Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

Tarea 3

IIC2026 — *Visualización de información*
Entrega: 29 de octubre del 2017, 23:59

Esta tarea pretende ser un pequeño acercamiento a los módulos avanzados de **D3.js** repasados en ayudantía. Esta tarea es **individual** y se entrega en una carpeta separada en su repositorio asignado de GitHub.

1. Visualización de dependencias de módulos de D3.js (6 puntos)

1.1. Objetivo

Esta tarea es la continuación a lo realizado en nuestra **novena ayudantía**. Deben generar una visualización, donde se plasmen las dependencias entre los distintos módulos de **D3.js**. Modelaremos esta relación como un grafo **dirigido** de dependencias $G = (V, E)$. Cada nodo $v \in V$ representa un módulo de **D3.js** y las aristias representan la relación de dependencia: si v tiene como dependencia a u entonces $(u, v) \in E$. En Siding se encuentra disponible un archivo **JSON** con esta relación; su contenido se extrajo del **respositorio público** de **D3.js** en GitHub. La estructura del **dataset.json** es:

- El atributo ***nodes*** contiene los distintos módulos, se especifica para cada uno:
  - ***name*** – nombre del módulo,
  - ***forks*** – cantidad de *forks* del módulo en GitHub,
  - ***stars*** – cantidad de estrellas del módulo en GitHub.
- El atributo ***links*** contiene las distintas dependencias, se especifica para cada una:
  - ***source*** – nombre de módulo del cual se depende.
  - ***target*** – nombre de módulo que depende de ***source***.
  - ***dev*** – valor booleano que especifica si es una ***devDependency***.

1.2. Características

Su visualización debe cumplir con las siguientes características de representación e interacción:

- Cada módulo debe ser representado por un nodo.
- Cada nodo debe mostrar, de alguna forma, los atributos ***name***, ***forks*** y ***stars*** del módulo que representa.
- Debe utilizar algún canal sobre el nodo para representar la cantidad de ***forks*** del módulo que representa.
- Debe utilizar algún canal sobre el nodo para representar la cantidad de ***stars*** del módulo que representa.
- Debe utilizar algún canal sobre el nodo para diferenciar aquellos módulos que no dependen de ningún otro, aquellos que sólo dependen de otros módulos y el resto.
- Cada dependencia debe ser representada mediante una arista y debe mostrar la dirección de dependencia.
- Debe utilizar algún canal sobre las aristas para diferenciar ***devDependencies*** de ***dependencies*** regulares (o más bien, dependencias que no son ***devDependencies***).
- Debe poder arrastrar cada nodo con el cursor (i.e. ***drag***).
- Al arrastrar un nodo, su posición debe quedar fija incluso si se deja de arrastrar. Sólo si se hace doble clic sobre el nodo, se libera su posición.
- Al pasar el cursor sobre un nodo (i.e. ***hover***), la visualización debe realizar énfasis sobre ese nodo, sobre los nodos relacionados y las aristas que los conectan.
- La visualización debe estar calibrada de tal forma que el posicionamiento inicial (o próximo al inicial) es lo menos confuso o enredado posible.

1.3. Consideraciones

Para esta tarea, **deben** utilizar **D3.js** en su **última versión** embebido en un informe, escrito en **HTML+CSS**, que muestre los resultados. Por otra parte, **debe** utilizar **d3-force** para simular el grafo de dependencias, y **debe** implementarse utilizando **SVG** y **no** el **Canvas API**. Nuevamente, en esta instancia, se espera que las decisiones de visualización —entiéndase como toda decisión de *visual encoding*

tomada en la visualización— estén fundamentadas correctamente en base a lo visto en el curso, ya sea, en términos de abstracción de datos, percepción, *rules of thumb*, etcétera. Además, **debe documentar** estas decisiones en el informe final entregado. Y, al igual que para la tarea anterior, se espera un buen nivel estético para este informe.

1.4. Entregables

Los entregables de esta tarea son:

- Un documento escrito en **HTML+CSS** que sirva como presentación o informe final de la visualización solicitada.
- Todo archivo para la correcta apertura y funcionamiento del ítem anterior.
- Un documento **README.md** con las instrucciones necesarias e información relevante sobre el trabajo realizado. Los *emoji* son bienvenidos.

Recuerde que estos entregables deben subirse en una **carpeta separada** en su repositorio privado asignado de GitHub.

1.5. Atrasos

Se practicará una política de atrasos de nota decreciente por número de días de atraso. Por cada día, se descontará **un punto** del máximo posible. Es decir, la nota obtenida se escala según el nuevo máximo permitido, por la cantidad de días de atraso. Puede pensarlo como la siguiente escala de **D3.js** que recibe la nota sin descuentos:

```
const markScale = d3.scaleLinear()
                    .domain([1, 7])
                    .range([1, d3.max([1, 7 - daysAfterDeadline])])
```