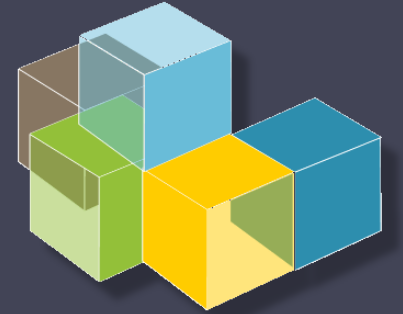


Universidad de Oviedo



Escuela de
Ingeniería
Informática



ARQUITECTURA
DEL SOFTWARE

Arquitectura del Software

Lab. 12

Monitorización y optimización

2022-23

José Emilio Labra Gayo
Pablo González
Irene Cid
Cristian Augusto

Monitorización y Profiling

Monitorizar: Observar comportamiento de un software

- Cuadros de mando

- Habitualmente, después del despliegue

Profiling (caracterizar): Medir rendimiento de un software mientras se ejecuta

- Identificar partes que contribuyen a un problema

- Mostrar dónde centrar los esfuerzos para mejorar rendimiento

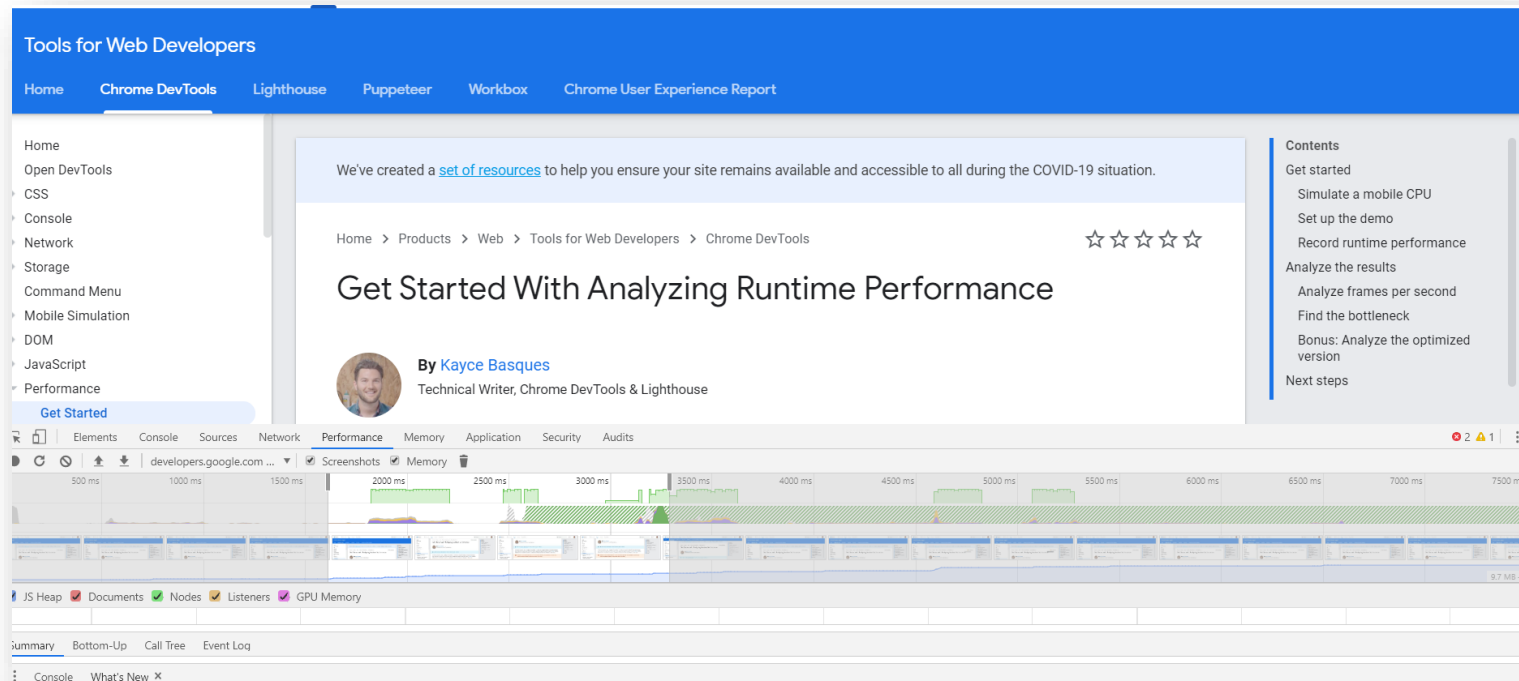
- Suele hacerse antes del despliegue

Profiling

- Herramientas que nos permite optimizar el código que empeora alguna de las métricas que queremos conseguir (tiempo de respuesta, uso de memoria)
 - La optimización es un proceso iterativo :
 - Localizar código que presenta problemas.
 - Modifica el código .
 - Repite hasta que tu código cumple con las métricas
- Específicas por componente software:
 - Jboss profiler – sql server profilers – javascript profiler..

Navegador: developer tools

- Monitorizar/chequear rendimiento



<https://developers.google.com/web/tools/chrome-devtools/evaluate-performance>

Navegador Ejemplo: Google Chrome

Modo incognito

En la esquina superior derecha, click en los tres puntos y nueva ventana incógnito

Windows, Linux, or Chrome OS: Ctrl + Shift + n.

Mac: ⌘ + Shift + n.

Chrome DevTools

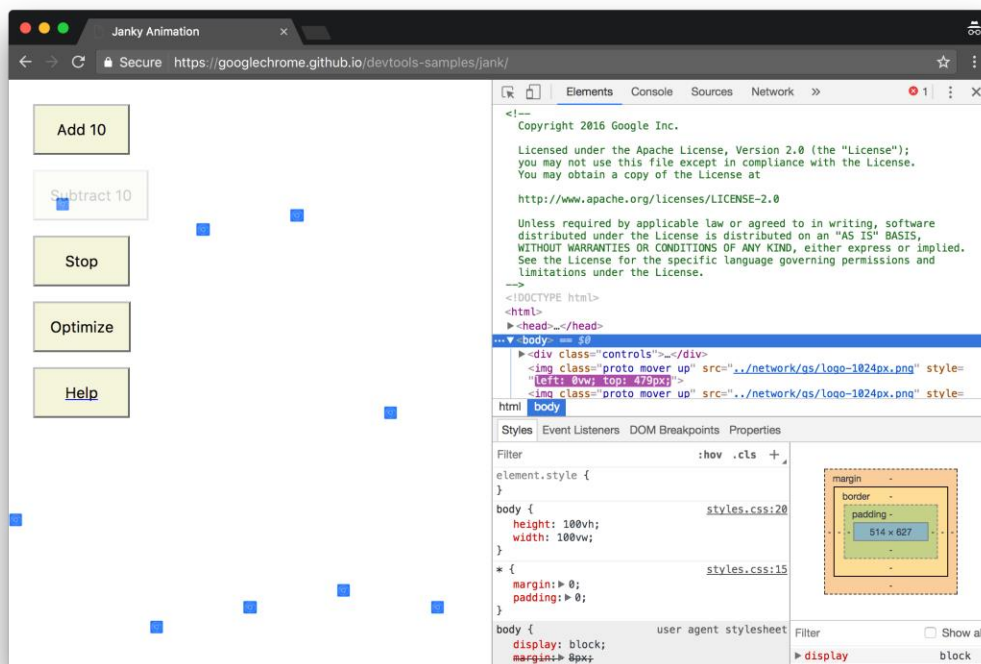
Windows, Linux: Control+Shift+I

Mac: Command+Option+I

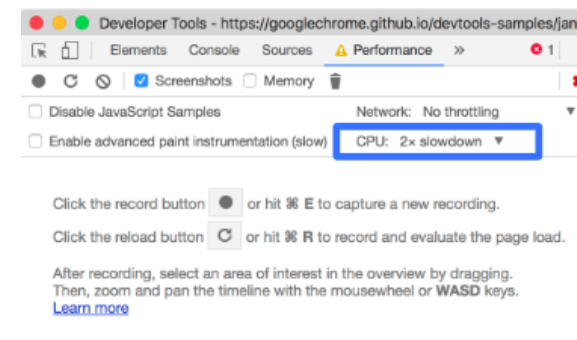


Navegador Ejemplo: Google Chrome

<https://googlechrome.github.io/devtools-samples/jank/>



Performance > CPU > 2 x Slowdown



Performance > Record
click Add 10 (20 veces)
Optimize / Un-optimize
Stop

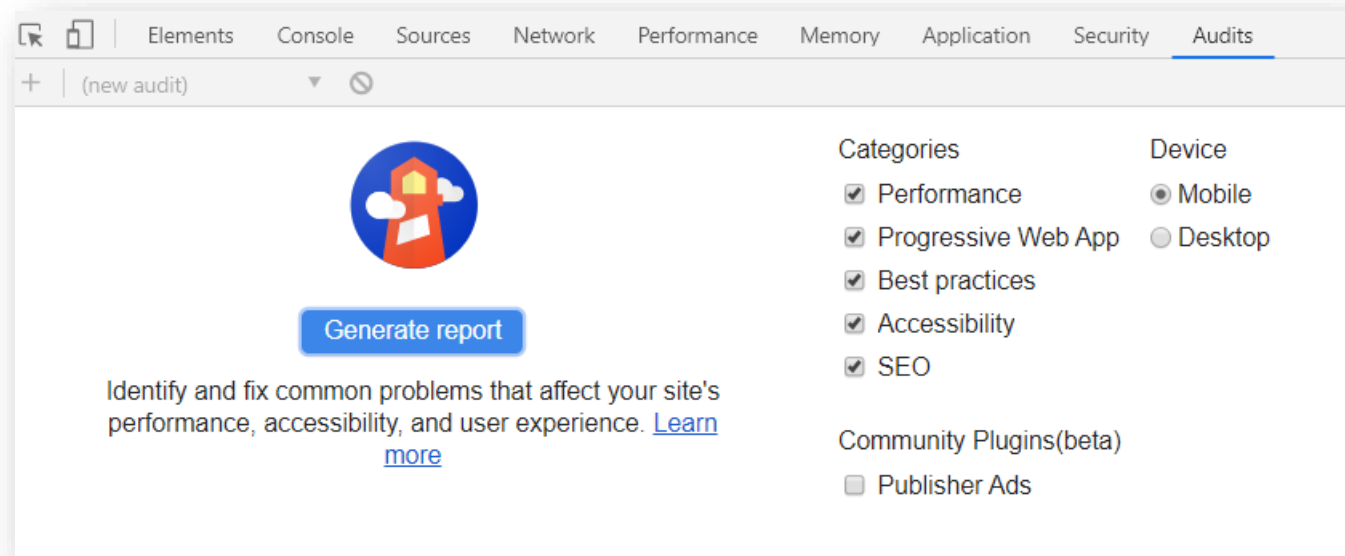
Otras herramientas de navegador

RAIL model (Response, Animation, Idle, Load)

<https://developers.google.com/web/fundamentals/performance/rail>

<https://webpagetest.org/easy>

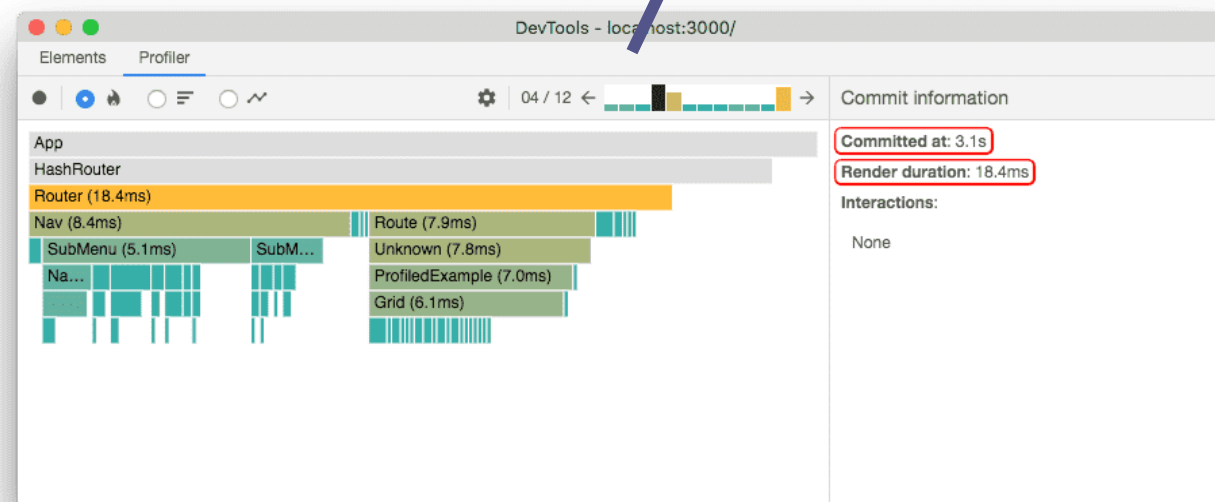
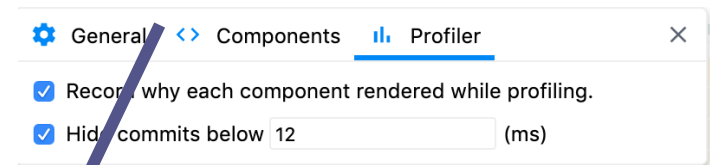
Lighthouse (viene con Chrome)



React Herramientas desarrollo

React trabaja en dos fases:

- Render
- Commit



React Herramientas desarrollo

inrupt

Dirección

avda galicia

Locality

Oviedo

Postal Code

33005

Region

Asturias

General

Components

Profiler

☒ Record why each component rendered while profiling.

☒ Hide commits below 12 (ms)

Elements

Console

Sources

Network

Performance

Memory

Application

Security

Audits

Profiler

Flamegraph

Ranked

Interactions

05 / 18

Bf (0.2ms)

Xd key="subject:._: userprofile_shex_UserProfileAddress__parts_4" (0.2ms)

Anonymous (Memo) (0.2ms)

Context.Consumer (0.2ms)

Xd key="subject:._: userprofile_shex_UserProfileEmail__parts_1" (0.2ms)

Xd key="subject:._: userprofile_shex_UserProfile__parts_0" (0.1ms)

Bf key="928d078d-f6d4-4558-9cad-681ed06be0d1" (0.1m...

Bf key="790e97ce-bf12-4c9b-9ce9-425b28df22fb" (0.1ms)

Bf key="24a59de4-8ab1-43a2-b705-9ce2737ca53e" (0.1ms)

Xd key="subject:formHeading" (0.1ms)

Xd key="subject:._: userprofile_shex_UserProfileAddr...

Why did this render?

• Props changed: (fieldData, modifyFormObject, formObject, onSave)

Rendered at:

4.1s for 17.3ms

4.5s for 83.8ms

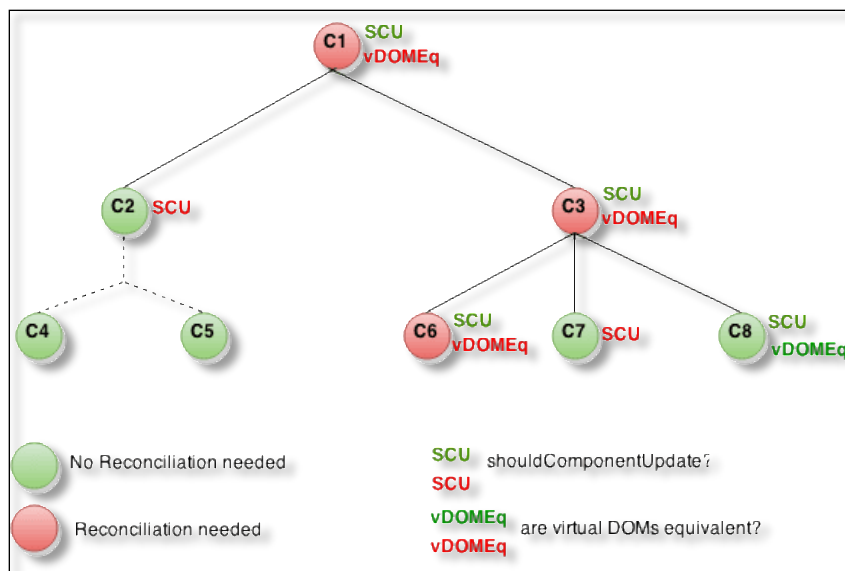
4.6s for 19.9ms

4.7s for 19.9ms

5s for 16.1ms

6.6s for 21.4ms

React: DOM - Virtual DOM



```
class CounterButton extends React.PureComponent {
  constructor(props) {
    super(props);
    this.state = {count: 1};
  }

  render() {
    return (
      <button
        color={this.props.color}
        onClick={() => this.setState(state => ({count: state.count
+ 1}))}>
        Count: {this.state.count}
      </button>
    );
  }
}
```

```
shouldComponentUpdate(nextProps, nextState) {
  if (this.props.color !== nextProps.color) {
    return true;
  }
  if (this.state.count !== nextState.count) {
    return true;
  }
  return false;
}
```

Fuente: <https://es.reactjs.org/docs/optimizing-performance.html>

Monitorización

Monitorizar una aplicación mientras se ejecuta
Registrar uso de CPU, memoria, hilos, etc.

JavaScript:

Chrome (Timeline), Firefox Developer Edition
(Performance tool)

Herramientas de servidor:

JVisualVM, JProfiler, YourKit, Jconsole, etc.

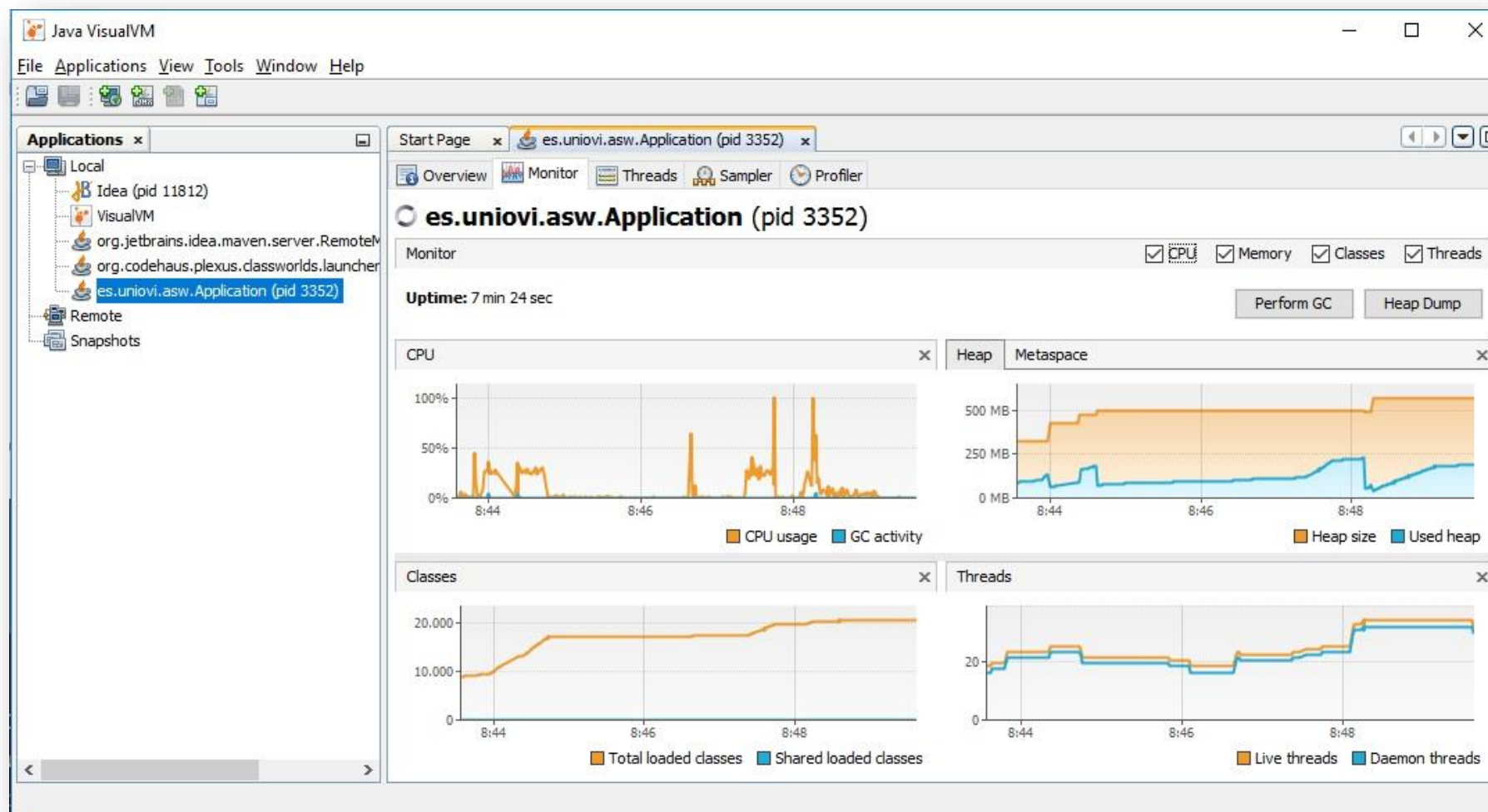
Graphite, Datdog, Prometheus, Graphana

VisualVM

<https://visualvm.github.io/>

Ya está instalada con el JDK: `jvisualvm`

Server/Java: JVisualVM



Monitorización en servidor

- Las plataformas en la nube brindan soluciones de monitoreo
 - También disponible en Google Cloud, Amazon AWS, ..
 - En el caso de Heroku, esta solución no es gratuita
- Aunque también existen soluciones de terceros
 - Prometheus, Graphite, Grafana, Datadog, Nagios, Sensu, ...
- Usaremos: **Prometheus** y **Grafana**
 - lomap_0: https://github.com/arquisoft/lomap_0/tree/master/restapi#monitoring-prometheus-and-grafana



- **Prometheus:** servidor de almacenamiento de datos en series de tiempo
 - Modelo de datos multidimensional
 - Lenguaje flexible de consultas
 - Nodos autónomos de servidor único
 - Configuración estática
- **Grafana:** Visualización de datos. Permite crear, explorar y compartir tableros

Monitorización en servidor

- Necesitamos una biblioteca que pueda extraer algunas métricas de nuestro restapi

1. Instalar el cliente

```
npm install prom-client express-prom-bundle
```

2. Modificamos *restapi/server.js*

```
//Monitoring middleware  
const metricsMiddleware = promBundle({includeMethod: true});  
app.use(metricsMiddleware);
```

3. Si lanzamos el restapi, en */metrics* podremos ver algunos datos de fila que Graphana usaría para trazar buenos gráficos.

Podemos elegir que métrica medir [[doc](#)]



Monitorización en servidor

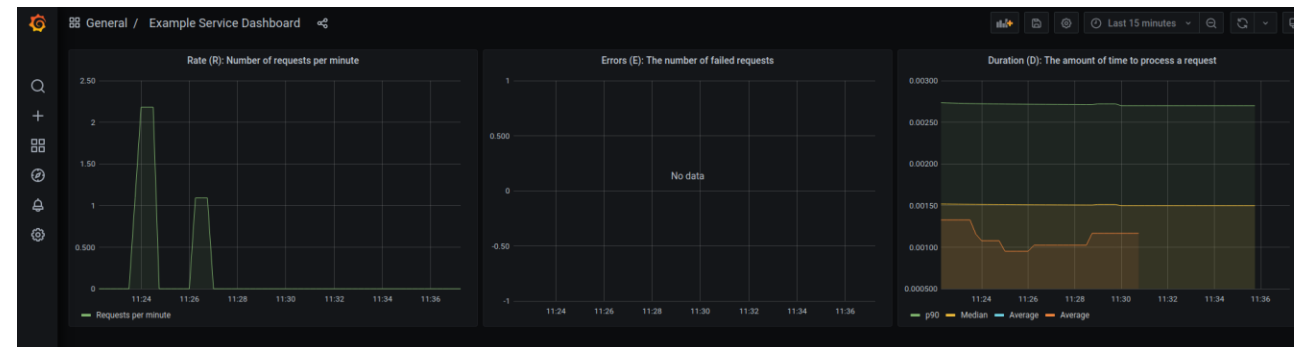
- Graphana no puede usar esta información directamente, necesita [Prometheus](#)
 - Prometheus recuperará los datos expuestos por el restapi y los almacenará para que Grafana pueda consumirlos.
 - Trabajaremos con una docker image [prom/prometheus] que se puede configurar a través de un solo archivo

```
restapi > monitoring > prometheus > ! prometheus.yml
1  global:
2    scrape_interval: 5s
3  scrape_configs:
4    - job_name: "example-nodejs-app"
5      static_configs:
6        - targets: ["restapi:5000"]
```



Monitorización en servidor

- Como configurar Graphana
 - Graphana usará Prometheus como fuente de datos
 - Tenemos una docker image para ejecutarlo [grafana/grafana]
 - Nosotros necesitamos configurar datasource y el dashboard (gráficos a visualizar)



Referencias

- Monitorización y Profiling
 - **Get Started With Analyzing Runtime Performance**
<https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/>
 - **How to Use the Timeline Tool**
<https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool#profile-js>
 - **Otro Ejemplo**
<https://github.com/coder-society/nodejs-application-monitoring-with-prometheus-and-grafana>