



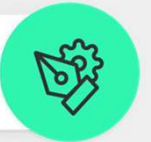
MOCKS Y NORMALIZACIÓN

MOCKS

Formato: link a un repositorio en Github con el proyecto cargado.

Sugerencia: no incluir los node_modules

Desafío
entregable



>> Consigna 1:

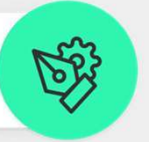
Sobre el desafío entregable de la clase 8 (sql y node: nuestra primera base de datos), crear una vista en forma de tabla que consuma desde la ruta '/api/productos-test' del servidor una lista con 5 **productos** generados al azar utilizando **Faker.js** como generador de información aleatoria de test (en lugar de tomarse desde la base de datos). Elegir apropiadamente los temas para conformar el objeto 'producto' (nombre, precio y foto).

NORMALIZACIÓN

Formato: link a un repositorio en Github con el proyecto cargado.

Sugerencia: no incluir los node_modules

Desafío
entregable



>> Consigna 2:

Ahora, vamos a **reformatar el formato de los mensajes** y la forma de comunicación del chat (centro de mensajes).

El nuevo formato de mensaje será:

```
{
  author: {
    id: 'mail del usuario',
    nombre: 'nombre del usuario',
    apellido: 'apellido del usuario',
    edad: 'edad del usuario',
    alias: 'alias del usuario',
    avatar: 'url avatar (foto, logo) del usuario'
  },
  text: 'mensaje del usuario'
}
```

CODER HOUSE

NORMALIZACIÓN

Formato: link a un repositorio en Github con el proyecto cargado.

Sugerencia: no incluir los node_modules

Desafío
entregable



>> Aspectos a incluir en el entregable:

1. Modificar la persistencia de los mensajes para que utilicen un contenedor que permita guardar objetos anidados (archivos, mongodb, firebase).
2. El mensaje se envía del frontend hacia el backend, el cual lo almacenará en la base de datos elegida. Luego cuando el cliente se conecte o envíe un mensaje, recibirá un **array de mensajes** a representar en su vista.
3. El array que se devuelve debe estar **normalizado con normalizr**, conteniendo una entidad de autores. Considerar que el array tiene sus autores con su correspondiente id (mail del usuario), pero necesita incluir para el proceso de normalización un **id para todo el array** en su conjunto (podemos asignarle nosotros un valor fijo).

Ejemplo: { id: 'mensajes', mensajes: [] }

1. El frontend debería poseer el **mismo esquema de normalización** que el backend, para que este pueda desnormalizar y presentar la información adecuada en la vista.

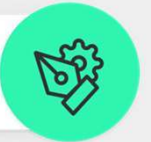
CODER HOUSE

NORMALIZACIÓN

Formato: link a un repositorio en Github con el proyecto cargado.

Sugerencia: no incluir los node_modules

Desafío
entregable



5. Considerar que se puede **cambiar el nombre del id** que usa normalizr, agregando un tercer parametro a la función schema.Entity, por ejemplo:

```
const schemaAuthor = new schema.Entity('author', {...}, {idAttribute: 'email'});
```

En este schema cambia el nombre del id con que se normaliza el nombre de los autores a 'email'. Más info en la [web oficial](#).

5. Presentar en el frontend (a modo de test) el **porcentaje de compresión** de los mensajes recibidos. Puede ser en el título del centro de mensajes.

>> **Nota:** incluir en el frontend el script de normalizr de la siguiente cdn:

<https://cdn.jsdelivr.net/npm/normalizr@3.6.1/dist/normalizr.browser.min.js>

Así podremos utilizar los mismos métodos de normalizr que en el backend. Por ejemplo: new normalizr.schema.Entity , normalizr.denormalize(...,...,...)

CODER HOUSE


Desafío
entregable





Formulario de Ingreso de Producto x +


localhost:8080

Centro de Mensajes (Compresión: 62%)

d@s [28/3/2021 12:43:34] : *holis!!!* 

g@s [28/3/2021 12:44:16] : *Holaaaaaaaaaaaaa* 

g@s [28/3/2021 12:44:33] : *Como están!* 

c@s [28/3/2021 12:45:05] : *heyyyyyyyyyy* 

Application Network Console Elements Sources >> ⚙️ ⋮ ✕

top Filter Default levels ⚙️

```
▶ {entities: {...}, result: "mensajes"} 7343 chat.js:20
▶ {id: "mensajes", mensajes: Array(44)} 11644 chat.js:25
Porcentaje de compresión 63% chat.js:28
chat.js:20
▼ {entities: {...}, result: "mensajes"} ⓘ
  ▼ entities:
    ▼ author:
      ▶ c@s: {email: "c@s", nombre: "Cecilia", apellido: "Sanchez", edad: ...
      ▶ d@s: {email: "d@s", nombre: "Daniel", apellido: "Sanchez", edad: "...
      ▶ g@s: {email: "g@s", nombre: "Gabriela", apellido: "Sanchez", edad: ...
      ▶ __proto__: Object
    ▶ post: {6060a426412d552b8cae776d: {...}, 6060a450412d552b8cae776e: {...}, ...
    ▼ posts:
      ▶ mensajes: {id: "mensajes", mensajes: Array(45)}
      ▶ __proto__: Object
    ▶ __proto__: Object
    result: "mensajes"
    ▶ __proto__: Object
  7497
chat.js:25
▼ {id: "mensajes", mensajes: Array(45)} ⓘ
  id: "mensajes"
  ▶ mensajes: (45) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, ...
  ▶ __proto__: Object
  11939
Porcentaje de compresión 62% chat.js:28
>
```

CODER HOUSE