

Ⓞ Problema 1:

¿Puedes mostrar estas características de los usuarios de manera ordenada y coherente en consola? ¿Tienen los usuarios alguna funcionalidad, pueden hacer algo?

No se pueden mostrar las características de forma ordenada, pueden ser mostradas en un orden completamente diferente dependiendo de lo que se ponga en la consola. Los usuarios u objetos no tienen ninguna funcionalidad por el momento, por lo tanto, no pueden hacer nada

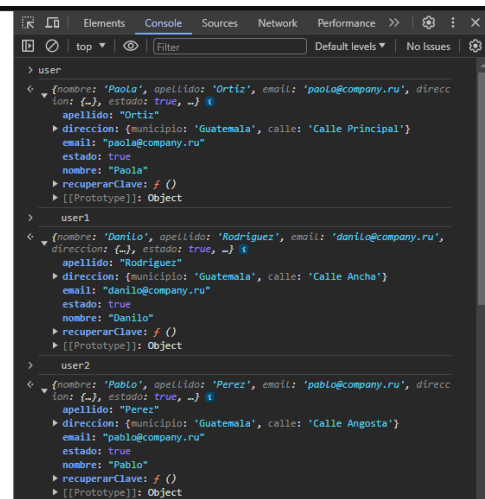


## ⌘ Problema 2:

¿Qué diferencias conceptuales observas entre el Problema I y el Problema II? ¿Para crear usuarios es más fácil y coherente la manera del Problema I o la manera en que se crean en el Problema II?

La diferencia principalmente es el orden de la estructura de objetos (usuarios), es más fácil verificar los datos por cada objeto y más completa la información al poner cada uno de ellos en consola, a diferencia del problema 1 que había que llamar cada una de las propiedades del objeto para verificar detalle por detalle.

### Tarea 5 JS

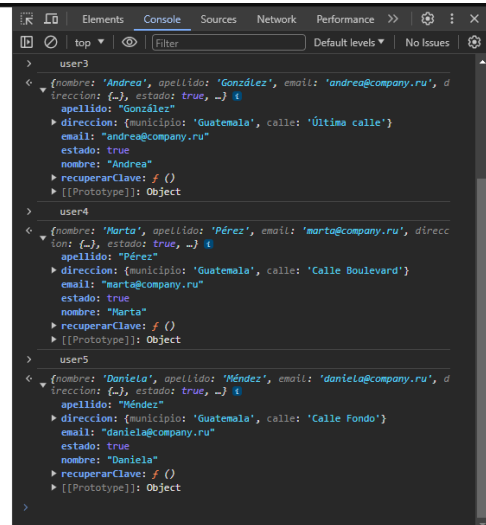


```
> user
{
  nombre: 'Paola',
  apellido: 'Ortiz',
  email: 'paula@company.ru',
  direccion: {
    municipio: 'Guatemala',
    calle: 'Calle Principal'
  },
  estado: true,
  recuperarClave: f()
}

> user1
{
  nombre: 'Danilo',
  apellido: 'Rodriguez',
  email: 'danilo@company.ru',
  direccion: {
    municipio: 'Guatemala',
    calle: 'Calle Ancha'
  },
  estado: true,
  recuperarClave: f()
}

> user2
{
  nombre: 'Pablo',
  apellido: 'Perez',
  email: 'pablo@company.ru',
  direccion: {
    municipio: 'Guatemala',
    calle: 'Calle Angosta'
  },
  estado: true,
  recuperarClave: f()
}
```

## Tarea 5 JS



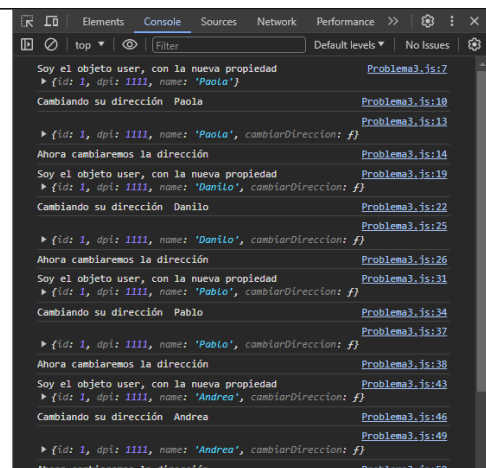
```
<
>
user3
  {
    nombre: 'Andrea', apellido: 'González', email: 'andrea@company.ru', direccion: {municipio: 'Guatemala', calle: 'Última calle'}, estado: true, nombre: 'Andrea', recuperarClave: f ()
  }
user4
  {
    nombre: 'Marta', apellido: 'Pérez', email: 'marta@company.ru', direccion: {municipio: 'Guatemala', calle: 'Calle Boulevard'}, estado: true, nombre: 'Marta', recuperarClave: f ()
  }
user5
  {
    nombre: 'Daniela', apellido: 'Méndez', email: 'daniela@company.ru', direccion: {municipio: 'Guatemala', calle: 'Calle Fondo'}, estado: true, nombre: 'Daniela', recuperarClave: f ()
  }
>
```

### ⌚ Problema 3:

Crea otro código utilizando el Problema II, pero ahora agrega a cada usuario (objeto) la propiedad dpi y el método cambiarDireccion. Utiliza el ejemplo 05-Objetos/02-dinamico.js como guía para solucionar este problema.

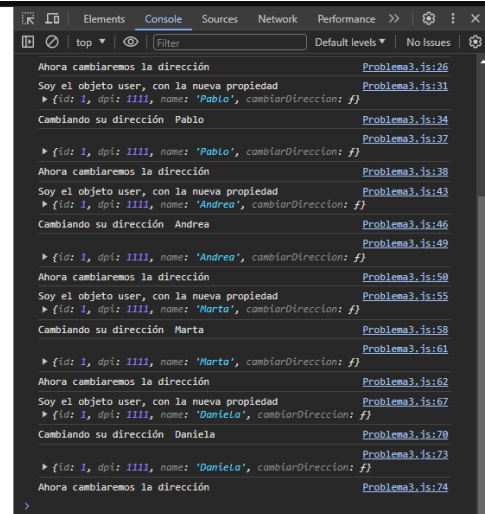
Muestra a cada usuario en la consola.

## Tarea 5 JS



```
Soy el objeto user, con la nueva propiedad Problema3.js:7
  {id: 1, dpi: 1111, name: 'Paola'}
Cambiando su dirección Paola Problema3.js:10
  {id: 1, dpi: 1111, name: 'Paola', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:14
Soy el objeto user, con la nueva propiedad Problema3.js:19
  {id: 1, dpi: 1111, name: 'Danilo', cambiarDireccion: f}
Cambiando su dirección Danilo Problema3.js:22
  {id: 1, dpi: 1111, name: 'Danilo', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:26
Soy el objeto user, con la nueva propiedad Problema3.js:31
  {id: 1, dpi: 1111, name: 'Pablo', cambiarDireccion: f}
Cambiando su dirección Pablo Problema3.js:34
  {id: 1, dpi: 1111, name: 'Pablo', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:38
Soy el objeto user, con la nueva propiedad Problema3.js:43
  {id: 1, dpi: 1111, name: 'Andrea', cambiarDireccion: f}
Cambiando su dirección Andrea Problema3.js:46
  {id: 1, dpi: 1111, name: 'Andrea', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:50
```

## Tarea 5 JS



```

Ahora cambiaremos la dirección Problema3.js:26
Soy el objeto user, con la nueva propiedad Problema3.js:31
  {id: 1, dpi: 1111, name: 'Pablo', cambiarDireccion: f}
Cambiando su dirección Pablo Problema3.js:34
  {id: 1, dpi: 1111, name: 'Pablo', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:38
Soy el objeto user, con la nueva propiedad Problema3.js:43
  {id: 1, dpi: 1111, name: 'Andrea', cambiarDireccion: f}
Cambiando su dirección Andrea Problema3.js:46
  {id: 1, dpi: 1111, name: 'Andrea', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:50
Soy el objeto user, con la nueva propiedad Problema3.js:55
  {id: 1, dpi: 1111, name: 'Marta', cambiarDireccion: f}
Cambiando su dirección Marta Problema3.js:58
  {id: 1, dpi: 1111, name: 'Marta', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:62
Soy el objeto user, con la nueva propiedad Problema3.js:67
  {id: 1, dpi: 1111, name: 'Daniela', cambiarDireccion: f}
Cambiando su dirección Daniela Problema3.js:70
  {id: 1, dpi: 1111, name: 'Daniela', cambiarDireccion: f}
Ahora cambiaremos la dirección Problema3.js:74

```

Entre problema podemos verificar a cada uno de los objetos (que son los usuarios en este caso) y que cada uno tiene una nueva propiedad recién añadida (dpi) y que también el método se cambió a cambiarDirección, por lo tanto llama su función y después de crearla, se llama y ahí es cuando avisa que la dirección se está cambiando.

### α Problema 4:

Utilizando las propiedades y métodos agrupados en cada usuario creado en el Problema III, vuelve a crear esos usuarios (objetos) pero en esta ocasión crea los usuarios (objetos) utilizando una factory function.

Aquí creamos una factory function con la función devolverUser para que nos regrese la información que se le pide de cada uno de los usuarios u objetos en este caso, y que aparezca en consola.

Muestra a los usuarios en la consola.

Tarea 5 JS

```

Problema4.js:25
{
  id: 0, nombre: 'Paola', apellido: 'Ortiz', email: 'pao@company.ru',
  cambiarDireccion: f()
  apellido: 'Ortiz'
  ▶ cambiarDireccion: f()
  email: 'pao@company.ru'
  id: 0
  nombre: 'Paola'
  ▶ [[Prototype]]: Object
}

Problema4.js:26
{
  id: 1, nombre: 'Danilo', apellido: 'Rodriguez', email: 'danilo@compan
  y.ru', cambiarDireccion: f()
  apellido: 'Rodriguez'
  ▶ cambiarDireccion: f()
  email: 'danilo@company.ru'
  id: 1
  nombre: 'Danilo'
  ▶ [[Prototype]]: Object
}

Problema4.js:27
{
  id: 2, nombre: 'Pablo', apellido: 'Pérez', email: 'pablo@company.ru',
  cambiarDireccion: f()
  apellido: 'Pérez'
  ▶ cambiarDireccion: f()
  email: 'pablo@company.ru'
  id: 2
  nombre: 'Pablo'
  ▶ [[Prototype]]: Object
}

```

Tarea 5 JS

```

y.ru', cambiarDireccion: f()
}

Problema4.js:27
{
  id: 2, nombre: 'Pablo', apellido: 'Pérez', email: 'pablo@company.ru',
  cambiarDireccion: f()
}

Problema4.js:28
{
  id: 3, nombre: 'Andrea', apellido: 'González', email: 'pablo@company.
  ru', cambiarDireccion: f()
  apellido: 'González'
  ▶ cambiarDireccion: f()
  email: 'pablo@company.ru'
  id: 3
  nombre: 'Andrea'
  ▶ [[Prototype]]: Object
}

Problema4.js:29
{
  id: 4, nombre: 'Marta', apellido: 'Pérez', email: 'marta@company.ru',
  cambiarDireccion: f()
  apellido: 'Pérez'
  ▶ cambiarDireccion: f()
  email: 'marta@company.ru'
  id: 4
  nombre: 'Marta'
  ▶ [[Prototype]]: Object
}

Problema4.js:30
{
  id: 5, nombre: 'Daniela', apellido: 'Méndez', email: 'daniela@compan
  y.ru', cambiarDireccion: f()
  apellido: 'Méndez'
  ▶ cambiarDireccion: f()
  email: 'daniela@company.ru'
  id: 5
  nombre: 'Daniela'
  ▶ [[Prototype]]: Object
}

```