72, 82, 92, 102, 232, 342, 842, 952, 1862, 3562, 2222];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("II"); }
});
```



```
8 specs, 0 failures, randomized with seed 29245                    finished in 0.194s

AppComponent
  • should give 1 get I'
  • should create the app
  • should give 1, x1, xx1, xxx1 get I'
  • should give 5 get V'
  • should give 3 get III'
  • should give 2 get II'
  • should give 5 get V'
  • should give 4 get IV'
```

ESCRIBIR LA PRUEBA

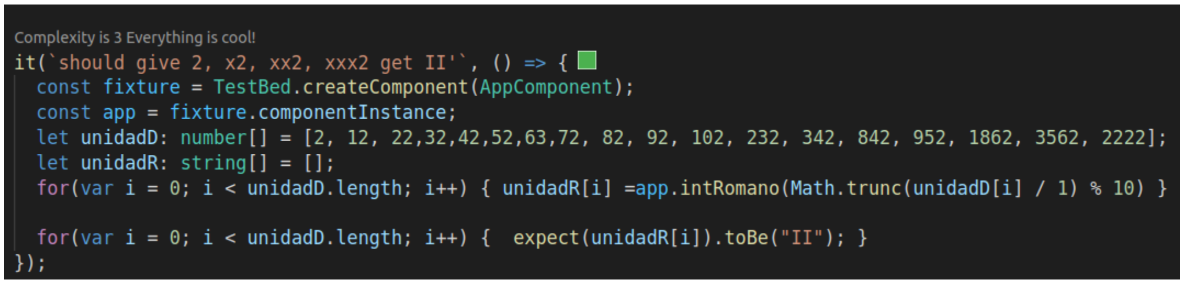
Ahora se evaluan los casos en los que la unidad es igual a 2.



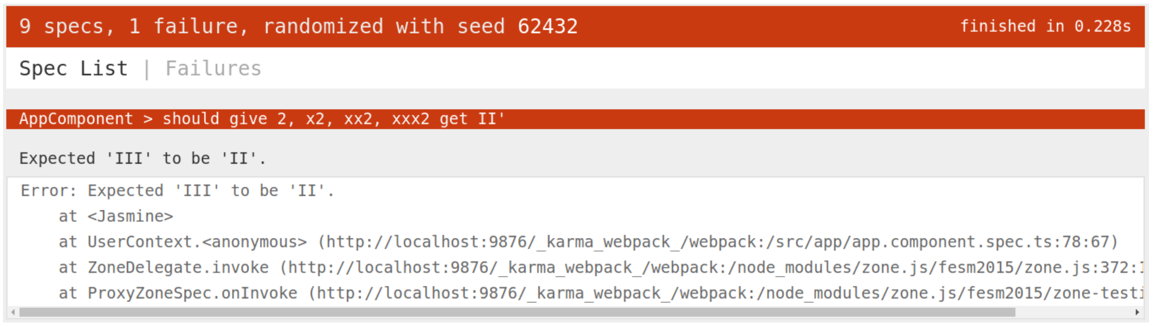
```
Complexity is 3 Everything is cool!
it(`should give 2, x2, xx2, xxx2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [2, 12, 22,32,42,52,62,72, 82, 92, 102, 232, 342, 842, 952, 1862, 3562, 2222];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("II"); }
});
```

PROBAR LAS QUE FALLAN

Para los casos que fallan se cambió el 62 por 63.

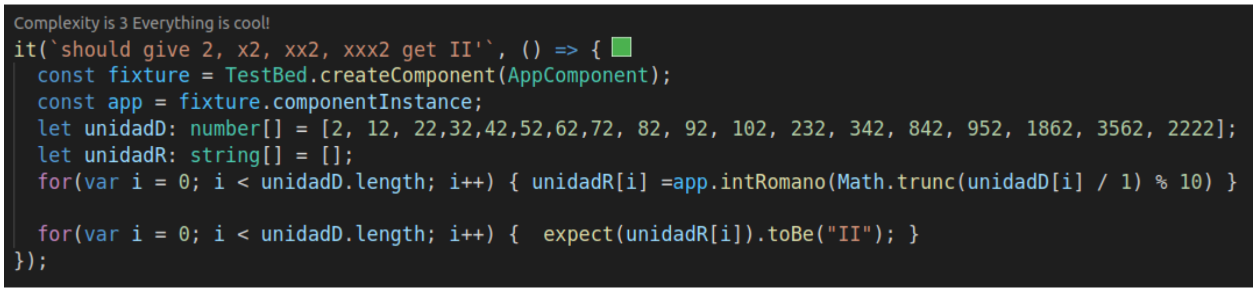


```
Complexity is 3 Everything is cool!
it(`should give 2, x2, xx2, xxx2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [2, 12, 22,32,42,52,63,72, 82, 92, 102, 232, 342, 842, 952, 1862, 3562, 2222];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("II"); }
});
```

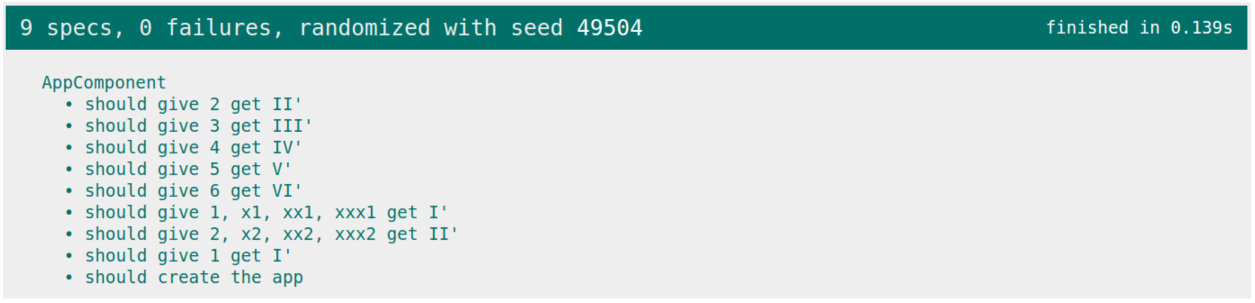


```
9 specs, 1 failure, randomized with seed 62432                     finished in 0.228s
Spec List | Failures

AppComponent > should give 2, x2, xx2, xxx2 get II'

Expected 'III' to be 'II'.

Error: Expected 'III' to be 'II'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:78:67)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testi
```

CÓDIGO PARA QUE PASE:



```
Complexity is 3 Everything is cool!
it(`should give 2, x2, xx2, xxx2 get II'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [2, 12, 22,32,42,52,62,72, 82, 92, 102, 232, 342, 842, 952, 1862, 3562, 2222];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("II"); }
});
```

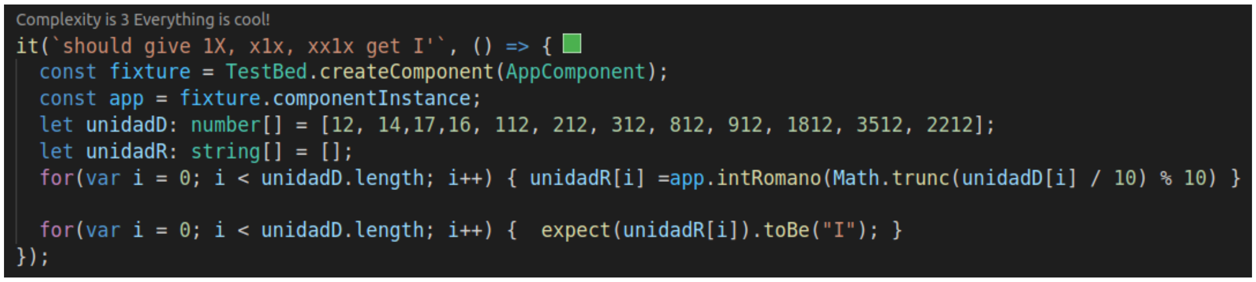


```
9 specs, 0 failures, randomized with seed 49504                    finished in 0.139s

  AppComponent
    • should give 2 get II'
    • should give 3 get III'
    • should give 4 get IV'
    • should give 5 get V'
    • should give 6 get VI'
    • should give 1, x1, xx1, xxx1 get I'
    • should give 2, x2, xx2, xxx2 get II'
    • should give 1 get I'
    • should create the app
```

Este desarrollo se puede hacer de forma iterativa para las 9 posibles unidades, y toda la lista de números enteros que se encuentran entre 1 y 3999 para cada caso. por ejemplo en este caso, la lista sería de los números terminados en 2.

ESCRIBIR LA PRUEBA

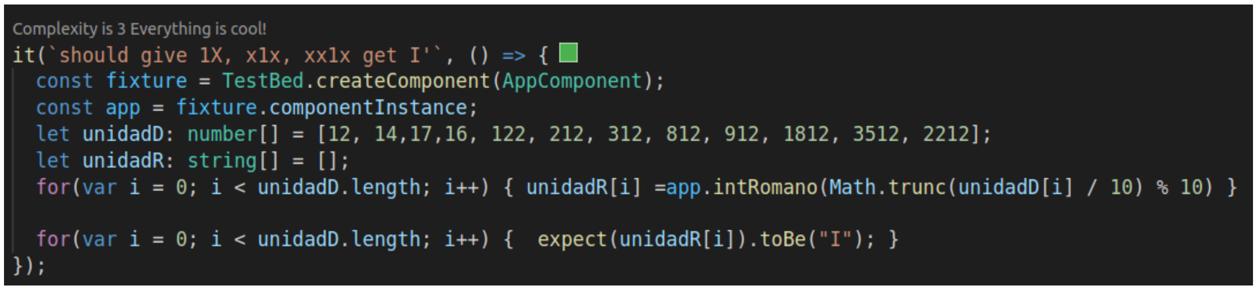
Para evaluar las decenas se usan los mismos códigos anteriores, pero con una lista diferente y ahora se hace una división por 10 y se saca el módulo 10, en el caso anterior se dividía por 1 y se saca igualmente módulo 10. Ya que esta prueba es tan similar se pueden copiar las 9 anteriores cambiando el valor de la división por 10 en lugar de uno, y la lista de números que se va a evaluar.



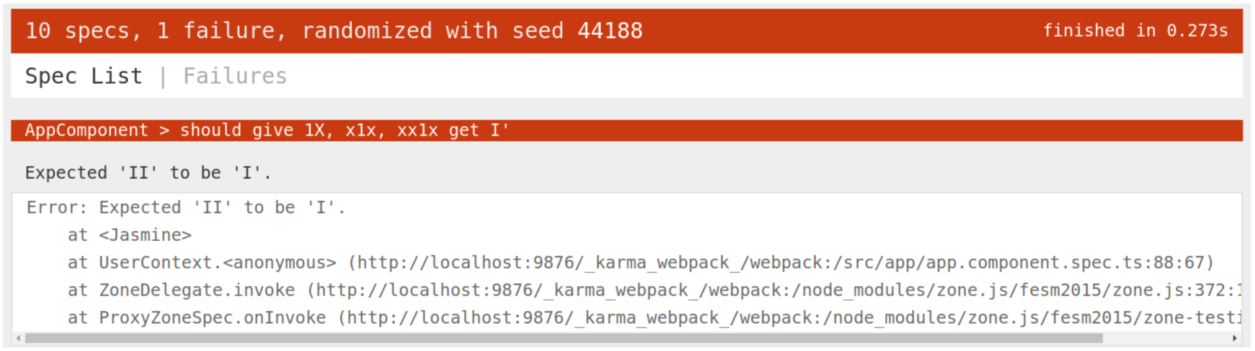
```
Complexity is 3 Everything is cool!
it(`should give 1X, x1x, xx1x get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [12, 14,17,16, 112, 212, 312, 812, 912, 1812, 3512, 2212];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 10) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

PROBAR LAS QUE FALLAN

Para los casos que fallan se cambia un 1 de las decenas.

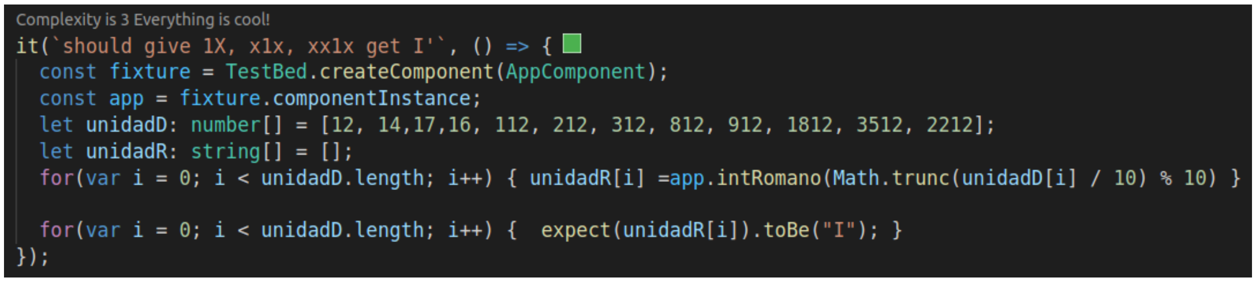


```
Complexity is 3 Everything is cool!
it(`should give 1X, x1x, xx1x get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [12, 14,17,16, 122, 212, 312, 812, 912, 1812, 3512, 2212];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 10) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

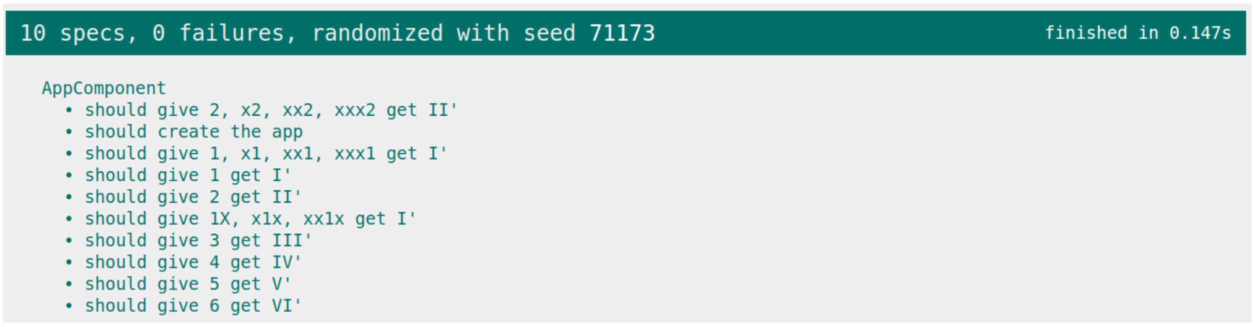


```
10 specs, 1 failure, randomized with seed 44188                                    finished in 0.273s
Spec List | Failures

AppComponent > should give 1X, x1x, xx1x get I'

Expected 'II' to be 'I'.
Error: Expected 'II' to be 'I'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:88:67)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testi
```

CÓDIGO PARA QUE PASE:



```
Complexity is 3 Everything is cool!
it(`should give 1X, x1x, xx1x get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [12, 14,17,16, 112, 212, 312, 812, 912, 1812, 3512, 2212];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 10) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

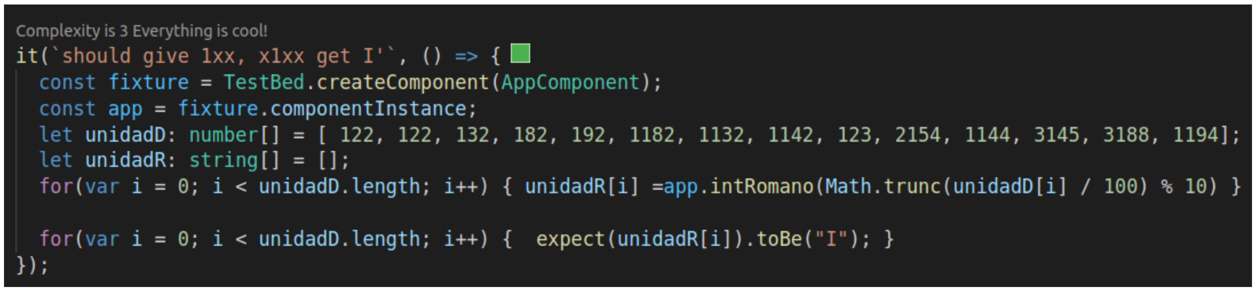


```
10 specs, 0 failures, randomized with seed 71173                                   finished in 0.147s

  AppComponent
    • should give 2, x2, xx2, xxx2 get II'
    • should create the app
    • should give 1, x1, xx1, xxx1 get I'
    • should give 1 get I'
    • should give 2 get II'
    • should give 1X, x1x, xx1x get I'
    • should give 3 get III'
    • should give 4 get IV'
    • should give 5 get V'
    • should give 6 get VI'
```

Este proceso se puede realizar de forma iterativa para las 10 posibles decenas.

ESCRIBIR LA PRUEBA

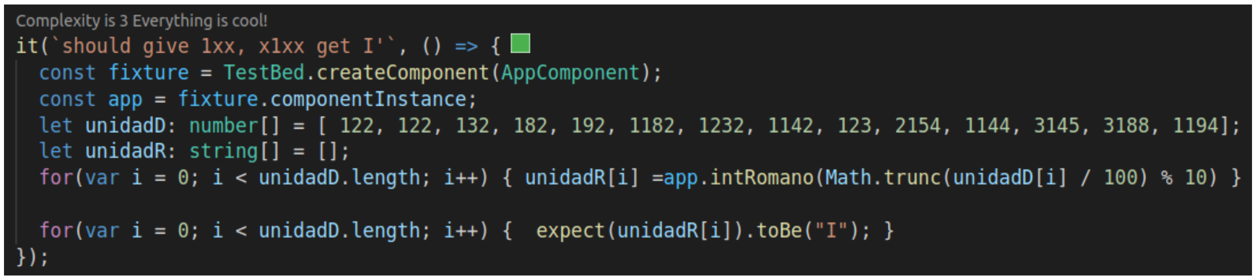
Para evaluar las centenas igual que en el caso anterior se usan los mismos códigos anteriores, pero con una lista diferente y ahora se hace una división por 100.



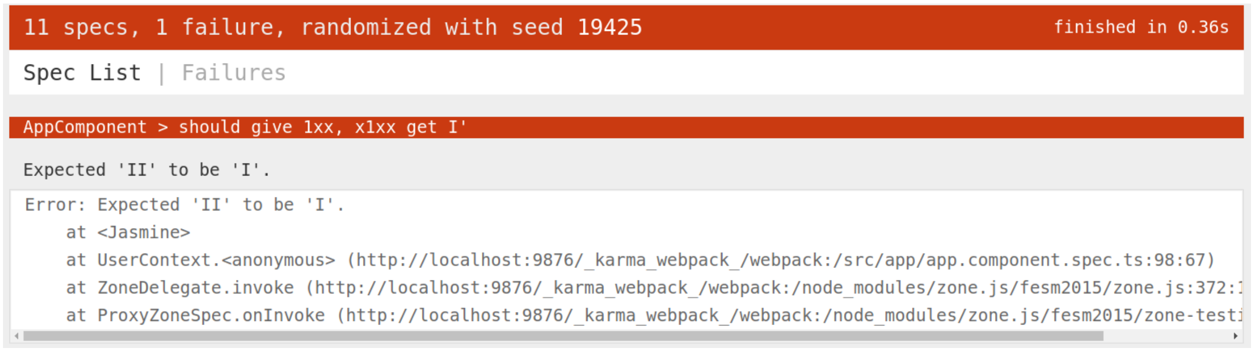
```
Complexity is 3 Everything is cool!
it(`should give 1xx, x1xx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 122, 122, 132, 182, 192, 1182, 1132, 1142, 123, 2154, 1144, 3145, 3188, 1194];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 100) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

PROBAR LAS QUE FALLAN

Para los casos que fallan se cambia el valor de las centenas.

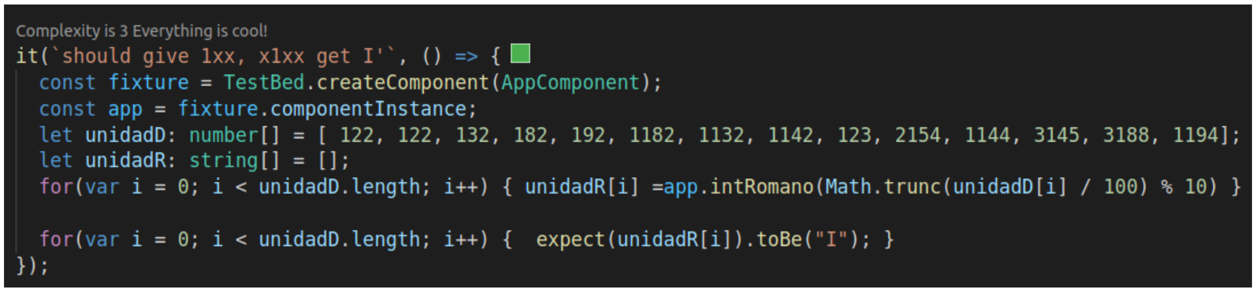


```
Complexity is 3 Everything is cool!
it(`should give 1xx, x1xx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 122, 122, 132, 182, 192, 1182, 1232, 1142, 123, 2154, 1144, 3145, 3188, 1194];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 100) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

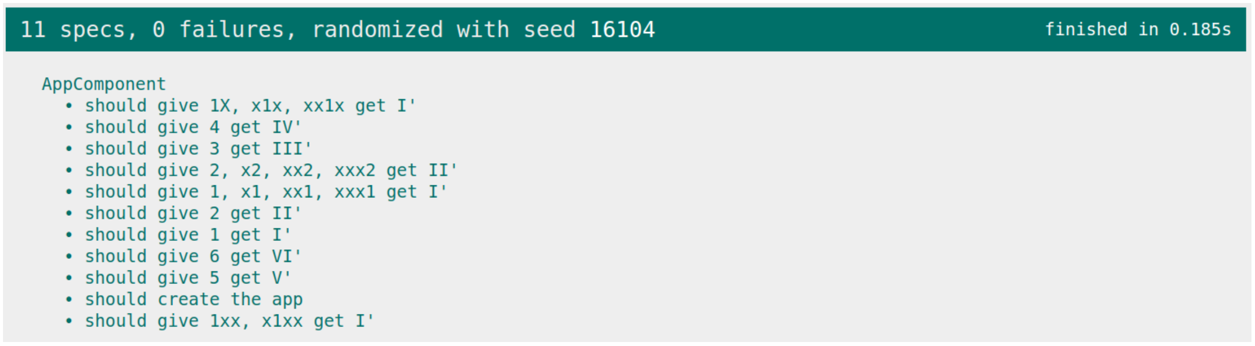


```
11 specs, 1 failure, randomized with seed 19425                                finished in 0.36s
Spec List | Failures

AppComponent > should give 1xx, x1xx get I'

Expected 'II' to be 'I'.

Error: Expected 'II' to be 'I'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:98:67)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testi
```

CÓDIGO PARA QUE PASE:



```
Complexity is 3 Everything is cool!
it(`should give 1xx, x1xx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 122, 122, 132, 182, 192, 1182, 1132, 1142, 123, 2154, 1144, 3145, 3188, 1194];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 100) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

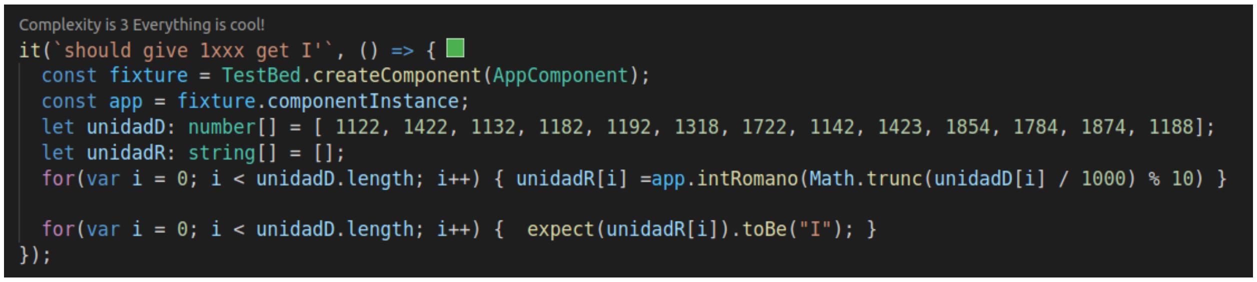


```
11 specs, 0 failures, randomized with seed 16104                              finished in 0.185s

  AppComponent
    • should give 1X, x1x, xx1x get I'
    • should give 4 get IV'
    • should give 3 get III'
    • should give 2, x2, xx2, xxx2 get II'
    • should give 1, x1, xx1, xxx1 get I'
    • should give 2 get II'
    • should give 1 get I'
    • should give 6 get VI'
    • should give 5 get V'
    • should create the app
    • should give 1xx, x1xx get I'
```

Este proceso se puede realizar de forma iterativa para las 10 posibles centenas.

ESCRIBIR LA PRUEBA

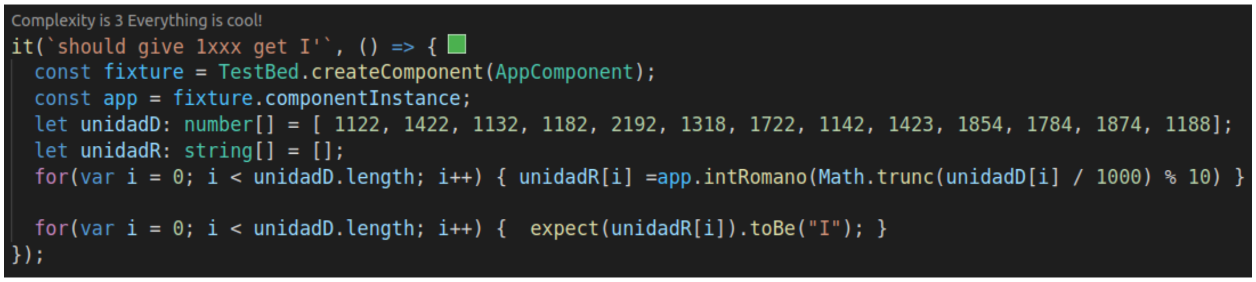
Para evaluar el millar se hace lo mismo que el caso anterior, pero ahora solo es necesario hacer 3 códigos más, uno para 1xxx otro para 2xxx y otro para 3xxx.



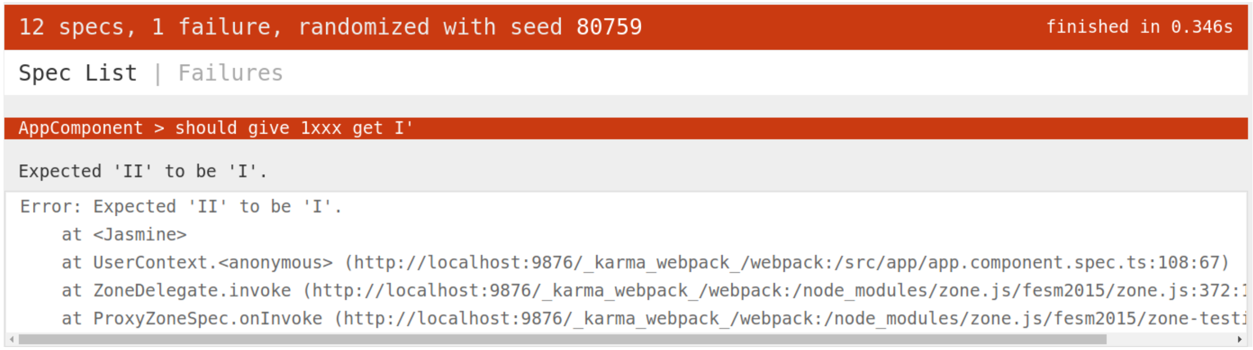
```
Complexity is 3 Everything is cool!
it(`should give 1xxx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 1122, 1422, 1132, 1182, 1192, 1318, 1722, 1142, 1423, 1854, 1784, 1874, 1188];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1000) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

PROBAR LAS QUE FALLAN

Para los casos que fallan se cambia el valor del millar.

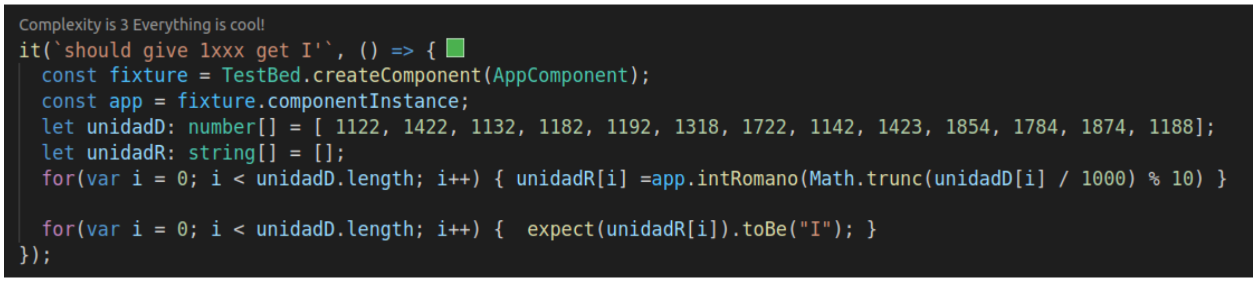


```
Complexity is 3 Everything is cool!
it(`should give 1xxx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 1122, 1422, 1132, 1182, 2192, 1318, 1722, 1142, 1423, 1854, 1784, 1874, 1188];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1000) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```

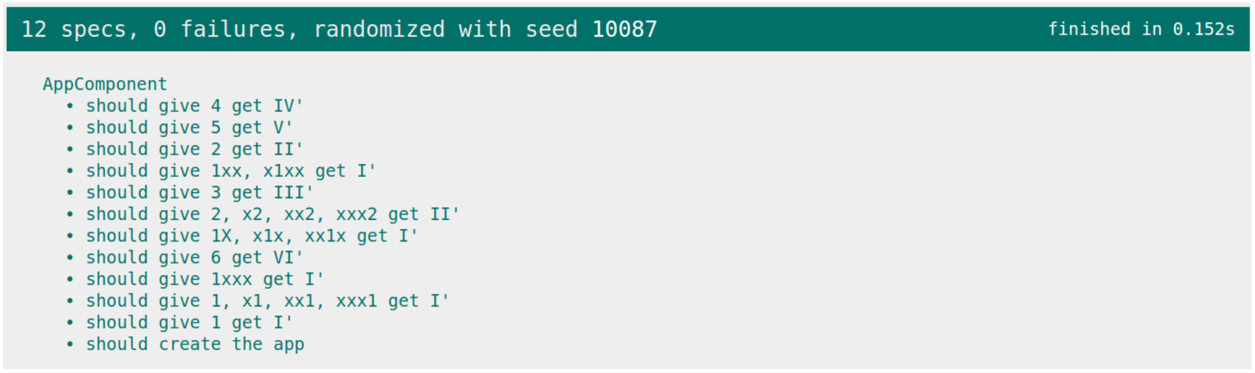


```
12 specs, 1 failure, randomized with seed 80759                finished in 0.346s
Spec List | Failures

AppComponent > should give 1xxx get I'

Expected 'II' to be 'I'.

Error: Expected 'II' to be 'I'.
    at <Jasmine>
    at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/app.component.spec.ts:108:67)
    at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone.js:372:1
    at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2015/zone-testi
```

CÓDIGO PARA QUE PASE:



```
Complexity is 3 Everything is cool!
it(`should give 1xxx get I'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  let unidadD: number[] = [ 1122, 1422, 1132, 1182, 1192, 1318, 1722, 1142, 1423, 1854, 1784, 1874, 1188];
  let unidadR: string[] = [];
  for(var i = 0; i < unidadD.length; i++) { unidadR[i] =app.intRomano(Math.trunc(unidadD[i] / 1000) % 10) }

  for(var i = 0; i < unidadD.length; i++) {  expect(unidadR[i]).toBe("I"); }
});
```



```
12 specs, 0 failures, randomized with seed 10087                    finished in 0.152s

  AppComponent
    • should give 4 get IV'
    • should give 5 get V'
    • should give 2 get II'
    • should give 1xx, x1xx get I'
    • should give 3 get III'
    • should give 2, x2, xx2, xxx2 get II'
    • should give 1X, x1x, xx1x get I'
    • should give 6 get VI'
    • should give 1xxx get I'
    • should give 1, x1, xx1, xxx1 get I'
    • should give 1 get I'
    • should create the app
```

Este proceso se puede realizar de forma iterativa para los 3 posibles valores del millar.