escuela técnica superior
de ingeniería informática

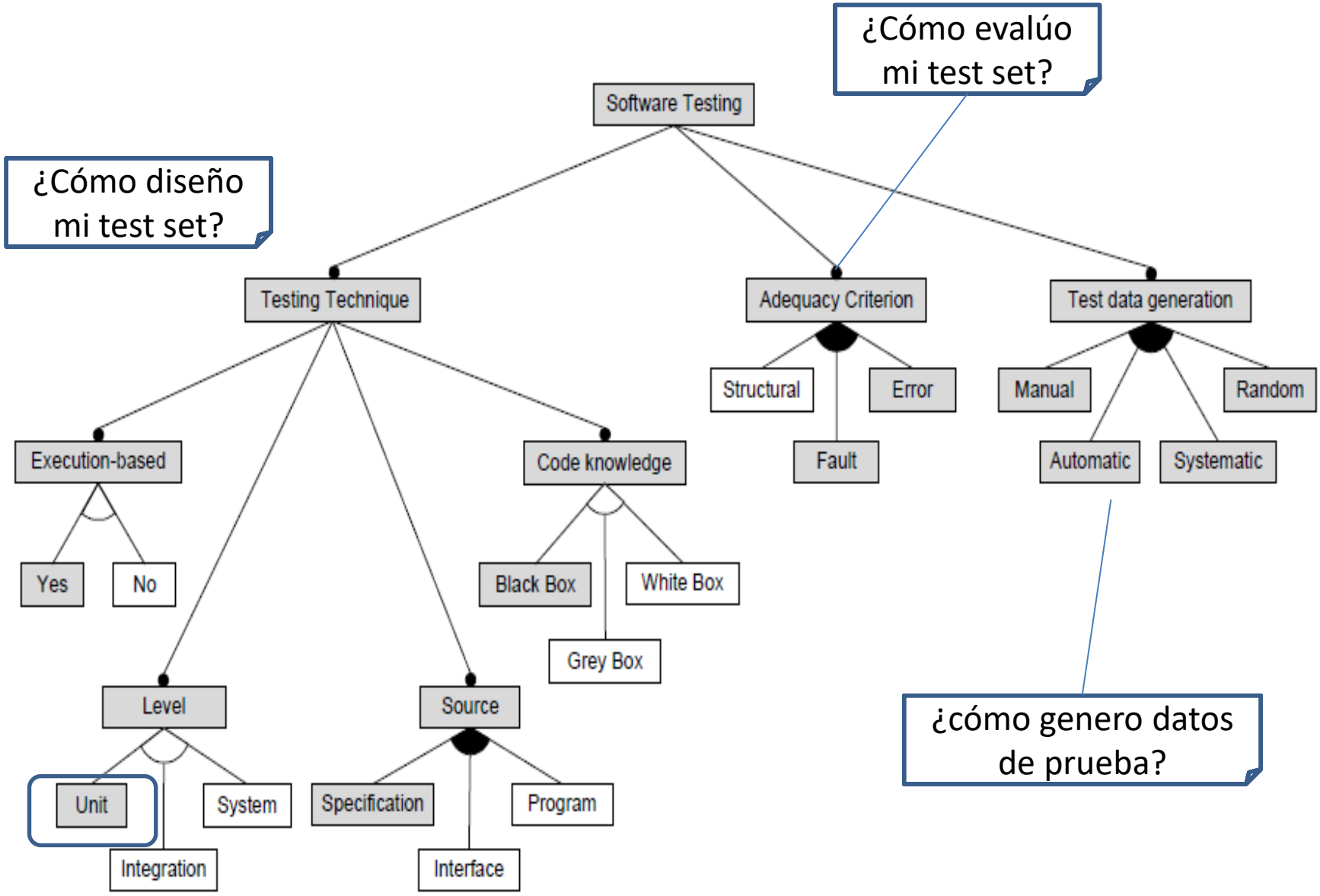
Pruebas de software

Departamento de Lenguajes y Sistemas Informáticos

Evolución y gestión de la configuración

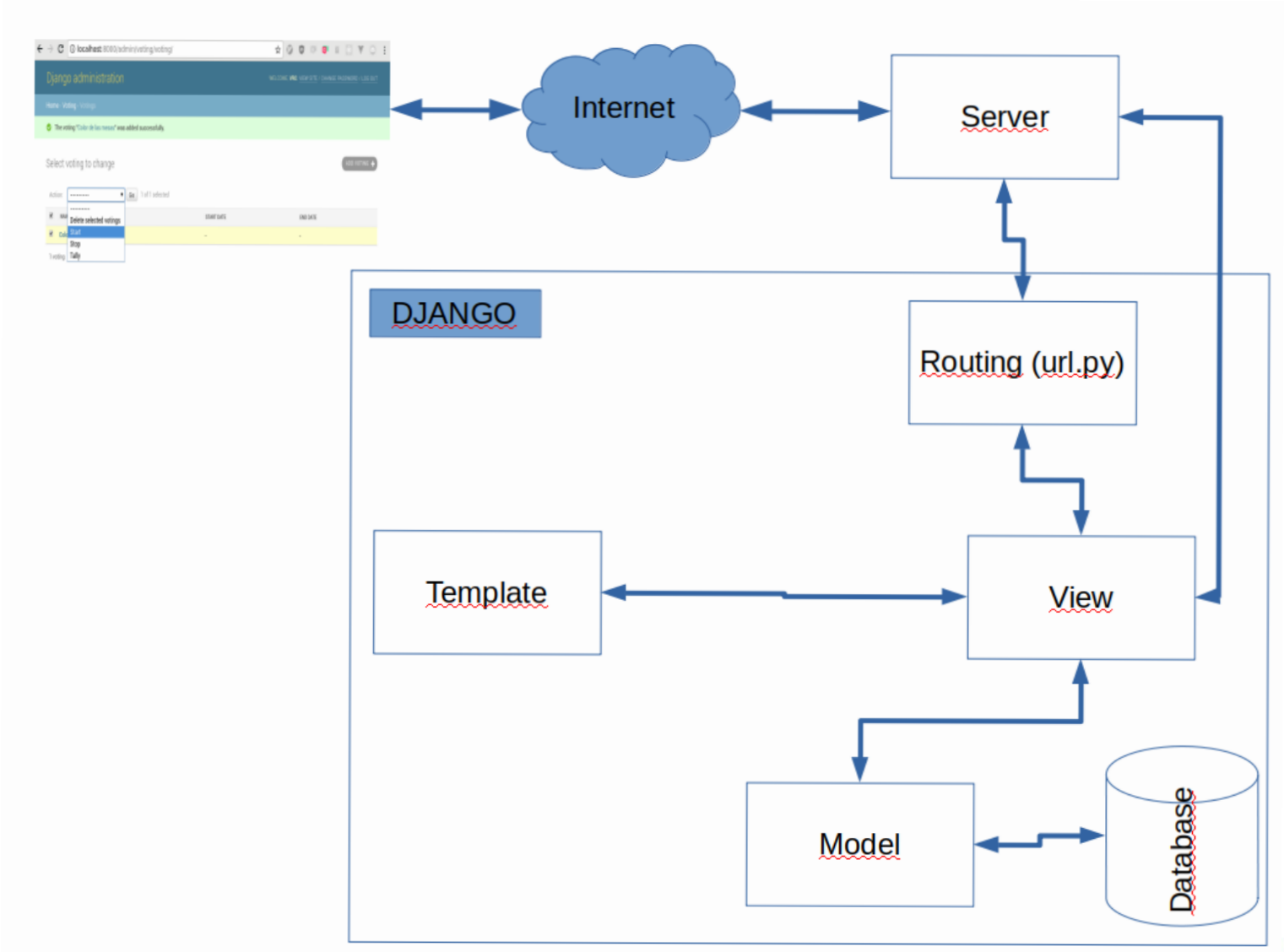
4º Grado en Ingeniería Informática - Ingeniería del Software

Testing:max 2^26 testing posibles!!



Las pruebas unitarias están diseñadas para ejercitar una parte pequeña y específica de funcionalidad

La estructura de Django



Proceso general de prueba

Implementación de pruebas en django

Un Ejemplo:

```
from django.test import TestCase

class SimpleTest(TestCase):
        def test_basic_addition(self):
        """
        Tests that 1 + 1 always equals 2.
        """
                self.assertEqual(1 + 1, 2)
```

Framework de testing unitario

(**unnittest** → Inspirado en *JUnit*)

Conceptos

- **test fixture** - Preparación necesaria para realizar las pruebas
- **test case** - Caso concreto e individual que se quiere probar
- **test suite** - Conjunto de casos de prueba.
- **test runner** - Componente que ejecuta los tests.

Lugar de implementación y ejecución

- La aplicación crea un fichero **tests.py** por defecto.
- Si necesitamos **más complejidad** →
  Crear nuevos scripts de formato **test*.py**

Una vez escritos, se ejecutan desde la terminal:

```
#Corre todos los tests disponibles
$./manage.py test

#Corre los tests dentro de "voting"
$./manage.py test voting
```

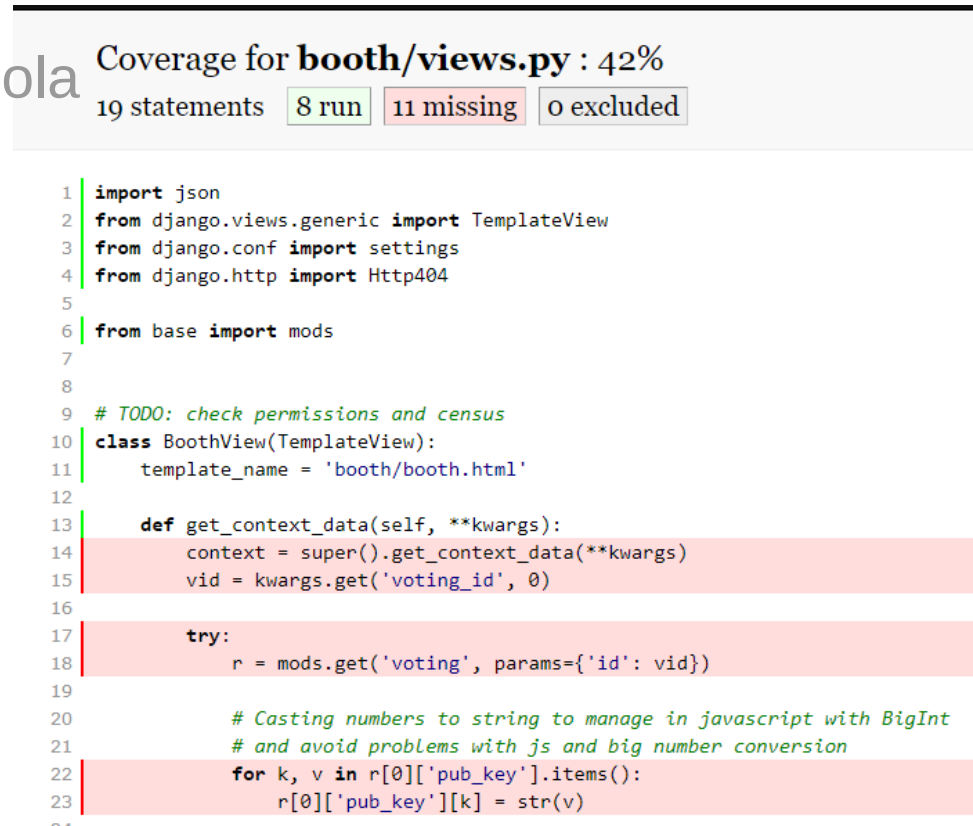
Análisis de Cobertura de Pruebas

- Nos permitirá saber qué partes de la aplicación no están probadas

#Analiza la “./manage.py test”
$coverage run --source . ./manage.py test

#Cuestra el reporte en consola
$coverage report -m

#Crea un reporte en html
$coverage report html



Coverage for **booth/views.py** : 42%

19 statements | 8 run | 11 missing | 0 excluded

```
1  import json
2  from django.views.generic import TemplateView
3  from django.conf import settings
4  from django.http import Http404
5
6  from base import mods
7
8
9  # TODO: check permissions and census
10 class BoothView(TemplateView):
11     template_name = 'booth/booth.html'
12
13     def get_context_data(self, **kwargs):
14         context = super().get_context_data(**kwargs)
15         vid = kwargs.get('voting_id', 0)
16
17         try:
18             r = mods.get('voting', params={'id': vid})
19
20             # Casting numbers to string to manage in javascript with BigInt
21             # and avoid problems with js and big number conversion
22             for k, v in r[0]['pub_key'].items():
23                 r[0]['pub_key'][k] = str(v)
```

Probando modelos

- En Django, los tests referentes a la base de datos no usan la BBDD de **producción**.

- (No es necesario declararla en settings.py)

Probando modelos

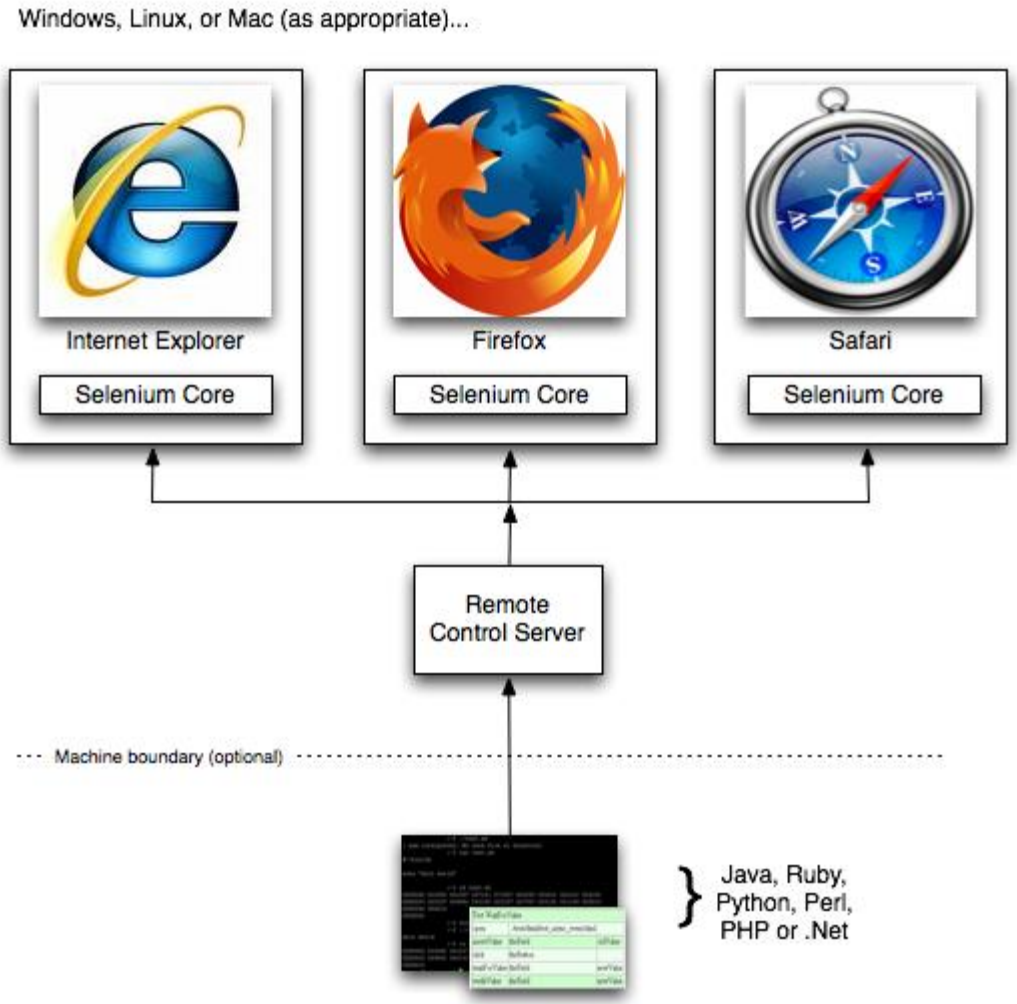
```
def setUp(self):
        super().setUp()
        self.census = Census(voting_id=1, voter_id=1)
        self.census.save()

def tearDown(self):
        super().tearDown()
        self.census = None

def test_store_census(self):
        self.assertEqual(Census.objects.count(), 1)
```

Probando las vistas de API

- Cada módulo ofrece funcionalidad en las vistas (views.py) que habrá que probar

```
def test_update_voting_400(self):
        v = self.create_voting()
        data = {} #El campo action es requerido en la request
        self.login()
        response = self.client.put('/voting/{}'.format(v.pk),
 data, format= 'json')
        self.assertEquals(response.status_code, 400)
```

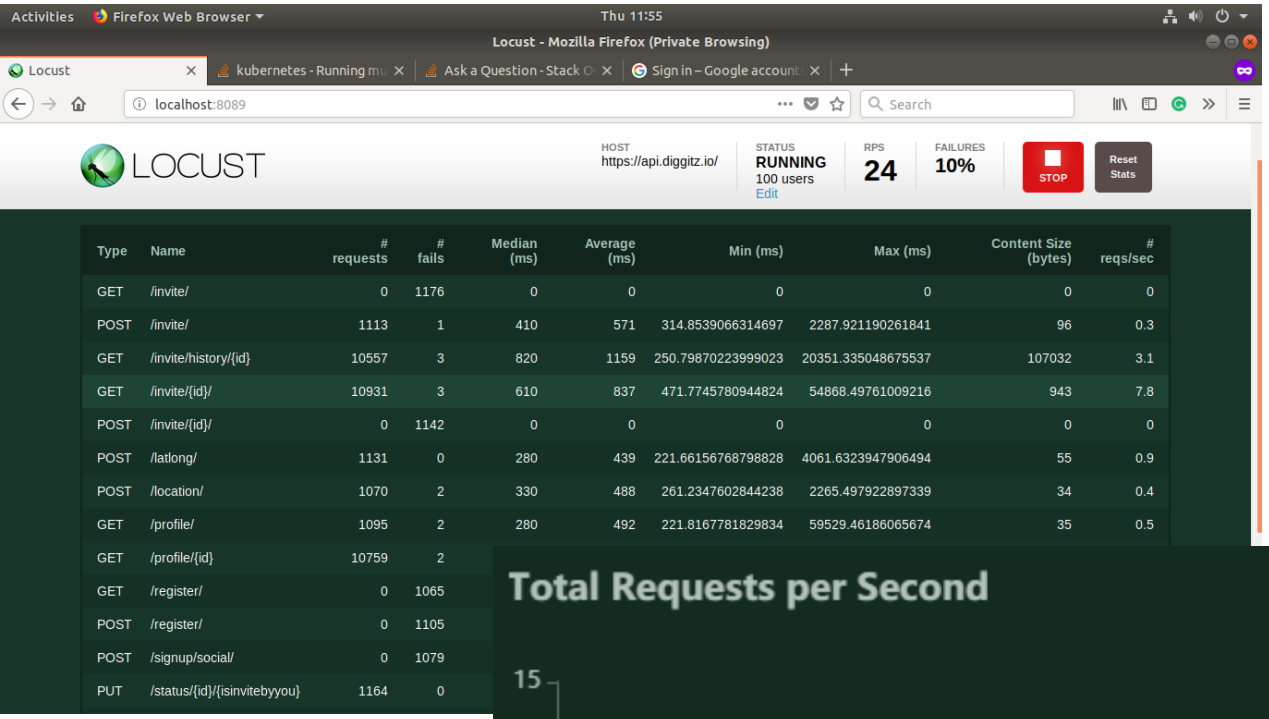
Probando las vistas con navegación



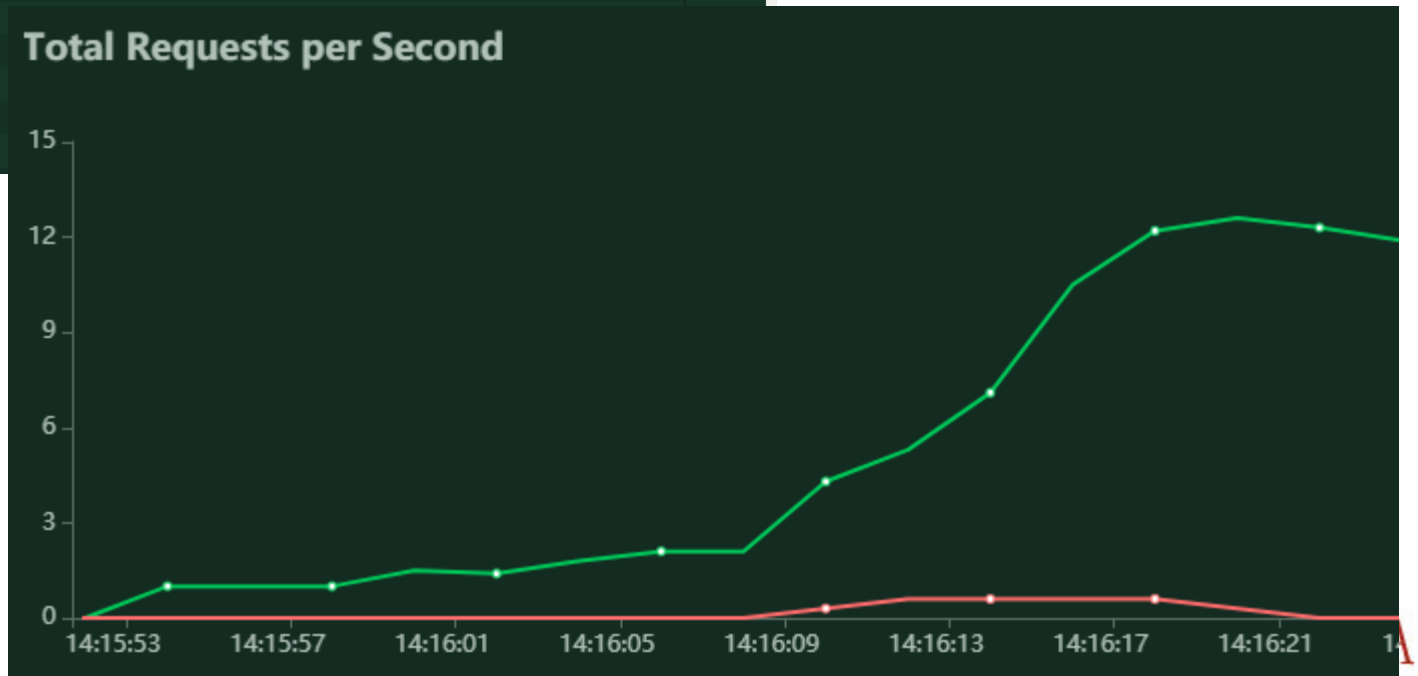
Probando la carga del sistema

Source: Experian Marketing Services' Hitwise

Probando la carga del sistema



Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	/invite/	0	1176	0	0	0	0	0	0
POST	/invite/	1113	1	410	571	314.8539066314697	2287.921190261841	96	0.3
GET	/invite/history/{id}	10557	3	820	1159	250.79870223999023	20351.335048675537	107032	3.1
GET	/invite/{id}/	10931	3	610	837	471.7745780944824	54868.49761009216	943	7.8
POST	/invite/{id}/	0	1142	0	0	0	0	0	0
POST	/latlong/	1131	0	280	439	221.66156768798828	4061.6323947906494	55	0.9
POST	/location/	1070	2	330	488	261.2347602844238	2265.497922897339	34	0.4
GET	/profile/	1095	2	280	492	221.8167781829834	59529.46186065674	35	0.5
GET	/profile/{id}	10759	2						
GET	/register/	0	1065						
POST	/register/	0	1105						
POST	/signup/social/	0	1079						
PUT	/status/{id}/{isinvitebyyou}	1164	0						



escuela técnica superior
de ingeniería informática

Integración continua

Departamento de
Lenguajes y Sistemas Informáticos

Evolución y gestión de la configuración

4º Grado en Ingeniería Informática - Ingeniería del Software