Walmart

Mantenibilidad de Software en una Aplicación en Javascript

Gonzalo Fernández - Senior FulStack Developer

vadokdev VadokDev VadokDev

María Paz Morales - FullStack Developer

paz-morales-llopis paz52

Agenda

1. ¿Qué es la Mantenibilidad de Software?
2. Workshop con una aplicación en Javascript
3. Conclusiones generales

Mantenibilidad de Software

"La facilidad con la que un software o componente puede ser modificado para corregir fallas, mejorar su desempeño u otros atributos, o adaptarse a un entorno cambiante"

IEEE Standard Glossary of Software Engineering Terminology

Fases del Software

Desarrollo

Mantención

"In software, adding a six-lane automobile expressway to a railroad bridge is considered maintenance—and it would be particularly good if you could do it without stopping the train traffic."

Paul Stachour & David Collier-Brown
"You Don't Know Jack About Software Maintenance"

Aspectos de la Mantenibilidad

Analizabilidad

Modificabilidad

Modularidad

Reusabilidad

Testeabilidad

Analizabilidad

Grado de efectividad y eficiencia con el cual es posible determinar el impacto que un cambio tiene en una o más partes de un sistema.

Permite diagnosticar fallos en el sistema, e identificar las partes que deben ser modificadas.

¿Qué herramientas tenemos para mejorarlo?

- Typescript (tipado),
- Prettier y Linters.
- Comentarios.
- Documentación.
- Complejidad ciclomática (métrica).

Modificabilidad

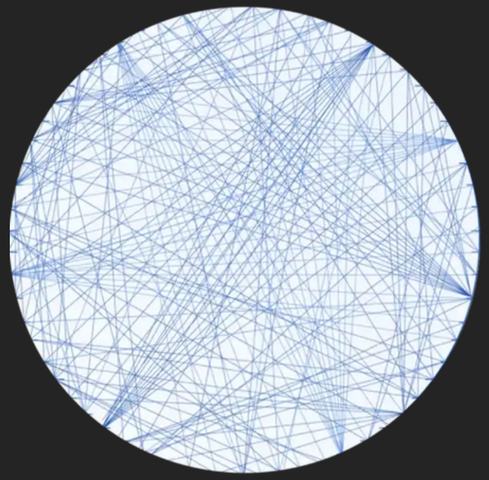
Grado con el cual se puede modificar un sistema sin introducir defectos o degradar su calidad.

¿Qué herramientas tenemos para mejorarlo?

- Linters.
- Tests.
- Typescript.
- Documentación.
- Comentarios.
- Reducir la deuda técnica.

Modularidad

Grado con que un sistema está compuesto de componentes independientes, tal que un cambio en uno de ellos tiene un impacto mínimo en los demás.



No modular

Modular

Modularidad

Grado con que un sistema está compuesto de componentes independientes, tal que un cambio en uno de ellos tiene un impacto mínimo en los demás.

¿Qué herramientas tenemos para mejorarlo?

- Typescript.
- Forzar inmutabilidad con linters (JavaScript tiene soporte nativo parcial para la inmutabilidad de sus datos).

Reusabilidad

Grado con que una parte del sistema puede ser utilizado en otros sistemas, o bien, o en construir otras partes del mismo.

Una parte del sistema puede ser reutilizada si su interfaz es definida correctamente.

¿Qué herramientas tenemos para mejorarlo?

- Tratar de minimizar lo más posible la cantidad de dependencias utilizadas en el proyecto.

Testeabilidad

Grado de eficiencia con el que se pueden establecer criterios de prueba para un sistema, y se pueden realizar pruebas para determinar si se han cumplido esos criterios.

Capacidad de que tan fácil que es definir qué testear.

Capacidad de crear y ejecutar esas pruebas.

Resumiendo

Método	Analizabilidad	Modificabilidad	Modularidad	Reusabilidad	Testeabilidad
Calidad de Programación	+				
Documentación	+	+			
Comentarios en el Código	+	+			
Lenguaje de tipado estático	+	+		+	
Reducir la deuda técnica		+			
Namespaces			+		
Inmutabilidad			+		
Minimizar las dependencias				+	
Arquitectura					+
Tests automáticos					+
Testeado manual de procesos					+

Complejidad Ciclomática (CC)

Métrica que define el número de posibles caminos de ejecución dentro de un bloque de código,

Mientras más alta es la CC, mayor cantidad de ramificaciones tendrá el código y, por lo tanto, será mucho más complejo de analizar.

Complejidad Ciclomática (CC)

Se calcula mediante el Grafo de Ejecución:

- Cada línea de código es un nodo del grafo.
- De cada nodo saldrá una arista al nodo inmediatamente siguiente según la línea de ejecución.

Complejidad Ciclomática (CC)

Se calcula mediante el Grafo de Ejecución:

- Cada línea de código es un nodo del grafo.
- De cada nodo saldrá una arista al nodo inmediatamente siguiente según la línea de ejecución.

$$CC = E - N + 2P$$

- E: Número total de aristas.
- N: Número total de nodos.
- P: Número de componentes conexas.

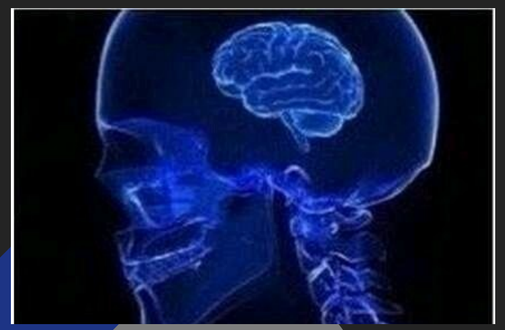
Complejidad Ciclomática (CC)

Se calcula mediante el Grafo de Ejecución:

- Cada línea de código es un nodo del grafo.
- De cada nodo saldrá una arista al nodo inmediatamente siguiente según la línea de ejecución.

$$CC = E - N + 2P$$

- E: Número total de aristas.
- N: Número total de nodos.
- P: Número de componentes conexas.



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```

1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```

1

2

Complejidad Ciclomática (CC)

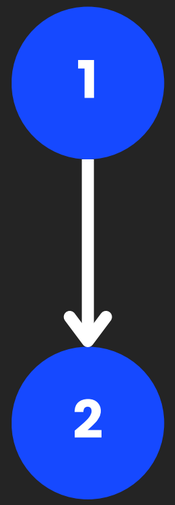
Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```



$$CC = E - N + 2P$$

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```

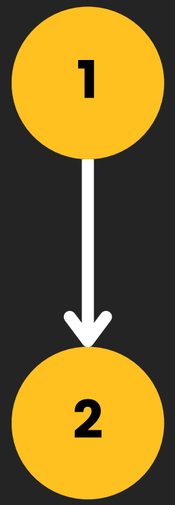
$$CC = E - N + 2P$$

E (aristas) = 1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```



CC = E - N + 2P

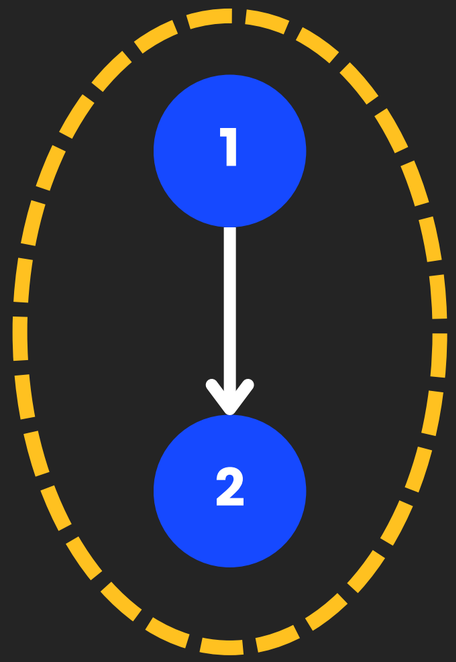
E (aristas) = 1

N (nodos) = 2

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```



CC = E - N + 2P

E (aristas) = 1

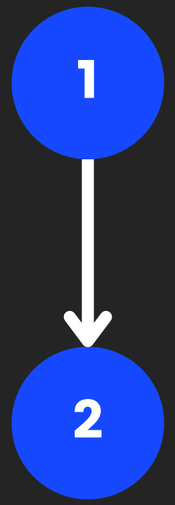
N (nodos) = 2

P (componentes conexas) = 1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```



CC = E - N + 2P

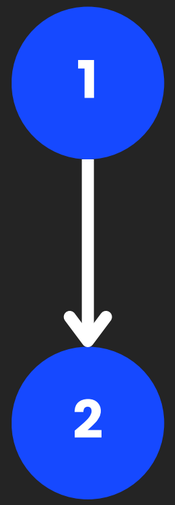
E (aristas) = 1
N (nodos) = 2
P (componentes conexas) = 1

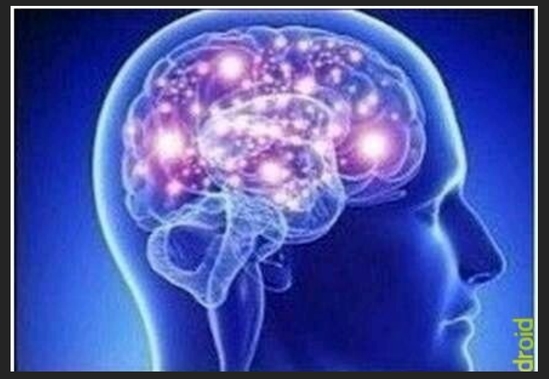
CC = 1 - 2 + 2·1 = 1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
return name;
```





$CC = E - N + 2P$

E (aristas) = 1
N (nodos) = 2
P (componentes conexas) = 1

$CC = 1 - 2 + 2 \cdot 1 = 1$

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

1

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

1

2

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

1

2

3

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

1

2

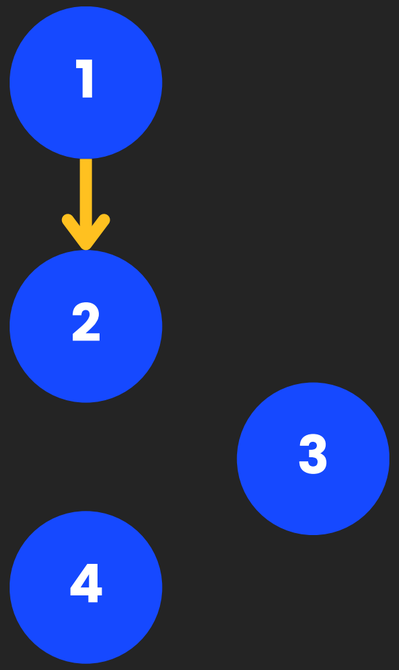
3

4

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

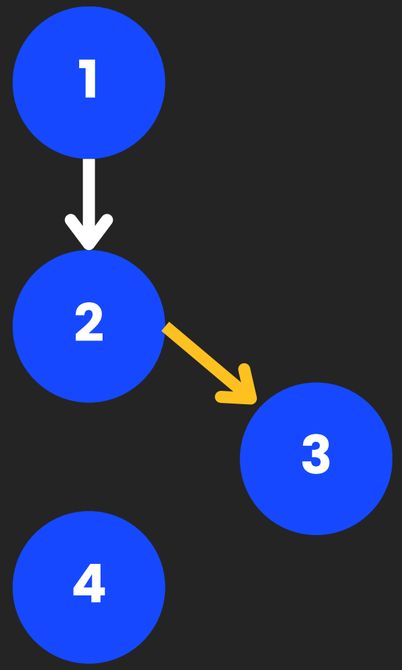
```
const name = "Jack"
if(name == "Bob")
  name += " Uncle"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

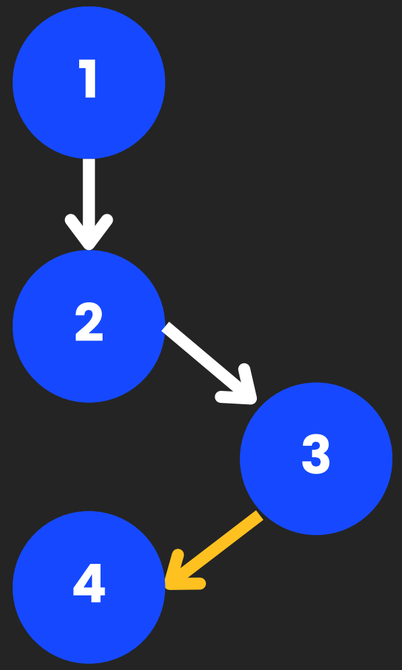
```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

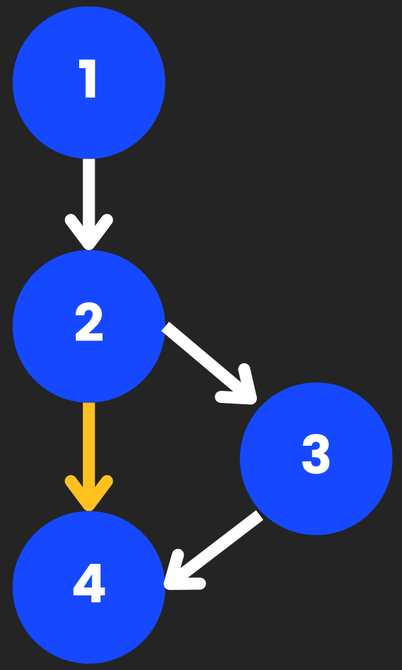
```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

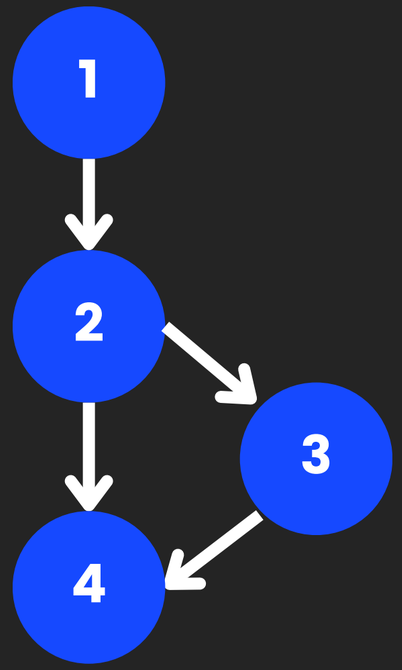


Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```

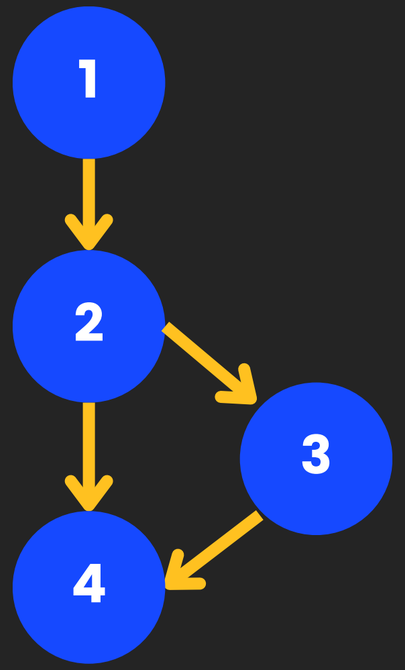
CC = E - N + 2P



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



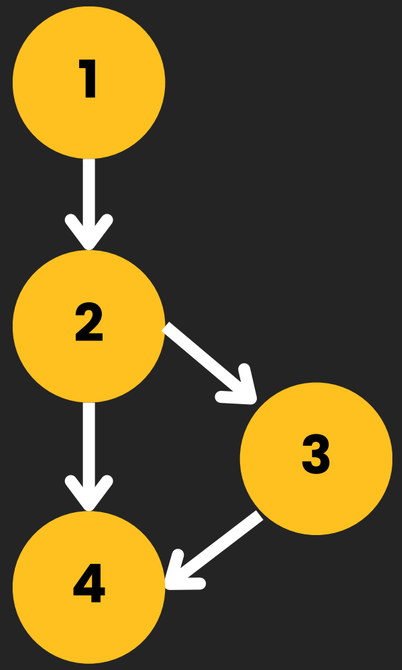
CC = E - N + 2P

E (aristas) = 4

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



CC = E - N + 2P

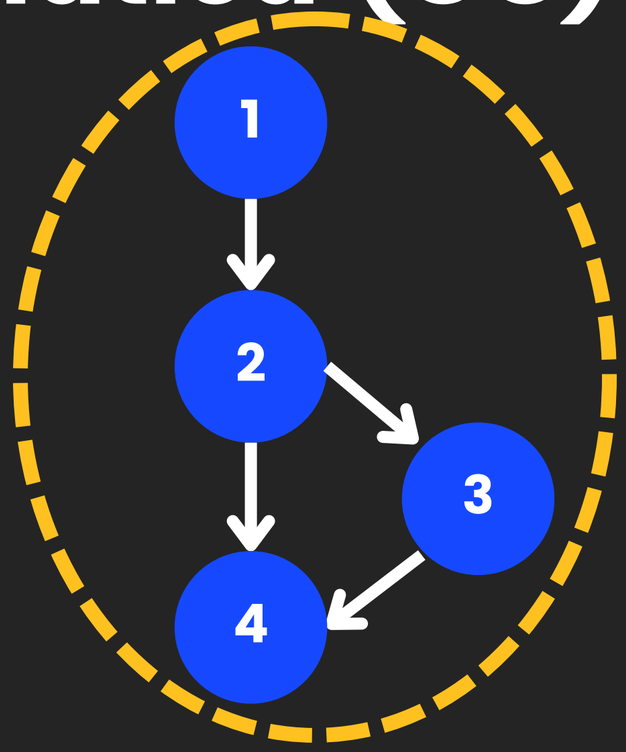
E (aristas) = 4

N (nodos) = 4

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



CC = E - N + 2P

E (aristas) = 4

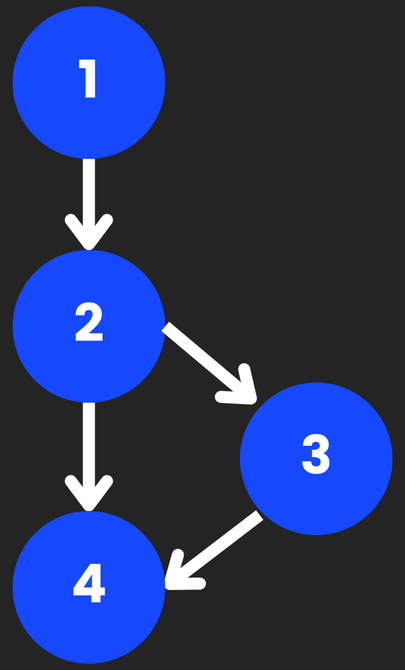
N (nodos) = 4

P (componentes conexas) = 1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



CC = E - N + 2P

E (aristas) = 4

N (nodos) = 4

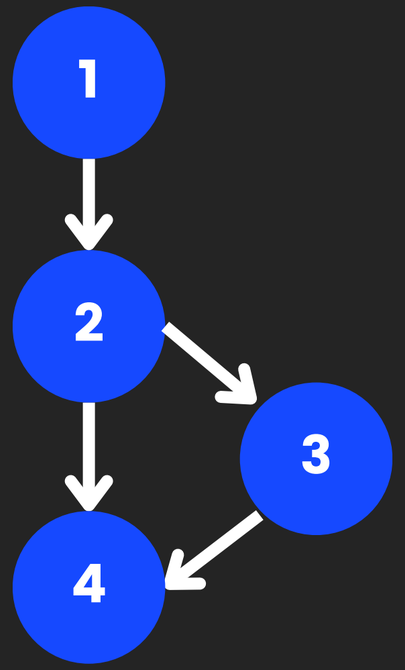
P (componentes conexas) = 1

CC = 4 - 4 + 2·1 = 2

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



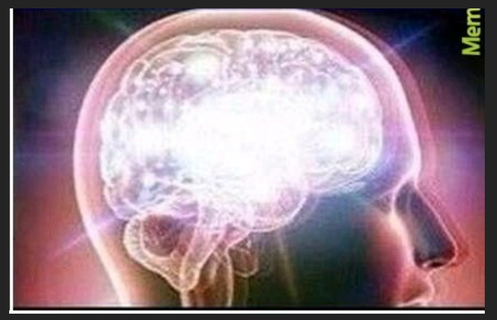
CC = E - N + 2P

E (aristas) = 4

N (nodos) = 4

P (componentes conexas) = 1

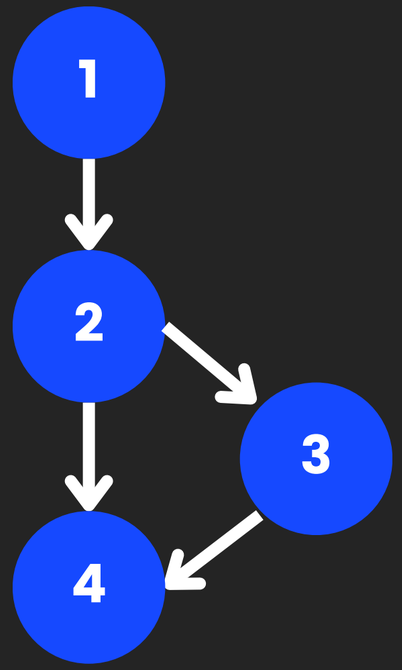
CC = 4 - 4 + 2·1 = 2



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

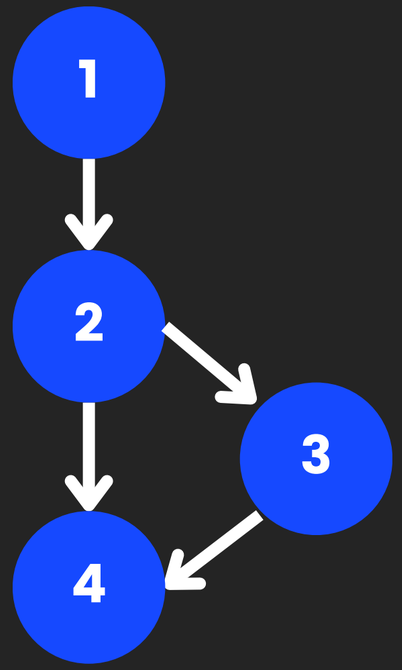
```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

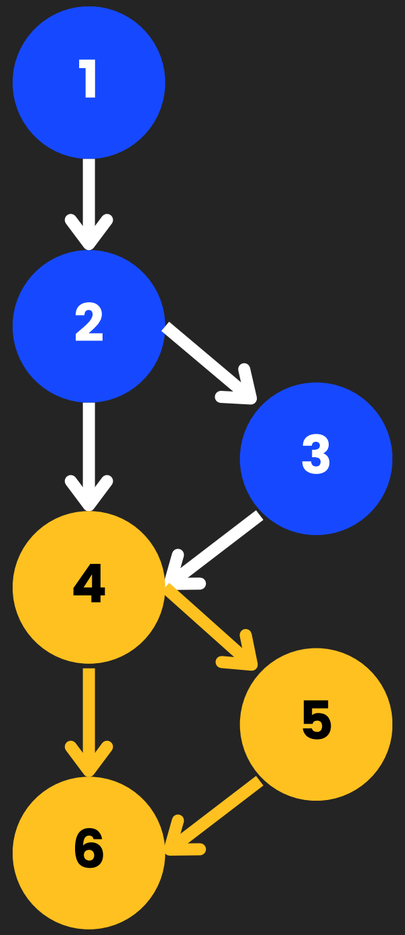
```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

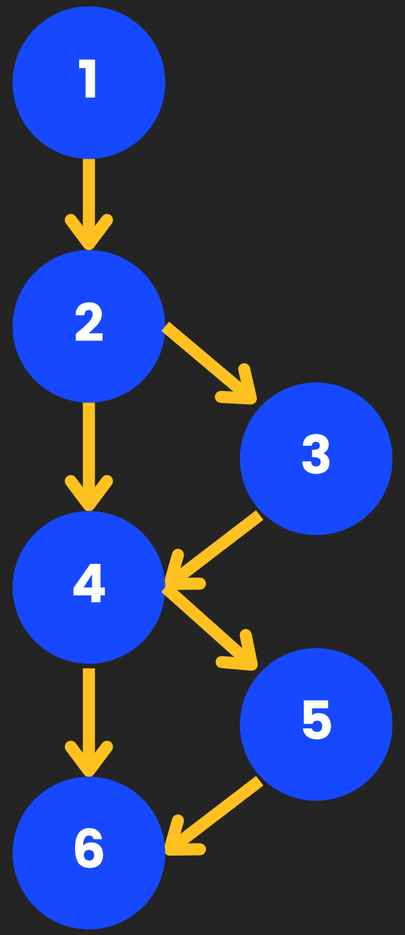
```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```

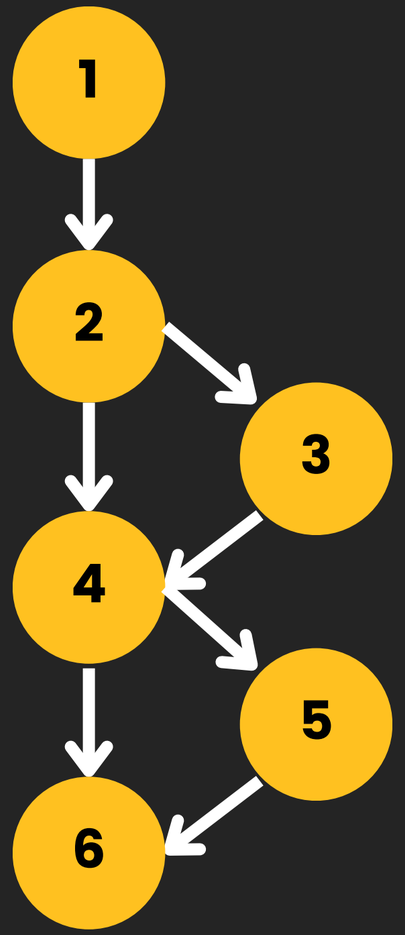


E (aristas) = 7

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



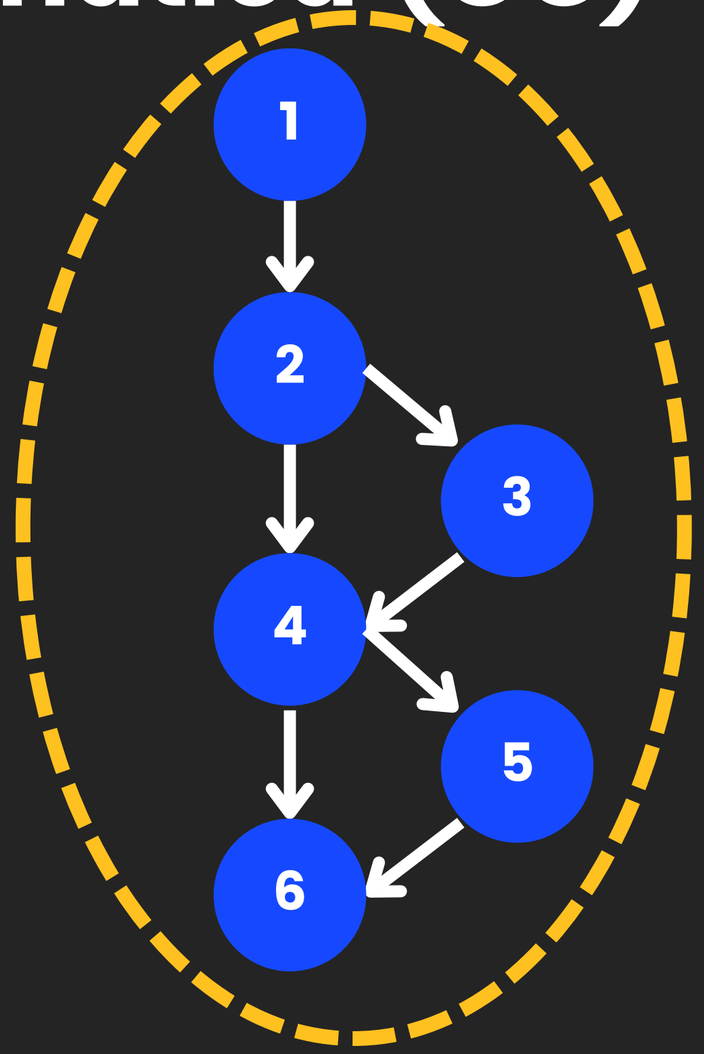
E (aristas) = 7

N (nodos) = 6

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



E (aristas) = 7

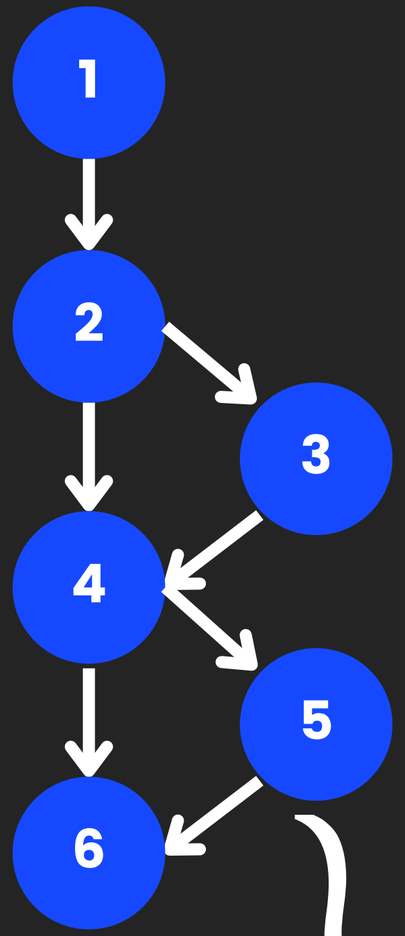
N (nodos) = 6

P (componentes conexas) = 1

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



E (aristas) = 7

N (nodos) = 6

P (componentes conexas) = 1

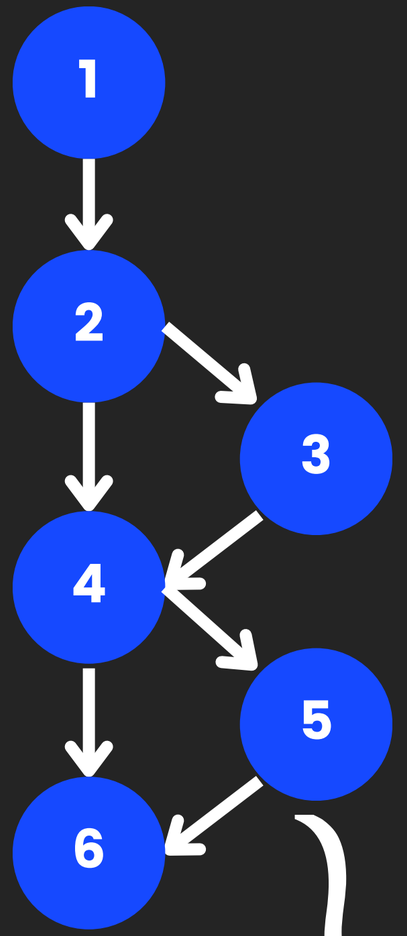
CC = E - N + 2P

CC = 7 - 6 + 2·1 = 3

Complejidad Ciclomática (CC)

Veamos algunos ejemplos:

```
const name = "Jack"
if(name == "Bob")
 name += " Uncle"
if(name == "Bob Uncle")
 name += " Tm"
return name;
```



E (aristas) = 7

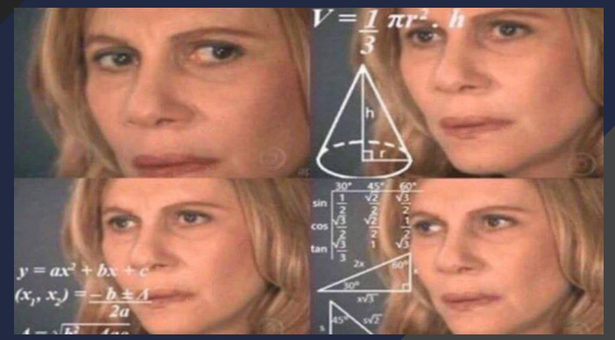
N (nodos) = 6

P (componentes conexas) = 1

CC = E - N + 2P

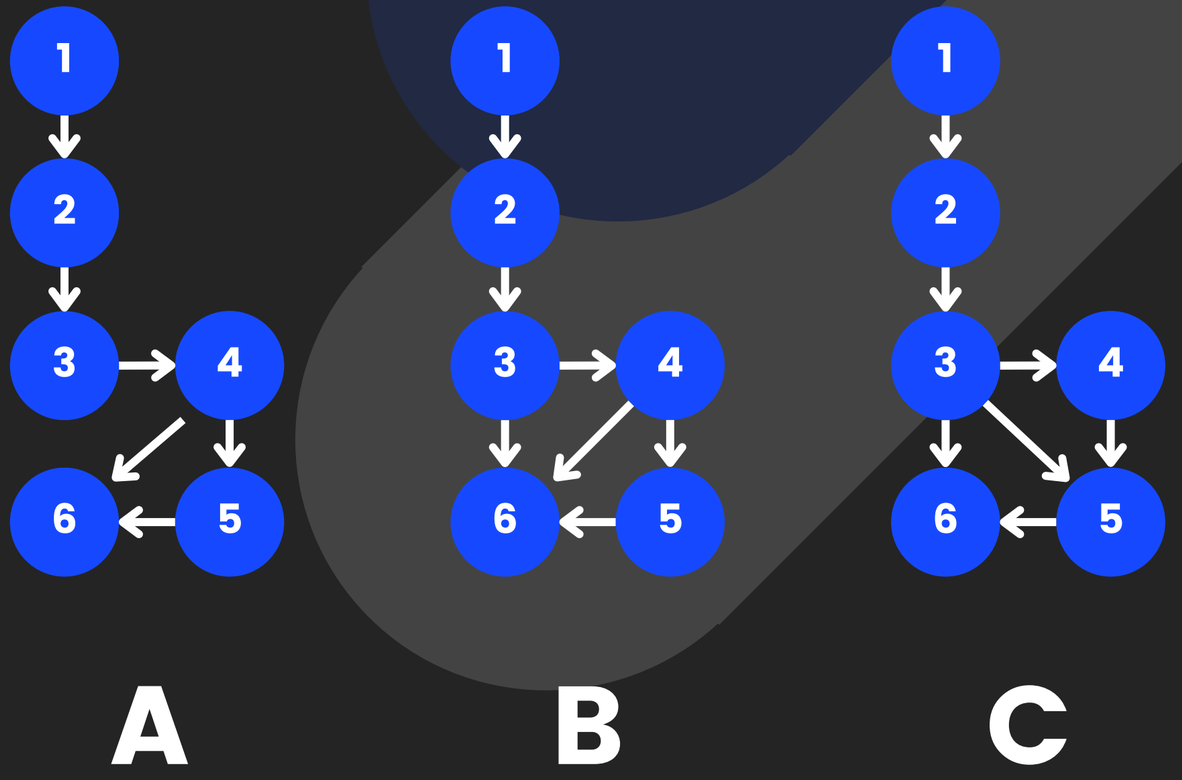
CC = 7 - 6 + 2·1 = 3

Veamos si lo entendieron:

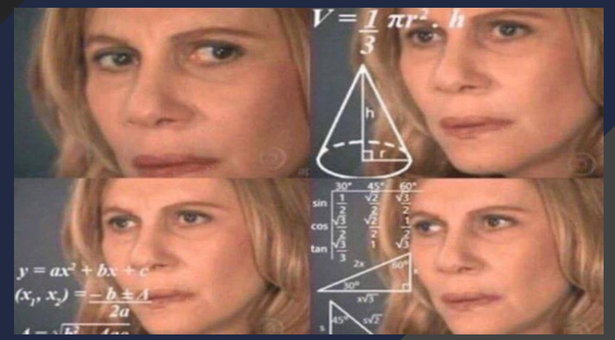


¿Cuál de los siguientes grafos corresponde al grafo de ejecución del código?

```
1.let foo = 0
2.let bar = 10;
3.if( foo == 0 ){
4.   if( bar == 10){
5.      console.log( 'foo = 0, bar = 10');
      }
   }
6.return;
```

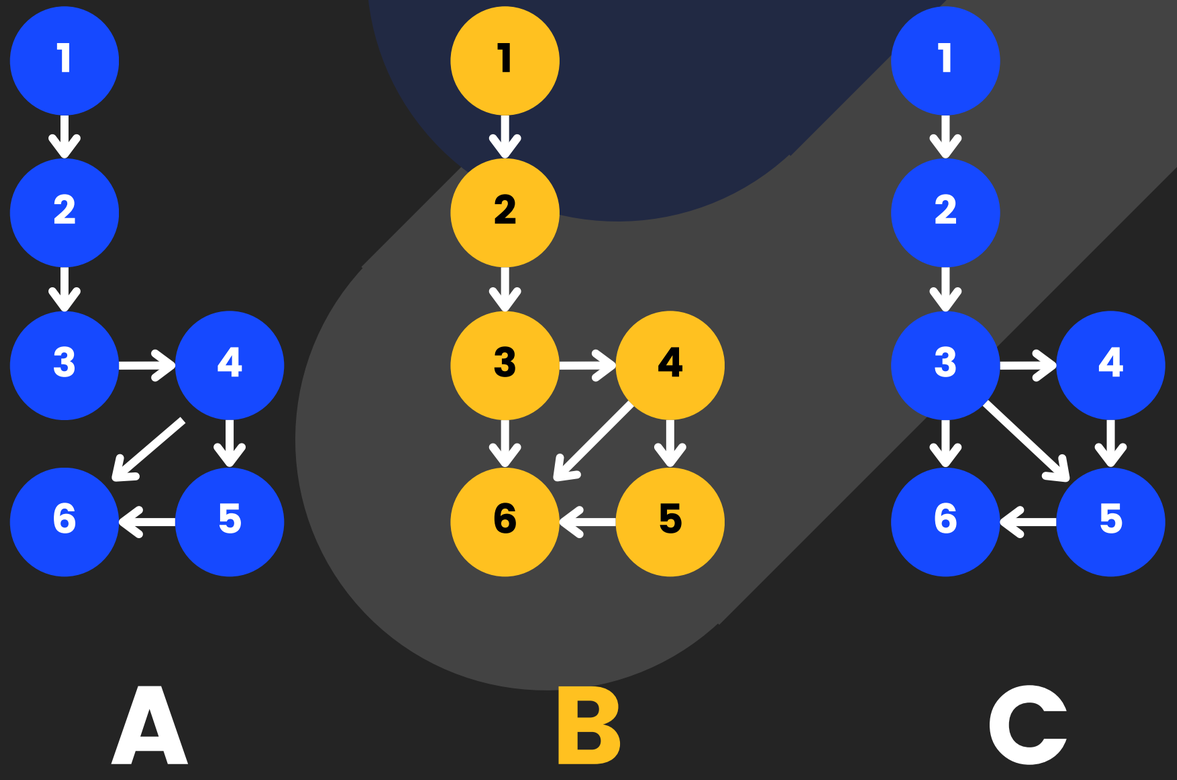


Veamos si lo entendieron:

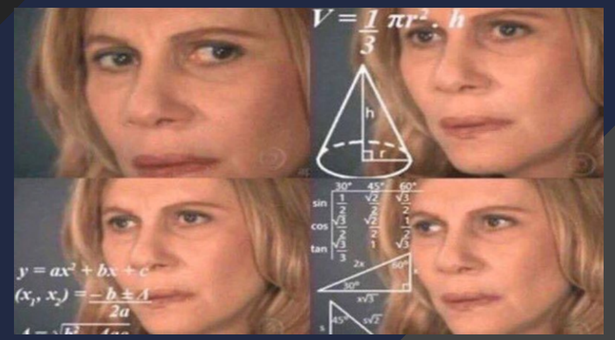


¿Cuál de los siguientes grafos corresponde al grafo de ejecución del código?

```
1.let foo = 0
2.let bar = 10;
3.if( foo == 0 ){
4.   if( bar == 10){
5.      console.log( 'foo = 0, bar = 10');
      }
   }
6.return;
```



Veamos si lo entendieron:

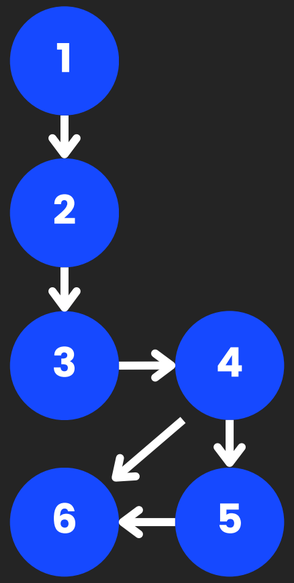


¿Cuál de los siguientes grafos corresponde al grafo de ejecución del código?

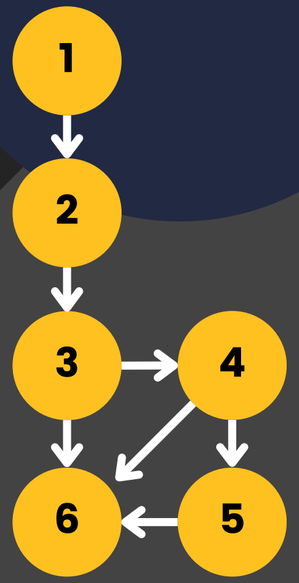
```
1.let foo = 0
2.let bar = 10;
3.if( foo == 0 ){
4.   if( bar == 10){
5.      console.log( 'foo = 0, bar = 10');
     }
   }
6.return;
```

E = 7
N = 6
P = 1

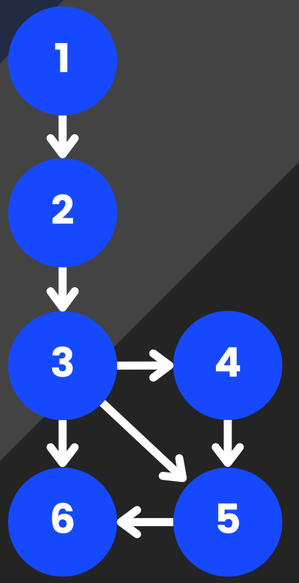
CC = 7 - 6 + 2·1 = 3



A



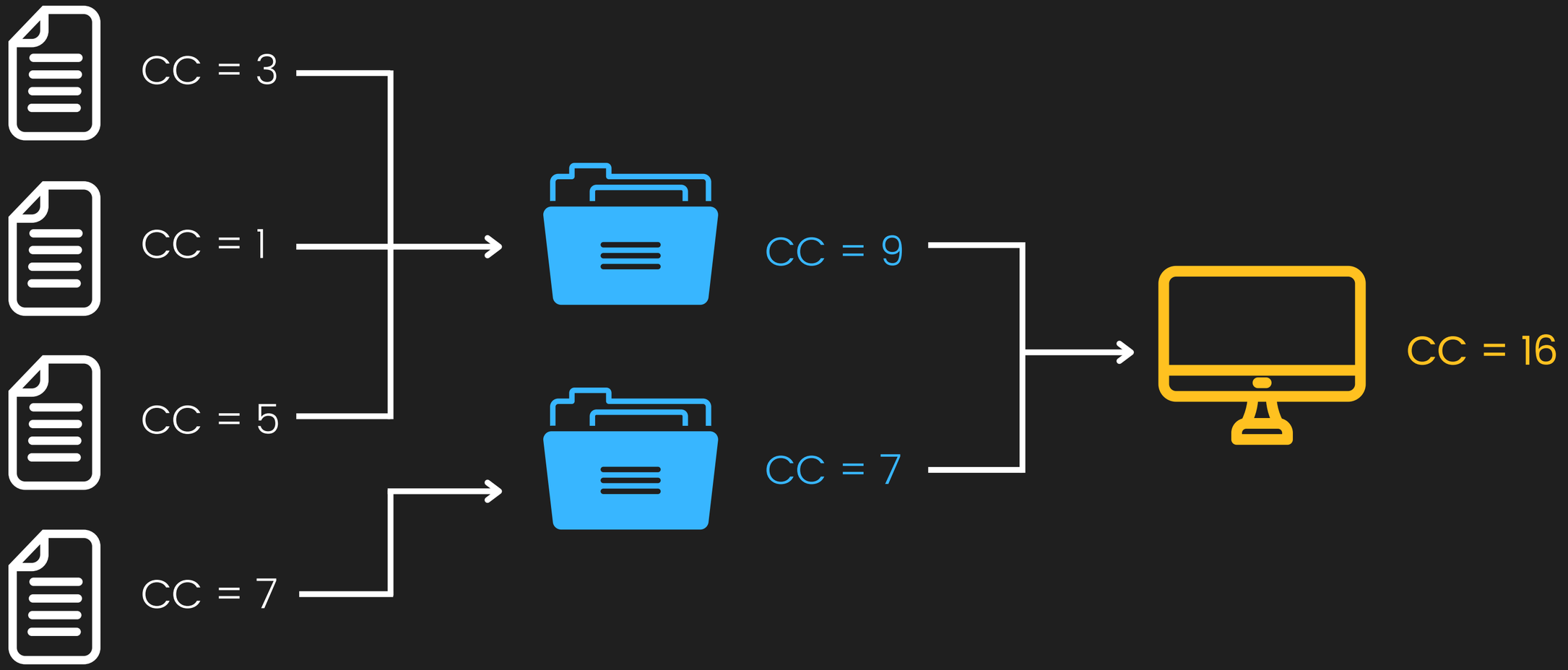
B



C

Complejidad Ciclomática (CC)

¿Y qué pasa con la Complejidad Ciclomática de un sistema completo?

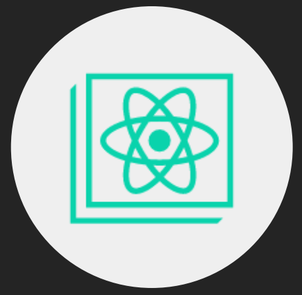


Parte Práctica

Apliquemos estos conceptos en un sistema simple:

Susblisher

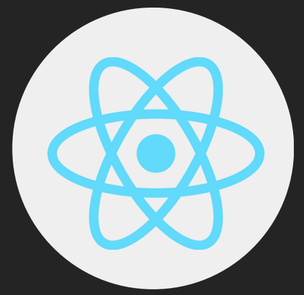
Stack de Trabajo



CRA

SonarQube

Axios



React

Material UI

ESLint

Jest

Testing Library

ⓘ Las herramientas están previamente configuradas y listas para usar.

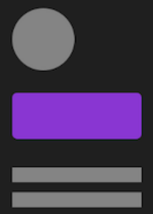
¡Manos a la obra!

Sistema de Diseño

Atomic Design

Sistema de Diseño

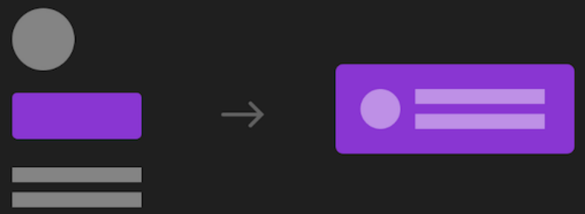
Atomic Design



Átomos

Sistema de Diseño

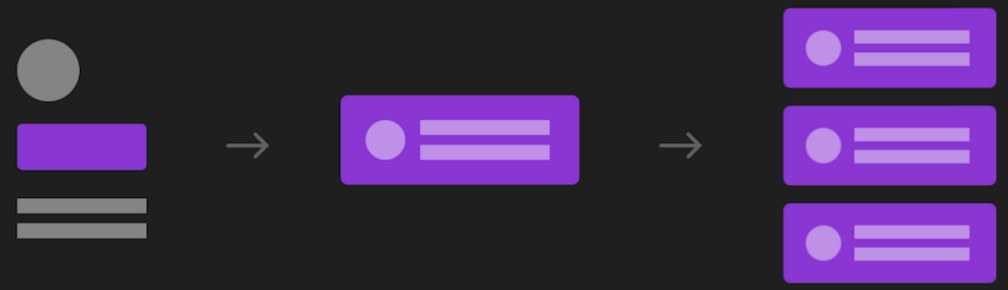
Atomic Design



Átomos Moléculas

Sistema de Diseño

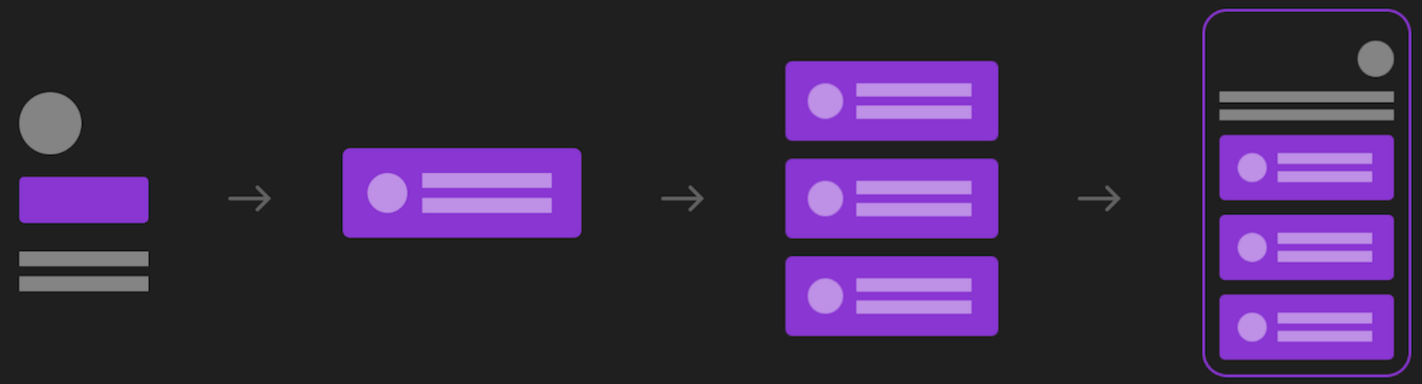
Atomic Design



Átomos Moléculas Organismos

Sistema de Diseño

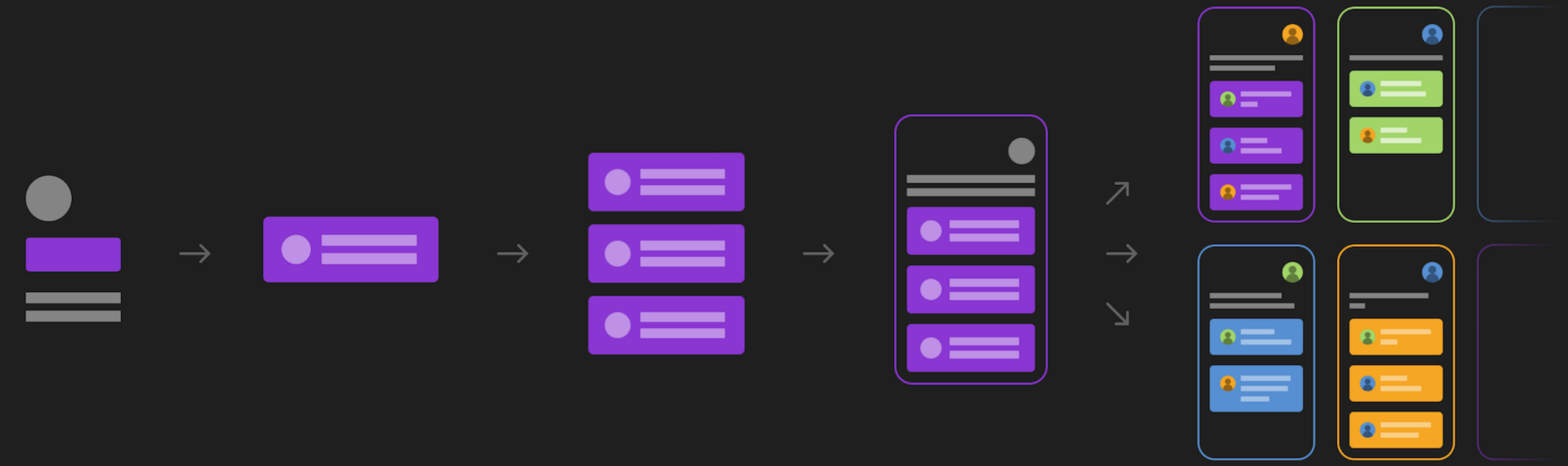
Atomic Design



Átomos Moléculas Organismos Templates

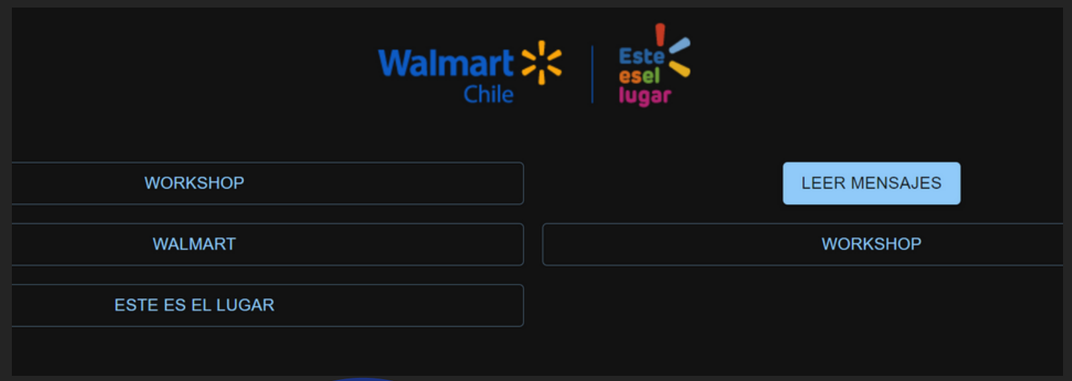
Sistema de Diseño

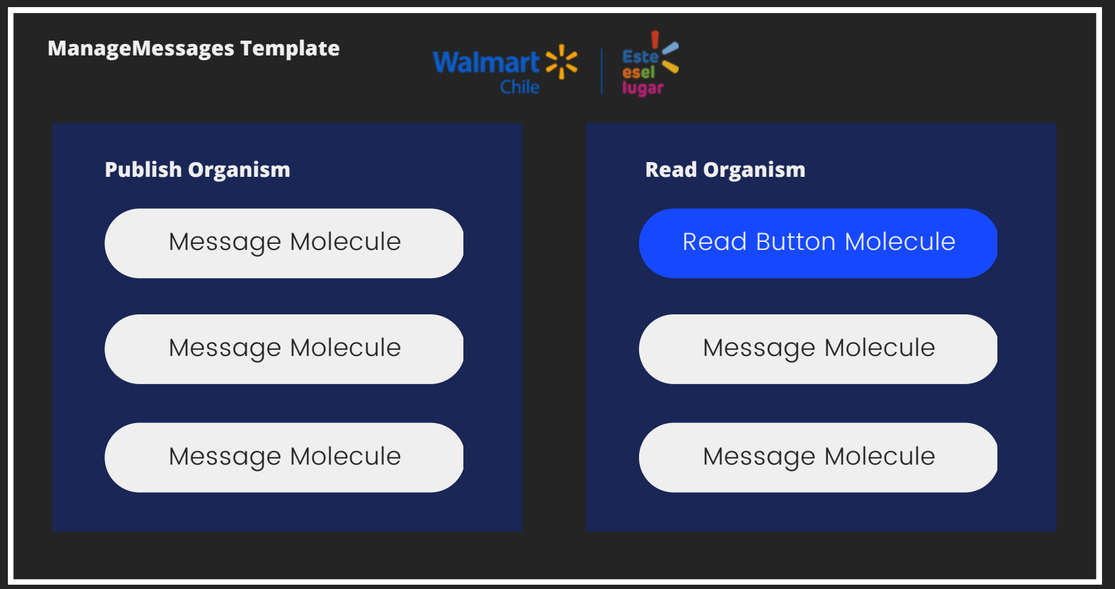
Atomic Design



Átomos Moléculas Organismos Templates Páginas

Analicemos el Sistema





Analicemos el Sistema

ManageMessages Template

Walmart Chile | Este es el lugar

Publish Organism

- Message Molecule
- Message Molecule
- Message Molecule

Read Organism

- Read Button Molecule
- Message Molecule
- Message Molecule

Message Molecule

examples > susblisher > src > molecules > Message > JS index.js > ...

```
1   import Button from '@mui/material/Button';
2
3   const Message = ({ content, onClickHandler }) => {
4     return (
5       <Button onClick={onClickHandler} variant='outlined'>
6         {content}
7       </Button>
8     );
9   };
10
11  export default Message;
```

Message Molecule

examples > susblisher > src > molecules > Message > JS index.js > ...

```
1   import Button from '@mui/material/Button';
2
3   const Message = ({ content, onClickHandler }) => {
4     return (
5       <Button onClick={onClickHandler} variant='outlined'>
6         {content}
7       </Button>
8     );
9   };
10
11  export default Message;
```

Message Molecule

examples > susblisher > src > molecules > Message > JS index.js > ...

```
1   import Button from '@mui/material/Button';
2
3   const Message = ({ content, onClickHandler }) => {
4     return (
5       <Button onClick={onClickHandler} variant='outlined'>
6         {content}
7       </Button>
8     );
9   };
10
11  export default Message;
```

Publish Organism

src > organisms > Publish > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import MessageMolecule from '../../molecules/Message';

const PublishOrganism = ({ messages, sendMessageHandler }) => {
  return (
    <Stack spacing={2}>
      {messages.map((message, i) => (
        <MessageMolecule
          key={`send-message-${i}`}
          onClickHandler={() => sendMessageHandler(message.content)}
          content={message.content}
        />
      ))}
    </Stack>
  );
};

export default PublishOrganism;
```

Publish Organism

Message Molecule

Message Molecule

Message Molecule

src > organisms > Publish > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import MessageMolecule from '../../molecules/Message';

const PublishOrganism = ({ messages, sendMessageHandler }) => {
  return (
    <Stack spacing={2}>
      {messages.map((message, i) => (
        <MessageMolecule
          key={`send-message-${i}`}
          onClickHandler={() => sendMessageHandler(message.content)}
          content={message.content}
        />
      ))}
    </Stack>
  );
};

export default PublishOrganism;
```

Publish Organism

Message Molecule

Message Molecule

Message Molecule

src > organisms > Publish > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import MessageMolecule from '../../molecules/Message';

const PublishOrganism = ({ messages, sendMessageHandler }) => {
  return (
    <Stack spacing={2}>
      {messages.map((message, i) => (
        <MessageMolecule
          key={`send-message-${i}`}
          onClickHandler={() => sendMessageHandler(message.content)}
          content={message.content}
        />
      ))}
    </Stack>
  );
};

export default PublishOrganism;
```

Read Organism

Read Button Molecule

Message Molecule

Message Molecule

src > organisms > Read > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import MessageMolecule from '../../molecules/Message';

const ReadOrganism = ({ messages, readMessagesHandler }) => {
  return (
    <Grid alignContent={'center'}>
      <Box textAlign='center'>
        <Button
          variant='contained'
          onClick={readMessagesHandler}
          sx={{ mb: 2 }}
        >
          Leer Mensajes
        </Button>
      </Box>
      <Grid item>
        <Stack spacing={2}>
          {messages.map((message, i) => (
            <MessageMolecule
              key={`published-message-${i}`}
              content={message.content}
            />
          ))}
        </Stack>
      </Grid>
    </Grid>
  );
};

export default ReadOrganism;
```

Read Organism

Read Button Molecule

Message Molecule

Message Molecule

src > organisms > Read > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import MessageMolecule from '../../molecules/Message';

const ReadOrganism = ({ messages, readMessagesHandler }) => {
  return (
    <Grid alignContent={'center'}>
      <Box textAlign='center'>
        <Button
          variant='contained'
          onClick={readMessagesHandler}
          sx={{ mb: 2 }}
        >
          Leer Mensajes
        </Button>
      </Box>
      <Grid item>
        <Stack spacing={2}>
          {messages.map((message, i) => (
            <MessageMolecule
              key={`published-message-${i}`}
              content={message.content}
            />
          ))}
        </Stack>
      </Grid>
    </Grid>
  );
};

export default ReadOrganism;
```

Read Organism

Read Button Molecule

Message Molecule

Message Molecule

src > organisms > Read > JS index.js > ...

```jsx
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import MessageMolecule from '../../molecules/Message';

const ReadOrganism = ({ messages, readMessagesHandler }) => {
  return (
    <Grid alignContent={'center'}>
      <Box textAlign='center'>
        <Button
          variant='contained'
          onClick={readMessagesHandler}
          sx={{ mb: 2 }}
        >
          Leer Mensajes
        </Button>
      </Box>
      <Grid item>
        <Stack spacing={2}>
          {messages.map((message, i) => (
            <MessageMolecule
              key={`published-message-${i}`}
              content={message.content}
            />
          ))}
        </Stack>
      </Grid>
    </Grid>
  );
};

export default ReadOrganism;
```

ManageMessages Template

Walmart Chile | Este es el lugar

Publish Organism

- Message Molecule
- Message Molecule
- Message Molecule

Read Organism

- Read Button Molecule
- Message Molecule
- Message Molecule

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };

  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };

  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };

  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };

  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };

  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```jsx
import Grid from '@mui/material/Grid';
import { useState } from 'react';
import logo from '../../logo.png';
import PublishOrganism from '../../organisms/Publish';
import ReadOrganism from '../../organisms/Read';

const availableMessages = [
  { content: 'Workshop' },
  { content: 'Walmart' },
  { content: 'Este es el lugar' },
];

const ManageMessages = ({ messageRepository }) => {
  const [messages, setMessages] = useState([]);

  const readMessages = () => {
    messageRepository
      .getMessages()
      .then((newMessages) => setMessages(newMessages));
  };

  const sendMessage = (content) => {
    messageRepository.sendMessage(content).then(() => readMessages());
  };
```

```jsx
  return (
    <Grid container direction='column' alignItems={'center'}>
      <Grid item pb={5}>
        <img src={logo} alt='Walmart Chile'></img>
      </Grid>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <PublishOrganism
            messages={availableMessages}
            sendMessageHandler={sendMessage}
          />
        </Grid>
        <Grid item xs={6}>
          <ReadOrganism
            messages={messages}
            readMessagesHandler={readMessages}
          />
        </Grid>
      </Grid>
    </Grid>
  );
};

export default ManageMessages;
```

src > layouts > ManageMessages > JS index.js > ManageMessages

```
1  import Grid from '@mui/material/Grid';
2  import { useState } from 'react';
3  import logo from '../../logo.png';
4  import PublishOrganism from '../../organisms/Publish';
5  import ReadOrganism from '../../organisms/Read';
6
7  const availableMessages = [
8    { content: 'Workshop' },
9    { content: 'Walmart' },
10   { content: 'Este es el lugar' },
11 ];
12
13 const ManageMessages = ({ messageRepository }) => {
14   const [messages, setMessages] = useState([]);
15
16   const readMessages = () => {
17     messageRepository
18       .getMessages()
19       .then((newMessages) => setMessages(newMessages));
20   };
21
22   const sendMessage = (content) => {
23     messageRepository.sendMessage(content).then(() => readMessages());
24   };
```

```
26   return (
27     <Grid container direction='column' alignItems={'center'}>
28       <Grid item pb={5}>
29         <img src={logo} alt='Walmart Chile'></img>
30       </Grid>
31       <Grid container spacing={2}>
32         <Grid item xs={6}>
33           <PublishOrganism
34             messages={availableMessages}
35             sendMessageHandler={sendMessage}
36           />
37         </Grid>
38         <Grid item xs={6}>
39           <ReadOrganism
40             messages={messages}
41             readMessagesHandler={readMessages}
42           />
43         </Grid>
44       </Grid>
45     </Grid>
46   );
47 };
48
49 export default ManageMessages;
```

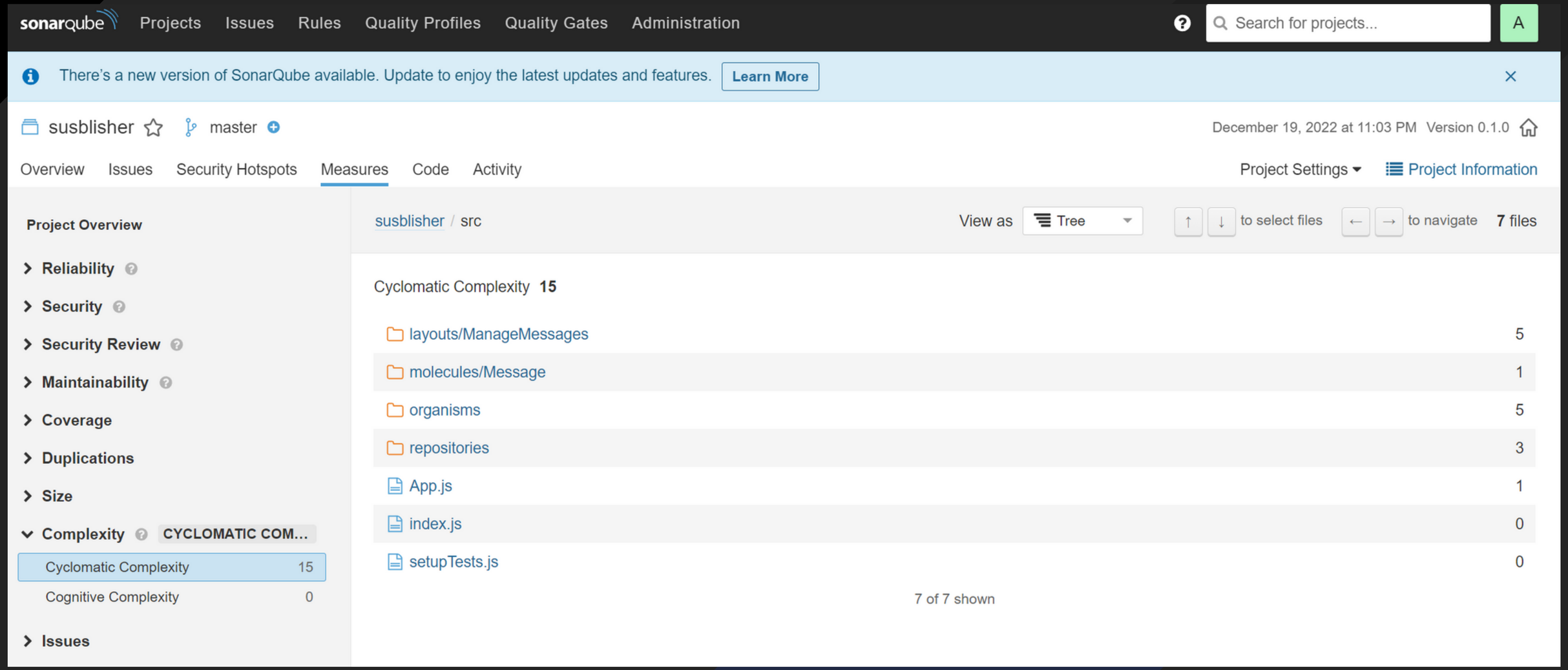
src > JS App.js > ...

```
1  import ManageMessages from './layouts/ManageMessages';
2  import { messageRepository } from './repositories';
3  import { Container } from '@mui/system';
4  import { ThemeProvider, createTheme } from '@mui/material/styles';
5  import CssBaseline from '@mui/material/CssBaseline';
6
7  const darkTheme = createTheme({
8    palette: {
9      mode: 'dark',
10   },
11 });
12
13 function App() {
14   return (
15     <ThemeProvider theme={darkTheme}>
16       <CssBaseline />
17       <Container sx={{ pt: 5 }}>
18         <ManageMessages messageRepository={messageRepository}></ManageMessages>
19       </Container>
20     </ThemeProvider>
21   );
22 }
23
24 export default App;
25
```

src > JS App.js > ...

```
1   import ManageMessages from './layouts/ManageMessages';
2   import { messageRepository } from './repositories';
3   import { Container } from '@mui/system';
4   import { ThemeProvider, createTheme } from '@mui/material/styles';
5   import CssBaseline from '@mui/material/CssBaseline';
6
7   const darkTheme = createTheme({
8     palette: {
9       mode: 'dark',
10    },
11  });
12
13  function App() {
14    return (
15      <ThemeProvider theme={darkTheme}>
16        <CssBaseline />
17        <Container sx={{ pt: 5 }}>
18          <ManageMessages messageRepository={messageRepository}></ManageMessages>
19        </Container>
20      </ThemeProvider>
21    );
22  }
23
24  export default App;
25
```

Vamos la Complejidad Ciclomática del Sistema Completo



Testing luego de toda esta modificación

examples > susblisher-basic > src > __tests__ > unit > JS App.test.js > ...

```
1   import { render, screen } from '@testing-library/react';
2   import App from '../../App';
3
    Run | Debug
4   describe('App unit testing suite', () => {
      Run | Debug
5     it('Should render the list of messages', () => {
6       render(<App />);
7       const message = screen.getByText('Walmart');
8       expect(message).toBeInTheDocument();
9     });
10  });
```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > unit > Molecules > Message > JS index.test.js > ...

```
1   import Message from '../../../../molecules/Message';
2   import { render, screen, fireEvent } from '@testing-library/react';
3
    Run | Debug
4   describe('Message molecule unit testing', () => {
      Run | Debug
5     it('Should render a message with the given content', () => {
6       const expectedContent = 'Hola';
7       render(<Message content={expectedContent} />);
8       expect(screen.getByText(expectedContent)).toBeInTheDocument();
9     });
10
      Run | Debug
11    it('Should call the handler when the button is clicked', () => {
12      const buttonText = 'Click me';
13      const onClickHandlerSpy = jest.fn();
14
15      render(<Message content={buttonText} onClickHandler={onClickHandlerSpy} />);
16      const button = screen.getByText(buttonText);
17      fireEvent.click(button);
18
19      expect(onClickHandlerSpy).toBeCalled();
20    });
21  });
22
```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > functional > JS PublishMessage.test.js > ...

```
import { render, fireEvent, screen } from '@testing-library/react';
import App from '../../App';
import nock from 'nock';

Run | Debug
test('Should publish a message successfully', async () => {
  render(<App />);
  const baseURL = 'http://localhost:8000';
  const testMessage = 'Prueba Mock';
  const mockResponse = [{ content: testMessage }];
  const corsHeaders = {
    'Access-Control-Allow-Methods':
      'PUT, OPTIONS, CONNECT, PATCH, GET, HEAD, POST, DELETE, TRACE',
    'Access-Control-Allow-Origin': 'http://localhost',
    'Access-Control-Expose-Headers': 'link, etag, location',
    'Access-Control-Allow-Headers': 'user-agent',
  };

  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).post('/messages').reply(200, null, corsHeaders);
  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).get('/messages').reply(200, mockResponse, corsHeaders);

  const firstMessageButton = screen.getAllByRole('button')[0];
  await fireEvent.click(firstMessageButton);
  const expectedMessage = await screen.findByText(testMessage);

  expect(expectedMessage).toBeInTheDocument();
});

```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > functional > JS PublishMessage.test.js > ...

```
import { render, fireEvent, screen } from '@testing-library/react';
import App from '../../App';
import nock from 'nock';

Run | Debug
test('Should publish a message successfully', async () => {
  render(<App />);
  const baseURL = 'http://localhost:8000';
  const testMessage = 'Prueba Mock';
  const mockResponse = [{ content: testMessage }];
  const corsHeaders = {
    'Access-Control-Allow-Methods':
      'PUT, OPTIONS, CONNECT, PATCH, GET, HEAD, POST, DELETE, TRACE',
    'Access-Control-Allow-Origin': 'http://localhost',
    'Access-Control-Expose-Headers': 'link, etag, location',
    'Access-Control-Allow-Headers': 'user-agent',
  };

  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).post('/messages').reply(200, null, corsHeaders);
  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).get('/messages').reply(200, mockResponse, corsHeaders);

  const firstMessageButton = screen.getAllByRole('button')[0];
  await fireEvent.click(firstMessageButton);
  const expectedMessage = await screen.findByText(testMessage);

  expect(expectedMessage).toBeInTheDocument();
});

```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > functional > JS PublishMessage.test.js > ...

```
import { render, fireEvent, screen } from '@testing-library/react';
import App from '../../App';
import nock from 'nock';

Run | Debug
test('Should publish a message successfully', async () => {
  render(<App />);
  const baseURL = 'http://localhost:8000';
  const testMessage = 'Prueba Mock';
  const mockResponse = [{ content: testMessage }];
  const corsHeaders = {
    'Access-Control-Allow-Methods':
      'PUT, OPTIONS, CONNECT, PATCH, GET, HEAD, POST, DELETE, TRACE',
    'Access-Control-Allow-Origin': 'http://localhost',
    'Access-Control-Expose-Headers': 'link, etag, location',
    'Access-Control-Allow-Headers': 'user-agent',
  };

  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).post('/messages').reply(200, null, corsHeaders);
  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).get('/messages').reply(200, mockResponse, corsHeaders);

  const firstMessageButton = screen.getAllByRole('button')[0];
  await fireEvent.click(firstMessageButton);
  const expectedMessage = await screen.findByText(testMessage);

  expect(expectedMessage).toBeInTheDocument();
});
```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > functional > JS PublishMessage.test.js > ...

```
import { render, fireEvent, screen } from '@testing-library/react';
import App from '../../App';
import nock from 'nock';

Run | Debug
test('Should publish a message successfully', async () => {
  render(<App />);
  const baseURL = 'http://localhost:8000';
  const testMessage = 'Prueba Mock';
  const mockResponse = [{ content: testMessage }];
  const corsHeaders = {
    'Access-Control-Allow-Methods':
      'PUT, OPTIONS, CONNECT, PATCH, GET, HEAD, POST, DELETE, TRACE',
    'Access-Control-Allow-Origin': 'http://localhost',
    'Access-Control-Expose-Headers': 'link, etag, location',
    'Access-Control-Allow-Headers': 'user-agent',
  };

  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).post('/messages').reply(200, null, corsHeaders);
  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).get('/messages').reply(200, mockResponse, corsHeaders);

  const firstMessageButton = screen.getAllByRole('button')[0];
  await fireEvent.click(firstMessageButton);
  const expectedMessage = await screen.findByText(testMessage);

  expect(expectedMessage).toBeInTheDocument();
});

```

Testing luego de toda esta modificación

examples > susblisher > src > __tests__ > functional > JS PublishMessage.test.js > ...

```
import { render, fireEvent, screen } from '@testing-library/react';
import App from '../../App';
import nock from 'nock';

Run | Debug
test('Should publish a message successfully', async () => {
  render(<App />);
  const baseURL = 'http://localhost:8000';
  const testMessage = 'Prueba Mock';
  const mockResponse = [{ content: testMessage }];
  const corsHeaders = {
    'Access-Control-Allow-Methods':
      'PUT, OPTIONS, CONNECT, PATCH, GET, HEAD, POST, DELETE, TRACE',
    'Access-Control-Allow-Origin': 'http://localhost',
    'Access-Control-Expose-Headers': 'link, etag, location',
    'Access-Control-Allow-Headers': 'user-agent',
  };

  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).post('/messages').reply(200, null, corsHeaders);
  nock(baseURL).options('/messages').reply(200, null, corsHeaders);
  nock(baseURL).get('/messages').reply(200, mockResponse, corsHeaders);

  const firstMessageButton = screen.getAllByRole('button')[0];
  await fireEvent.click(firstMessageButton);
  const expectedMessage = await screen.findByText(testMessage);

  expect(expectedMessage).toBeInTheDocument();
});

```

Conclusiones

1. Forzar estándares de programación.
2. Medir constantemente las métricas de mantenibilidad.
3. Apoyarnos en las herramientas de mantenibilidad.
4. Incentivar al ecosistema a crear software mantenible.

Otras Recomendaciones

"Is this an atom or a molecule?, what the hell is it, Jimmy?"

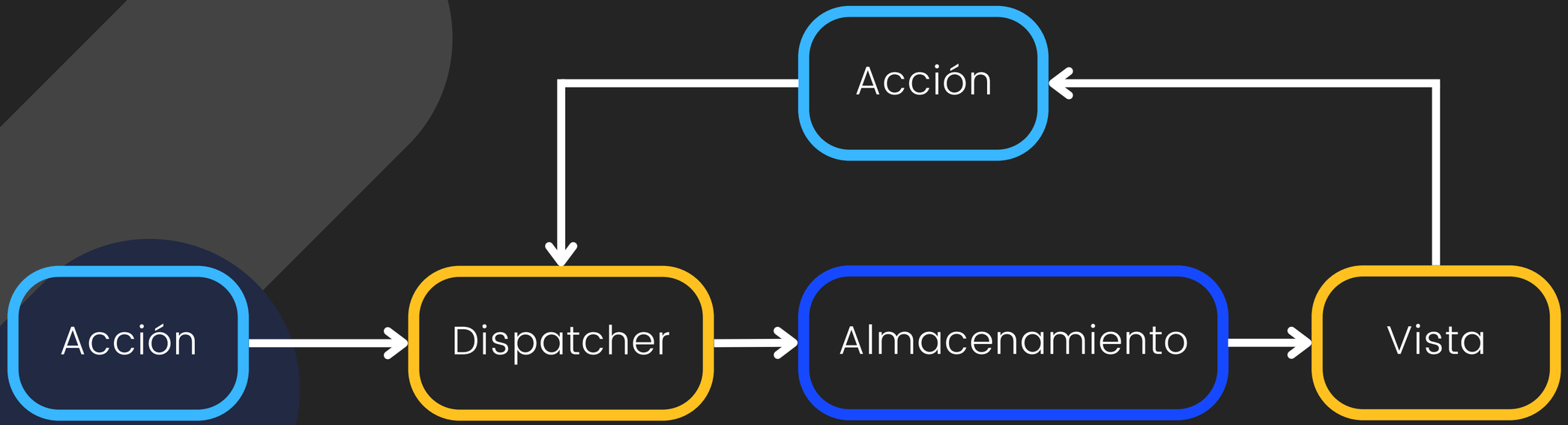
James Eccleston
"What's wrong with atomic design?"

“Your architectures should tell readers about the system, not about the frameworks you used in your system.”

Robert C. Martin
Conclusion from "Screaming Architecture" - The Clean Code Blog

Patrón de Arquitectura FLUX

Utilizar patrones arquitectónicos recomendados para aplicaciones web como FLUX.



Librerias para manejo de estados

Aprovechar soluciones que vienen listas para usar como Zustand, para resolver diveros problemas como manejo de estados.

Herramientas de Testing

Utilizar las mejores herramientas que ofrece el mercado para la testeabilidad del software.

Testing Library

Cypress

Bibliografía

- Paul Stachour and David Collier-Brown. 2009. You Don't Know Jack About Software Maintenance: Long considered an afterthought, software maintenance is easiest and most effective when built into a system from the ground up. Queue 7, 9 (October 2009), 50–55. https://doi.org/10.1145/1626135.1640399
- Hanenberg, S., Kleinschmager, S., Robbes, R. et al. An empirical study on the impact of static typing on software maintainability. Empir Software Eng 19, 1335–1382 (2014). https://doi.org/10.1007/s10664-013-9289-1
- Tupeli, Pauli (2020). Ensuring maintainability of JavaScript web applications. https://trepo.tuni.fi/handle/10024/123717
- Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, International Organization for Standardization, 2011
- "IEEE Standard Glossary of Software Engineering Terminology," in IEEE Std 610.12-1990 , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064.
- https://www.sonarqube.org/
- https://create-react-app.dev/
- https://mui.com/
- https://www.linkedin.com/pulse/whats-wrong-atomic-design-james-eccleston/
- https://blog.cleancoder.com/uncle-bob/2011/09/30/Screaming-Architecture.html

Walmart

Mantenibilidad de Software en una Aplicación en Javascript

Acceso al repositorio de este workshop.

Gonzalo Fernández - Senior FulStack Developer

vadokdev VadokDev VadokDev

María Paz Morales - FullStack Developer

paz-morales-llopis paz52