Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2026 — Visualización de Información

Tarea 2

D3.js avanzado

Fecha de inicio: **Viernes 05 de abril** a las **20:00:00** hrs

Fecha de entrega: **Sábado 20 de abril** a las **20:00:00** hrs.

Distribución de puntaje: **En este enlace.**

Evaluación en el contexto del curso

Esta evaluación es de naturaleza **sumativa** e **individual**, y pretende rescatar evidencias del desarrollo de ciertos resultados de aprendizaje. Por eso, tras su realización y entrega, recibirás retroalimentación sobre su desempeño y una nota que lo refleja. El material del curso relevante para esta evaluación es el siguiente:

1. **Tecnologías web**: Introducción práctica a las tecnologías web HTML, CSS y JavaScript.
2. **SVG**: Formato de gráfico vectorial y bidimensionales para diseñar las visualizaciones.
3. **Librería D3.js**: Librería de bajo nivel especializada para la creación de visualizaciones de información en conjunto a tecnologías web.

1. Implementación de herramientas de visualización

Esta evaluación busca que implementes, mediante programación y tecnologías web, una herramienta de visualización **artística** e interactiva para resolver una necesidad de comunicación de información puntual. Las tecnologías a utilizar en esta implementación son: HTML, CSS, SVG, JavaScript y D3.js.

Específicamente, se debe entregar como resultado dos documentos HTML, uno que introduzca la herramienta, y otro que produzca la herramienta de visualización artística. Esta herramienta deberá ser interactiva y basada únicamente en SVG para la situación puntual propuesta. El estilaje del documento se determina mediante declaraciones de CSS y funciona gracias a un programa escrito en JavaScript que utiliza la librería D3.js.

Además, un problema común en el desarrollo de visualizaciones es continuar trabajo heredado de otras personas sin empezar desde cero todo el trabajo. Por lo cual, en esta tarea te enfrentarás a tener que completar un código que ya viene con ciertos elementos y lógicas programados, y que no podrás modificar.

1.1. Visualización Dragon Ball

Gracias a la capacidad de D3.js de confeccionar visualizaciones desde el bajo nivel (dibujando cada figura y coordinando estas con los datos), las herramientas de visualización posibles son ilimitadas. Dentro de estas, las visualizaciones artísticas tienen un gran valor para poner a prueba las capacidades de D3.js y construir, de forma automática, visualizaciones diferentes a las que se pueden hacer con *software* de visualización.

Para esta ocasión, en honor al mangaka Akira Toriyama[1], vamos a presentar de una forma más artística información sobre una de sus creaciones más conocida Dragon Ball. En este contexto, lo único que se necesita saber que Dragon Ball tiene diferentes series (que se pueden ver como diferentes temporadas). Cada serie tiene diferentes aventuras (conocidas como "sagas") y, cómo una serie de acción y peleas, los personajes van aumentando su poder paulatinamente para ser más poderosos.

Dada esta información, por un lado queremos que las personas puedan explorar esta página, por medio de una visualización, para conocer tres series relacionadas a Dragon Ball (Dragon Ball, Dragon Ball Z y Dragon Ball GT) y mostrarles un resumen de información básica de cada serie como la cantidad de personajes, la cantidad de sagas/aventuras que tuvo la serie, etc. Por otro lado, dado una serie en particular, queremos presentar de una forma diferente la información de los personajes pertenecientes a dicha serie. Dentro de esta presentación se desea que las personas puedan identificar en qué serie aparece más el personaje, cantidad de sagas/aventuras en las que participaron, entre otros.

Para esta evaluación se utilizarán dos *datasets*. Uno sobre `las series` y otro sobre `personajes`. Ambos fueron generados a partir de la información de ***este dataset*** y procesado[2] para dejar una versión limpia y fácil de utilizar en la herramienta.

El primer *dataset* contiene la información de las tres series: Dragon Ball, Dragon Ball Z y Dragon Ball GT. El segundo, contiene la información de los diferentes personajes que aparecen en las tres series. No debes modificar estos *datasets* base y cualquier procesamiento requerido, por ejemplo filtros o transformación de datos, debe hacerse en el código una vez cargada la información.

Cada elemento de `db_series.csv` representa una serie y contiene las siguientes propiedades

Propiedad	Descripción
`serie`	Nombre de la serie.
`aventuras`	Cantidad de aventuras vividas en esa serie. Esto también es conocido como "Sagas" o "Arcos".
`personajes_recurrentes`	Cantidad de personajes que aparecen de forma recurrente en la serie.
`personajes_extras`	Cantidad de personajes que aparecen como extras en la serie.
`manga`	Indica si la serie está basada en un manga o es trabajo original.
`cantidad_caps`	Cantidad de capítulos de la serie.

[1] Quién falleció el 1 de marzo del 2024

[2] El procesamiento incluyó filtrado, derivación de nuevos atributos, entre otros.

Cada elemento de `db_personajes.csv` representa un personaje y contiene las siguientes propiedades:

Propiedad	Descripción
`personaje`	Nombre del personaje. Este dato es único.
`aventuras`	Cantidad de aventuras donde participó el personaje.
`primera_serie`	Nombre de la serie donde apareció por primera vez el personaje.
`serie_recurrente`	Nombre de la serie donde apareció en más ocasiones el personaje.
`Dragon_ball`	Indica si el personaje apareció en la serie Dragon Ball.
`Dragon_ball_z`	Indica si el personaje apareció en la serie Dragon Ball Z.
`Dragon_ball_gt`	Indica si el personaje apareció en la serie Dragon Ball GT.
`poder_aumenta`	Indica la cantidad de veces que el personaje tuvo un aumento de poder.
`poder_minimo`	Indica el poder mínimo que tuvo el personaje.
`poder_maximo`	Indica el poder máximo que tuvo el personaje.
`poder_promedio`	Indica el poder promedio que tuvo el personaje.

1.2. Herramienta objetivo, funcionalidades y requisitos

Como se mencionó anteriormente, la herramienta debe suplir varias necesidades y tareas para el usuario objetivo. Con esa intención, en esta sección se detallan las funcionalidades y requerimientos objetivo a implementar de forma explícita, para así orientar de mejor forma el desarrollo de tu herramienta.

Como adelanto, en la **Figura 1** y **Figura 2** se presentan ejemplos de las figuras artísticas a realizar en la primera y segunda visualización respectivamente. Ten claro que **no se espera una reproducción idéntica de estos ejemplos**, este es solo un referente para dar mayor claridad sobre las visualizaciones a diseñar.

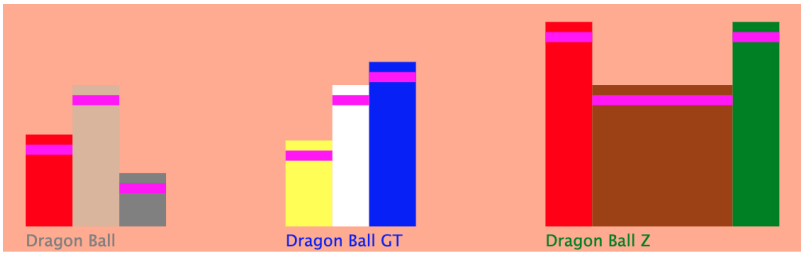

Figura 1: Ejemplo de la primera visualización: series. Cada conjunto de libros representa una serie. La altura, color y ancho de ellos codifica diferente información de cada serie.

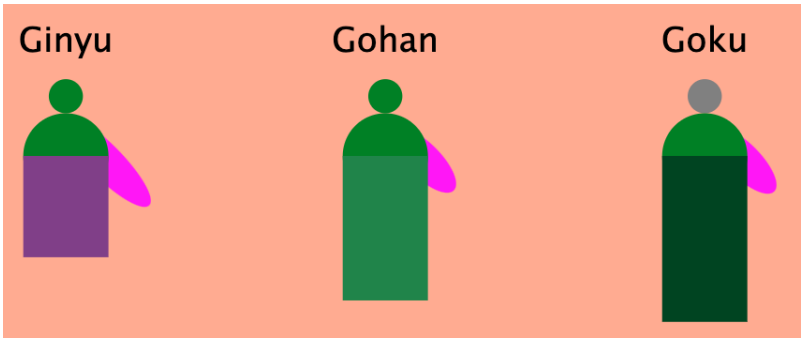

Figura 2: Ejemplo de la segunda visualización: personajes. Cada persona con la mano derecha hacia abajo representa un personaje. Los color utilizados y tamaño de cuerpo y brazo codifican diferentes propiedades del personaje.

A continuación se listan las funcionalidades y requisitos que se espera que cumpla la herramienta. Primero, aquellos generales de la herramienta:

- Construir un documento HTML para presentar la herramienta. Los mínimos elementos que debe incluir este documento son:
    1. Un título principal descriptivo de la herramienta artística.
    2. Una descripción de máximo 200 palabras que entregue información sobre las visualizaciones a observar.
    3. El creador de la visualización (tu nombre completo).
    4. Indicar el navegador (firefox, chrome, safari, etc) que fue utilizado al momento de confeccionar la tarea.
    5. Citar la fuente original de los datos utilizados (es decir **este *dataset***).
    6. Un botón o *link* que permita acceder a `vis.html`.
    7. Incluir las **referencias a todo código utilizado que no sea creación total del estudiante**. Con incluir el enlace o mencionar las clases y/o ayudantías en las cuales se utilizó el código es suficiente. En caso de no usar ningún código de internet, en esta sección se debe declarar que todo material es creación del estudiante.

       Si se detecta que el estudiantado no citó algún código externo utilizado en la tarea, se penalizará en el ítem de puntaje y además se aplicará un descuento adicional por no incluir referencia al código que no es de su autoría.
- El documento HTML debe incluir estilaje con *external CSS* para cambiar el color de fondo y dejar un margen a los costados del documento. El color y la cantidad de margen quedan a tu criterio.
- Utilizar el archivo `vis.html` y `main.js` para construir las dos visualizaciones solicitadas y **creadas a partir del `data join` de D3**.
- La implementación no debe presentar errores de funcionamiento que interrumpan su experiencia de uso, ni errores de código durante su funcionamiento.

Segundo, las funcionalidades y requisitos relacionados a implementar la primera visualización, Series:

- Cada serie será representado por un conjunto de 3 libros. Estos conjuntos no se pueden superponer y se componen de:
    1. **Libro izquierdo**: Su altura codifica la cantidad de personajes extras en la serie. Tiene un ancho constante y posee un color distinto si la serie está basada en manga o no. Queda a tu criterio cuál color elegir mientras sean 2 colores categóricos diferentes.
    2. **Libro del medio**: Está justo al lado del libro derecho. Su ancho es proporcional a la cantidad de aventuras que tiene la serie y su color es proporcional a la cantidad de capítulos: mientras más oscuro es el color, más capítulos tiene la serie. Tiene un altura constante. Queda a tu criterio cuál color elegir mientras se respete que más claro es menos capítulos y más oscuro es más capítulos.
    3. **Libro derecho**: Está justo al lado del libro del medio. Su altura codifica la cantidad de personajes recurrentes en la serie. Tiene un ancho constante y posee un color distinto según la serie que codifica. Queda a tu criterio cuál color elegir mientras sean 3 colores categóricos diferentes.

4. **Texto**: Está debajo de cada conjunto con el nombre de la serie. El color del texto es igual al color del libro derecho, es decir, de la serie que codifica.
5. **Tejuelo**: Todo libro posee una banda rectangular cuyo ancho es igual al libro, tocando ambos extremos del libro, y tiene una altura constante. Se ubica en la parte superior del libro, pero sin tocar el borde superior. Luego, el color del tejuelo será el mismo entre todos los libros, pero es distinto a cualquier color utilizado en los libros.

- Para todo color que dependa de algún valor **se deben utilizar correctamente escalas de D3**. Puede ser `scaleLinear`, `scaleOrdinal`, etc. Solo no se obligará utilizar escalas de D3 si es un color constante en todo el trabajo, por ejemplo, el color del tejuelo.
- Dentro del archivo base, la visualización incluye una breve explicación de cada aspecto visual. Dentro de esa explicación se incluyen 5 elementos `<span>` que deberás modificar. En particular, utilizando D3.js o *external CSS*, deberás definir su `background-color` para que sea igual a los colores categóricos utilizados en la visualización. Cada `<span>` posee un ID de elemento para poder identificarlo. Por ejemplo, `<span id=‘‘legendDBZ’’></span>` deberá tener un `background-color` igual al color elegido para representar a la serie Dragon Ball Z.
- Luego, la visualización incluye un contenedor con detalle de una serie. Dentro de este contenedor se incluyen 6 elementos `<span>` que deberás modificar. En particular, cuando el *mouse* esté sobre un grupo de libros, deberás utilizar únicamente D3.js para actualizar el contenido de cada `<span>` con las diferentes propiedades de la serie donde está el *mouse*. Cada elemento tiene un ID de elemento para poder identificarlo. Por ejemplo, si el *mouse* está sobre el conjunto de libros de Dragon Ball GT, el `<span id=‘‘detailName’’></span>` deberá tener como contenido, el *string* `"Dragon Ball GT"`.
- Es posible presionar, con el *mouse*, un conjunto de libros para actualizar la segunda visualización. Una vez presionado un conjunto, se debe identificar la serie presionada. Luego, se debe actualizar la segunda visualización para mostrar únicamente los personajes que aparecieron en dicha serie. Además, deberás dejar una marca visual, en esta visualización, que indique la serie seleccionada.
- Se incluyen 3 botones que simulan el seleccionar una de las 3 series. Una vez presionado uno de estos botones, se debe destacar el conjunto de libros correspondiente a la serie presionada y actualizar la segunda visualización.

Tercero, las funcionalidades y requisitos relacionados a la segunda visualización: los personajes.

- Cada personaje será representado por una persona y con uno de sus brazo hacía abajo. Estas personas no se pueden superponer y se componen de:

1. **Cabeza**: un círculo superior que representa su cabeza. El tamaño del círculo es constante. El color de la cabeza será igual a la serie donde apareció por primera vez el personaje. Los colores utilizados para este elemento deben ser los mismos que fueron definidos en la primera visualización.
2. **Cuerpo superior**: La parte superior del cuerpo representado por medio círculo, el cual está centrado verticalmente a la cabeza. Su tamaño es fijo y su color será igual a la serie donde aparece en más ocasiones el personaje. Los colores utilizados para este elemento deben ser los mismos que fueron definidos en la primera visualización.
3. **Brazo derecho**: Un brazo derecho representado por una elipse que está rotada en diagonal hacia abajo y conectado al cuerpo superior. El ancho del brazo es constante, mientras que el largo es proporcional al poder mínimo del personaje. Debes utilizar una escala logarítmica para

codificar este valor. Además, debes asegurar que el brazo solo se vea en el hemisferio derecho del cuerpo, es decir, ninguna parte de la elipse debe ser vista en el hemisferio izquierdo. Queda a tu criterio cómo lograr este desafió. Finalmente, el color del brazo es constante entre todos los personajes y queda a tu criterio elegir dicho color.

4. **Cuerpo inferior**: La parte inferior del cuerpo que será representada por un rectángulo, el cual está centrado verticalmente a la cabeza. El ancho es igual al tamaño del cuerpo superior, y el largo de este elemento codifica el poder promedio que tiene el personaje. La escala a utilizar debe ser logarítmica y el *range* de esta escala debe definirse de tal modo, que el valor mínimo del cuerpo sea igual que el radio utilizado en la parte superior del cuerpo.
5. **Color cuerpo inferior**: El color de la parte inferior del cuerpo representa la cantidad de aventuras que vivió el personaje. Deberás utilizar una escala divergente entre purpura y verde, para esto, se recomienda utilizar el siguiente código donde `aventuras_minimas` y `aventuras_maximas` representan los valores extremos de aventuras entre todos los personajes (d3.min y d3.max), mientras que `aventuras_mediana` representa el valor justo del medio de aventuras entre todos los personajes (d3.median).

```
d3.scaleDiverging(d3.interpolatePRGn)
  .domain([aventuras_minimas, aventuras_mediana, aventuras_maximas])
```

6. **Nombre**: encima de cada persona, se debe incluir el nombre del personaje que debe estar totalmente alineado verticalmente con el personaje. Se recomienda investigar sobre `text-anchor`.

- Se debe poder **filtrar** los personajes para ver solo aquellos que han logrado aumentar su poder en más de 10 ocasiones. Para esto, se incluyen dos botones en el HTML con toda la lógica programada para que llamen a la función encargada de crear la segunda visualización. Solo falta la parte de filtrar el *dataset*.
- Se deben desplegar entre 4 a 6 personajes por fila, tu decides cuantas poner por fila mientras respetes la cantidad[3].
- Se debe poder cambiar el **orden** de los personajes según: su nombre, la cantidad de veces que aumentó su poder o el orden por defecto (el orden original con el que vienen en el *dataset*), Este reorden de personajes debe darse mediante una animación gradual en el tiempo. Queda a tu criterio si el orden es ascendente o descendente. Toda la lógica del selector ya está programada, solo falta reordenar el *dataset* cuando corresponda y aplicar la animación.
- Solo mientras el *mouse* esté sobre uno de los personajes, se debe reducir la opacidad de todos los demás personajes a excepción del que tiene el *mouse* encima.
- Además de reducir la opacidad, se debe desplegar un *tooltip*, junto a la posición del *mouse*, que muestre la información toda la información del personaje. Debes mostrar el nombre de cada propiedad y su valor respectivamente. Una vez retirado el *mouse*, el *tooltip* se debe dejar de observar. Este *tooltip* no puede ser creado mediante el atributo `title` del SVG; debe ser un cuadro que aparezca de inmediato cuando se pasa el *mouse* e incluya el texto. Se recomienda fuertemente estudiar e inspirarse en **este ejemplo**. Deben citar esta página en caso de ocuparla.
- Utilizar correctamente el *Data Join* cuando es necesario actualizar y eliminar elementos del SVG. Es importante mencionar que eliminar y crear un nuevo SVG o eliminar todos los elementos del SVG fuera del *Data Join*, **no es una correcta aplicación del *Data Join***.

[3] Se recomienda fuertemente investigar sobre módulo y división entera en Javascript.

- Utilizar animaciones en el tiempo (transiciones de D3) cuando se crea, actualiza y filtra la visualización de los personajes. Recuerde el `data join` **de D3.js**: *enter*, *update* y *exit*. Las animaciones en específico a utilizar quedan a tu criterio.

Respecto a los filtros y ordenar los personajes, el orden que debe seguir tu código es:

1. Actualizar un H4 con la información que vamos a desplegar. Esta parte ya está lista en el código entregado.
2. Filtrar los personajes para quedarse solo con aquellos que aparecen en la serie seleccionada. Esta parte ya está lista en el código entregado.
3. Filtrar por la cantidad de aventuras en las que participaron los personajes según corresponda. **Esta parte la debes implementar.**
4. Ordenar los datos de (3) según corresponde. **Esta parte la debes implementar.**
5. Crear y actualizar la visualización con los datos de (4). **Esta parte la debes implementar.**

   Importante: Todas las escalas numéricas deben estar definidas según la información de los datos **después de haber filtrado**. Por ejemplo, si al principio una escala tenía un dominio de 10 a 100, pero post filtrar ahora el dominio es de 50 a 100, se tiene que actualizar la escala y todos los elementos visuales correspondientes.

1.3. Importante: Libertades y mínimos de implementación

Como fue mencionado previamente, no se espera un resultado idéntico visualmente al ejemplo provisto, mientras las funcionalidades implementadas sean las pedidas, se cumpla la pauta, y se respete el diseño de cada figura (libros y personitas).

Te invitamos a explorar una paleta de colores distinta para el fondo, texto y elementos de vistas. Además, puedes probar con distintos tamaños y fuentes, pero procura que sea legible y haga sentido. Además, debes respetar el enunciado y la pauta en todo momento.

Si hay algún aspecto a implementar que no te queda claro si permite libertad de realización, por favor consúltalo en el **foro del Syllabus**.

Por un lado, en cuanto a **requisitos totalmente obligatorios** a considerar en tu implementación, están las siguientes:

1. Tu programa solo puede hacer uso de funciones nativas de JavaScript o provistas por D3.js. No se permite utilizar otras librerías de JavaScript.
2. Puedes hacer uso tanto de la versión 6 o 7 de D3.js.
3. No utilizar *loops* (`for`, `while`, `forEach`, `each`) para construir la visualización. **Solo si necesitan calcular algún dato adicional, se permitirá utilizar *loops***. Se recomienda confirmar su uso en el foro antes de entregar.
4. Las visualizaciones deben ser desarrolladas con elementos propios de un SVG (`rect`, `circle`, `line`, `text`, `ellipse`, `g`).

El no seguir alguna de estas consideraciones mínimas producirá que tu evaluación no sea corregida y se califique con nota mínima.

Por otro lado, algunas **consideraciones mínimas** a tener en cuenta en tu implementación, están las siguientes:

1. Debes utilizar los datos entregados para construir la visualización.
2. Cada visualización debe estar contenida en un único SVG. Por lo tanto, `vis.html` solo puede contener 2 SVG.
3. Se espera que utilices estilamiento nativo mediante CSS escrito por ti para esta evaluación, es decir, no se permite importar *frameworks* o usar herramientas de estilamiento ya construidas.
4. Para todo color que dependa de algún valor **se deben utilizar correctamente escalas de D3**. Puede ser `scaleLinear`, `scaleLog`, `scaleOrdinal`, etc. Será penalizado el utilizar crear una función o diccionario de Javascript para administrar los colores en reemplazo de D3.
5. Aplicar **correctamente** *data join* de D3 para asociar elementos visuales a información del *datasets*. Esto incluye utilizarlo correctamente para actualizar y eliminar cualquier elemento de la visualización asociado a algún dato.
6. No puedes modificar los archivos que el enunciado indica explicitamente que no puedes modificar.

En este caso, el no seguir alguna de estas consideraciones repercutirá en un fuerte descuento, pero de todas formas se evaluará el resto de la tarea.

1.4. Archivos entregados

Para esta tarea, se te hará entrega de algunos archivos con varios elementos ya programados. Su trabajo será completar los archivos con lo solicitado en esta evaluación:

- `vis.html`: archivo HTML con los elementos necesarios para confeccionar las visualizaciones. **No debes modificar este archivo**.
- `style.css`: archivo CSS que aplica estilo a `vis.html`. Puedes modificar su contenido, pero no el nombre del archivo.
- `configuration.js`: archivo JS encargado de configurar la lógica de los botones y selectores.

  Se incluye una función llamada `preprocesarPersonajes` que recibe la serie seleccionada y si hay o no que filtrar los datos. Esta función llamará a `crearPersonajes` encargada de confeccionar la segunda visualización **No debes modificar este archivo**.
- `main.js`: archivo JS con 2 funciones que deberás completar para el correcto desarrollo de la evaluación.

  1. `crearSeries()`
     Esta función se llama una vez cuando se carga la página y es la encargada de crear la primera visualización. **Importante**: Cada vez que se haga *click* en un conjunto de libros (una serie), deberás llamar a `preprocesarPersonajes` con la información de la serie seleccionada e indicando que no hay que filtrar, es decir, `preprocesarPersonajes(serie_seleccionada, false)`. Esta función se encargará, posteriormente, se llamar a `crearPersonajes`.

2. `crearPersonajes(dataset, serie, filtrar_dataset, ordenar_dataset)`
   Esta función se llama cada vez que se crea o actualiza la segunda visualización. `dataset` es una lista de diccionario con la información de todos los personajes, `serie` es un *string* con la serie seleccionado a visualizar ("Dragon Ball", "Dragon Ball Z" o "Dragon Ball GT", y `filtrar_dataset` es un booleano indicando si hay que filtrar o no los datos según el radio y `ordenar_dataset` nos indica si hay que ordenar los datos por nombre alfabético, por si `poder_aumenta` es mayor a 10 o dejarlos tal como vienen en el *dataset* (orden por defecto).

2. Corrección y rúbrica de evaluación

Para la corrección de esta evaluación, se revisará la visualización confeccionada y el código de Javascript utilizado. Además, se usará una pauta como guía. La mayoría de los ítems serán calificado de forma ternaria: cumple totalmente el ítem, cumple parcialmente el ítem, o no cumple el ítem, aunque hay algunos que por su naturaliza binaria, serán calificados como cumple o no cumple. Además, esta pauta incluye tres ítems de descuento que serán calificados de forma binaria: hay o no hay descuento asociado según corresponda. Pueden encontrar el detalle de cada ítem a evaluar **en el siguiente *spreadsheet***, en la hoja llamada "Tarea 2".

Además de determinar el nivel de logro alcanzado, el equipo docente adjuntará retroalimentación escrita que complemente la corrección cuando sea necesario.

3. Entregables

Se espera que el entregable corresponda a: **una herramienta de visualización artística de información a partir de código JS y D3.js**. Este resultado debe entregarse como un ZIP como mínimo cinco archivos: un documento **HTML** para la primera página, el documento **HTML** entregado para las visualización, los dos archivos **JS** y el archivo de estilo **CSS**. No se aceptarán entregas en cualquier otro formato distinto al indicado anteriormente (archivos TXT, DOC, etc.). De no entregar o entregar un formato diferente al especificado, no se revisará la entrega y se colocará nota mínima. **No debes incluir el enunciado y el *dataset* en el ZIP.**

4. Dudas

Cualquier duda que tengas sobre esta evaluación, prefiere publicarla en el **Syllabus del curso** correspondiente a esta evaluación. También, siente la libertad de responder dudas de tus pares si crees que conoces la respuesta. En caso de tener dudas que impliquen mostrar tu solución o partes de ella, no utilice este medio de consulta. Para estos casos, envíe un correo al cuerpo docente o muestre su solución solo en reunión personal (remota o presencial) cuando se reúna con algún miembro del cuerpo docente.

5. Política de atraso

Existe la posibilidad de entregar esta evaluación con hasta **3 días de atraso** a partir de la fecha de entrega definida en el enunciado. En la eventualidad de entregar pasada la fecha de entrega, se aplicará una **reducción** a la nota máxima que podrás obtener en esta evaluación.

De haber atraso, **la nota máxima a obtener** se reduce en **0.5 puntos (5 décimas)** por cada día de atraso. Cada día de atraso se determina como el techo de días de atraso. Por lo tanto, en caso de entregas atrasadas, la nota final de la tarea se calcula mediante la siguiente fórmula:

$$\text{mín}(7 - (0.5 \times \texttt{días\_atraso}),\ \texttt{nota\_obtenida}) + \texttt{bonus}$$

Por otro lado, entregas con más de 3 días (72 hrs) de atraso no serán recibidas y serán evaluadas con la **calificación mínima (1.0)**.

6. Flexibilidad de entrega

En la eventualidad de que tengas problemas personales durante el plazo de esta evaluación, a tal punto que impida su realización de forma importante y **requieras más de los 3 días de atraso permitidos**, siéntete libre de contactar a alguien del equipo docente para buscar apoyo y opciones de flexibilidad. Para casos médicos, recuerda que antes de todo debes justificar con tu unidad académica para respetar el conducto regular de la Universidad.

Es completamente posible otorgar una extensión, de plazo individual o re-evaluar la política de atraso para cada caso en particular. Se espera que escribas explicando tu situación, al punto que sientas comodidad de hacerlo, para así entender y considerar tu caso. También se aprecia si se propone una cantidad de extensión a necesitar dentro de la solicitud.

Para esta flexibilidad posterior a los 3 días de atraso, escribir al docente del curso (**hfvaldivieso@uc.cl**).

7. *Bonus* de la evaluación

Esta evaluación presenta un *bonus* para bonificar la nota final de esta evaluación. Para poder optar al beneficio que entrega cada *bonus*, este debe estar implementado **en su totalidad**, es decir, **no se dará puntaje intermedio**. Adicionalmente, **todos los *bonus*** requieren que la nota en tu tarea (sin ningún tipo de *bonus*) debe ser **igual o superior a 4.0**.

7.1. Máximo poder (1 décimas)

Dentro de las propiedades de cada personaje estaba el `poder_maximo` que no se está ocupando en ninguna parte. Se otorgará 1 décima si se incluye esta información de alguna forma **visual** en la segunda visualización. Para esto, debes agregar un nuevo elemento visual (*line*, *cicle*, *rect*, etc) cuyo tamaño y/o color codifique el poder máximo de cada personaje. Este elemento visual no debe interferir con los elementos obligatorios de la tarea, por ejemplo, no puedes incluir un elemento que oculte la cabeza del personaje.

Adicionalmente, en caso de realizar esta bonificación, **debes indicar en el archivo index.html** que estás optando al *bonus* de máximo poder y con qué propiedad visual se está codificando esta información.

7.2. Documento *Responsive* (2 décimas)

Se otorgará una bonificación de 2 décimas si `vis.html` es *responsive*, es decir, su contenido al tamaño de la ventana. Este requerimiento incluye la visualización. Se recomienda investigar sobre **viewBox** en SVG. En **Google** encontrarán mucha información sobre este concepto. **Importante**: debes hacer esto solo utilizando comandos de CSS y JS, y por lo tanto, no puedes modificar `vis.html` bajo ningún motivo.

Adicionalmente, en caso de realizar esta bonificación, **debes indicar en el archivo index.html** que estás optando al *bonus* de documento responsive para evaluar si la visualización de ajusta al tamaño del navegador.