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

Hito 2

Implementación de herramientas de visualización

Fecha de inicio: **28 de octubre** a las **11:30** hrs

Fecha de entrega: **11 de noviembre** a las **20:00** hrs

Tiempo estimado de trabajo[1]: **10 horas**

Evaluación en el contexto del curso

Esta evaluación es de naturaleza **sumativa**, y pretende rescatar evidencias del desarrollo de ciertos resultados de aprendizaje. Por eso, tras su realización y entrega, recibirás retroalimentación sobre tu desempeño y una nota que lo refleja.

A diferencia de la Entrega que le precedió a esta evaluación, esta debes realizarla de manera **individual**, y no grupal.

Específicamente, esta evaluación se relaciona con el resultado de aprendizaje indicado en el programa del curso: **Construir herramientas de visualización interactivas mediante programación para resolver una necesidad de comunicación de información puntual**.

El material del curso relevante para esta evaluación es el siguiente:

1. Introducción a HTML, CSS y SVG
2. Introducción a JavaScript y D3.js
3. Selecciones y *join* de datos en D3
4. Utilidades de D3 I y II
5. *Layouts* tabulares en D3
6. *Zooming* y *panning* en D3
7. Visualización de datos espaciales

[1]El tiempo estimado de trabajo es una aproximación realizada por el equipo docente del tiempo promedio que debería tomarle a una persona desarrollar esta evaluación. Esta toma el supuesto de que esta persona ha interactuado con el material y participado de las sesiones sincrónicas del curso con anticipación a la fecha de inicio de la evaluación. Al ser un promedio, no es una garantía que siempre se complete la evaluación antes de este tiempo. A su vez, esto no asume ni sugiere que son horas seguidas de trabajo, si no que es la suma de tiempo en dedicar a sus partes.

1. Implementación de herramientas de visualización

Esta evaluación busca que implementes, mediante programación y tecnologías web, una herramienta de visualización 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 un documento HTML que produzca una herramienta de visualización de información interactiva y basada en SVG para una 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.

Dicha situación, contexto de ella, especificaciones de la herramienta y ejemplo demostrativo se detallan en las siguientes secciones.

1.1. Explorador de precipitaciones

La necesidad a satisfacer es la de una investigadora de los efectos del cambio climático en Chile, llamada Diana, que busca comprender en distintas dimensiones cómo ha sido la evolución de la cantidad de precipitaciones caídas en distintos lugares del país.

La herramienta a crear sería un portal de información para Diana, por lo que su función general será presentar la evolución de precipitaciones en distintas estaciones de medición a lo largo del Chile. Diana tiene conocimiento parcial sobre qué estaciones de medición existen y dónde se encuentran geográficamente, por lo que la herramienta debe ser permitir localizar estaciones por su nombre, sondear según su ubicación geográfica, y explorar estaciones en general.

En cuanto a las precipitaciones, Diana necesita conocer la evolución de esta cantidad a lo largo de los últimos años, para cualquiera de las estaciones disponibles, pero también necesita comparar la evolución de cantidad de precipitaciones entre distintas estaciones. Por esto, sería apropiado poder seleccionar múltiples estaciones para comparar sus respectivas evoluciones.

Los *datasets* a utilizar en este ejercicio son `estaciones.json` y `regiones.json`. El primero[2] contiene la información de distintas estaciones de medición de precipitaciones a lo largo de Chile. El segundo[3] es un archivo **TopoJSON** que contiene la geometría geográfica de Chile a nivel de regiones.

Los archivos entregados son tales que es posible cargar ambos en un programa JavaScript y usarlos directamente para implementar la herramienta solicitada. Si deseas pre-procesar aún más dichos archivos de alguna forma por conveniencia para tu programa, puedes hacerlo, pero recuerda incluir en tu entrega el resultado de tu procesamiento en tu entrega y el programa necesario para generarlo.

[2] *Dataset* original extraído de **El Centro de Ciencia del Clima y la Resiliencia**, y posteriormente procesado y filtrado por equipo docente.

[3] *Dataset* generado utilizando un **repositorio público** que permite generar archivos georreferenciados de Chile en distintos niveles.

Cada elemento en `estaciones.json` contiene las siguientes propiedades:

Propiedad	Descripción
`codigo_estacion`	Código identificador único de la estación en conjunto de datos.
`nombre`	Nombre con el cual se le conoce a la estación, también es único de la estación en el conjunto de datos.
`latitud`	Latitud de coordenada geográfica donde se encuentra la estación en el mundo.
`longitud`	Longitud de coordenada geográfica donde se encuentra la estación en el mundo.
`precipitaciones`	Arreglo de objetos que describen las distintas mediciones de precipitaciones en distintas fechas. Cada objeto tiene dos propiedades que describen la medición: `"fecha"` que es el mes en el que se hace la medición, y está en formato `yyyy-mm`; y `"valor"` que es la cantidad de precipitación en milímetros observadas.

Además, encontrarás instrucciones en **Anexo - Procesamiento de archivo TopoJSON** sobre cómo procesar el archivo TopoJSON entregado como un archivo GeoJSON.

1.2. Herramienta objetivo, funcionalidades y requisitos

Como se mencionó anteriormente, la herramienta debe suplir varias necesidades y tareas para su usuario objetivo. Con esa intención, se detallan en esta sección 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** puedes ver una captura de pantalla de un ejemplo de un **posible** resultado final de la herramienta objetivo. Además, en **este enlace podrás ver una demostración más detallada del ejemplo** en la **Figura 1**, donde se ven en acción las funcionalidades que se listan.

Ten claro que **no se espera una reproducción idéntica de este ejemplo**, este es solo un referente para mayor claridad sobre las funcionalidades a pedir.

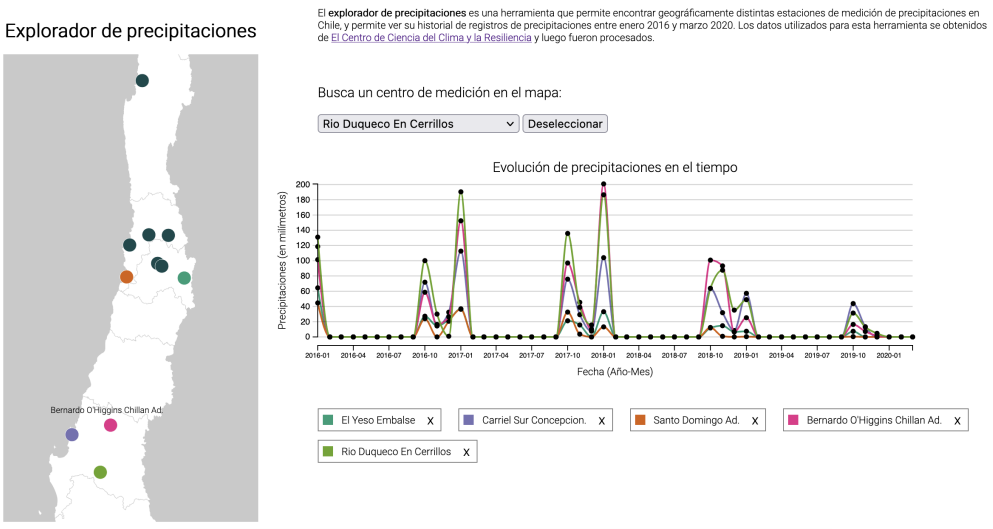


Figura 1: Captura de pantalla de ejemplo de herramienta.

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

- Contener información que contextualice la herramienta: tener un título principal, algún tipo de descripción que entregue información del fondo de la herramienta, y mencionar en algún lado la fuente original de los datos utilizados (es decir **El Centro de Ciencia del Clima y la Resiliencia**).
- Cargar los datos provistos (estaciones y geográficos) en la herramienta desde uno o varios archivos externos al programa principal escrito en JavaScript. Aquellos archivos cargados deben ser locales, es decir, archivos que viven en el mismo computador que ejecutan el programa.
- La implementación no presenta errores de funcionamiento que interrumpan su experiencia de uso, ni errores de código durante su funcionamiento.

Segundo, las funcionalidades y requisitos relacionados a implementar un mapa en la herramienta.

- Contener un mapa geográfico de Chile que muestre separaciones a nivel regional.
- Agregar marcas al mapa que representen las estaciones de medición en sus ubicaciones geográficas correspondientes.
- Al pasar el cursor sobre las marcas de estaciones en el mapa, se gatilla la aparición de una anotación con el nombre de la estación justo arriba de la marca en el mapa. Al sacar el cursor del área de la marca, la anotación desaparece.
- Al interactuar con el mapa, es posible gatillar navegación dentro de este mediante zoom y traslación manual. Esto permite cambiar la escala y ubicación de la vista actual del mapa. Además, los elementos dentro del mapa se ajustan a estos cambios para que mantener su tamaño aparente. Los tipos de interacciones que gatillan esta interacción sobre el mapa pueden ser cualquiera.

Tercero, las funcionalidades y requisitos relacionados a dos posibilidades de selección en la herramienta.

- Contener un ***input* de selección** que contenga todos los nombres de estaciones provistos en el archivo `estaciones.json`, y es posible cambiar el valor seleccionado.
- Al cambiar el valor de estación seleccionada en el *input* de selección, se gatilla un *zoom* automático animado sobre la estación en el mapa cuyo nombre fue seleccionado en el *input*.
- De forma separada al *input* de selección, es posible seleccionar hasta cinco estaciones de forma simultanea, y remover estaciones de dicha selección. El grupo seleccionado debe ser visible en la herramienta en todo momento. Es posible implementar esto de la forma que se estime conveniente, como mediante un botón específico de selección, interactuando con estaciones en mapa, o ambas incluso.

Cuarto, las funcionalidades y requisitos relacionados a implementar un gráfico de línea que cambia en el tiempo.

- Contener un gráfico de línea que se actualiza con el grupo de estaciones seleccionadas. Cada línea en el gráfico corresponde a una estación seleccionada.

- El gráfico de línea codifica la evolución de la cantidad de precipitaciones en el tiempo para las estaciones que muestra. Las líneas representan la conexión de puntos cuyas posiciones vertical y horizontal codifican los datos de precipitación: la posición vertical codifica la cantidad de precipitaciones, y la horizontal un punto en el tiempo de medición. El rango de tiempo va entre enero 2016 hasta marzo 2020, y el rango de precipitaciones va entre 0 y el máximo presente en las mediciones de estaciones mostradas.
- La implementación de gráfico de línea utiliza *join* de D3.js para el agregado, actualización y eliminación de líneas.
- Contener puntos para valores individuales de mediciones de precipitación en gráfico de línea, implementado mediante *join* de D3.js.
- Contener ejes visibles y con etiquetas de valores en gráfico de línea.
- Contener títulos en ejes de gráfico de línea y título general para el mismo gráfico.
- Utilizar un *colormap* categórico para identificar líneas en gráfico de línea.
- Utilizar de forma consistente el mismo *colormap* categórico en mapa para identificar estaciones entre vistas.
- Contener una leyenda para gráfico de línea sobre las estaciones mostradas.

Finalmente, otras funcionalidades y requisitos adicionales para la herramienta.

- Al pasar el cursor sobre puntos de mediciones en el gráfico de línea, se gatilla la aparición de una anotación con el valor de precipitación medida justo arriba al punto. Al sacar el cursor del área de un punto, la anotación desaparece.
- Utilizar animaciones en el tiempo para la actualización de ejes y líneas en gráfico de líneas.
- Es posible remover estaciones del grupo de estaciones seleccionadas desde la leyenda del gráfico de línea.

1.3. 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.

Puedes implementar los funcionamientos de la misma forma que el ejemplo, pero siéntete libre de explorar otras formas de organizar visualmente las distintas partes de la herramienta y de diseñar aquellas acciones que se dejan abiertas (como la selección de múltiples estaciones de medición).

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

También se te invita a explorar una paleta de colores distinta para el fondo, texto y elementos de vistas. Además, prueba con distintos tamaños y fuentes, pero procura que sea legible y haga sentido.

En cuanto a consideraciones mínimas a considerar en tu implementación, están las siguientes:

- Tu programa solo puede hacer uso de funciones nativas de JavaScript, provistas por D3.js, o por el cliente para trabajar con datos TopoJSON. No se permite utilizar otras librerías de JavaScript.

- Puedes hacer uso tanto de la versión 6 o 7 de D3.js, pero no una anterior a ambas.
- 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.

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

2. Corrección y rúbrica de evaluación

Para la corrección de esta evaluación, se revisará el resultado de la herramienta entregada y se usará una rúbrica como guía. Esta detalla niveles de cumplimiento para los distintos criterios establecidos y solicitados y puedes encontrarla **aquí**. Además de determinar el nivel de desempeño alcanzado, el equipo docente adjuntará retroalimentación escrita que complemente la información de la rúbrica.

Cada criterio tiene cierta ponderación de puntaje. El multiplicar el nivel obtenido en el criterio por su ponderador correspondiente es equivalente al puntaje obtenido en dicho criterio. La nota en la evaluación se obtiene de comparar la suma de puntajes obtenidos en todos los criterios y el máximo posible al lograr el mayor desempeño en todos los criterios. Se usará una escala lineal donde el puntaje mínimo obtiene nota 1, y el máximo 7.

3. Entregables

Se espera como entregables un resultado principal: **una herramienta de visualización de información**. Este resultado debe entregarse como un documento **HTML**, extendidos mediante archivos **CSS** y **JavaScript**. No se aceptarán entregas en cualquier otro formato distinto al indicado anteriormente (archivos PDF, TXT, DOC, etc...). De no entregar o entregar un formato diferente al especificado, no se revisará la entrega y se colocará nota mínima.

La entrega se realizará mediante un archivo comprimido de extensión **ZIP** que contenga la herramienta, mediante la plataforma Canvas en la **evaluación** correspondiente. Cualquier otro archivo (imágenes, hojas de estilo, *scripts*, etc.) que sea necesario para la visualización correcta de tu entrega debe también ir incluido en el archivo comprimido. Si no modificaste los archivos de datos entregados y tu herramienta funciona con los mismos archivos, no es necesario que se incluyan en la entrega. Si procesaste más aún los archivos de datos, debes entonces incluirlos en tu entrega, además del programa que realiza el procesamiento.

4. Dudas

Cualquier duda que tengas sobre esta evaluación, prefiere publicarla en el **foro** correspondiente a esta evaluación. También siente la libertad de responder dudas de tus pares si crees que conoces la respuesta.

Dudas específicas y personales de implementación, puedes realizarlas en algún horario de atención personal en el **servidor** de Discord del curso, o durante las sesiones de trabajo en horario de clase.

5. 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, siéntete libre de contactar a alguien del equipo docente para buscar apoyo y opciones de flexibilidad.

Es completamente posible otorgar una extensión de plazo individual. Se espera 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.

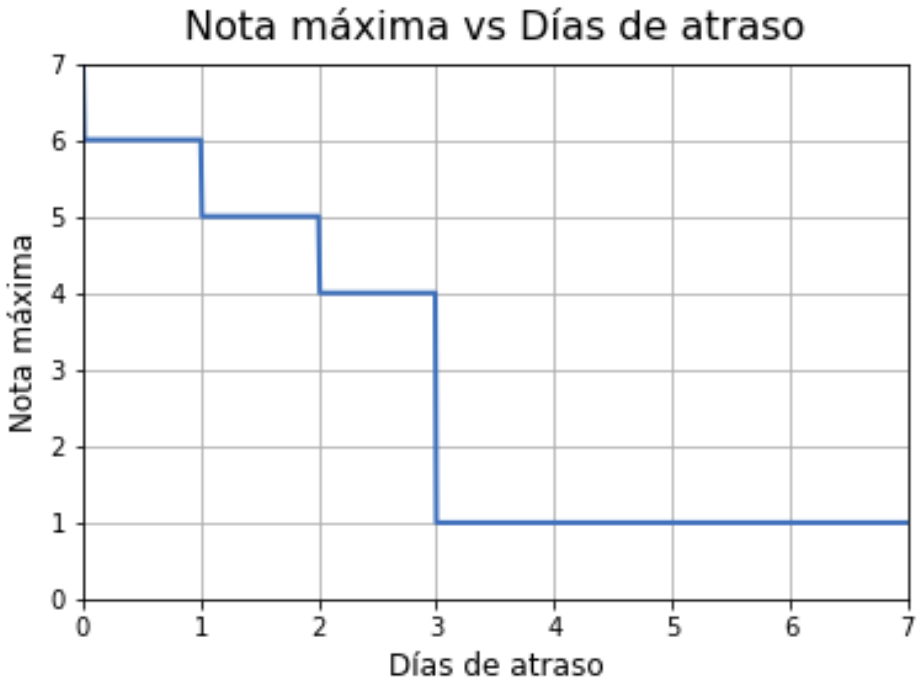
De preferencia escribe un correo a la ayudante de bienestar (**diflores@uc.cl**) o al docente del curso (**fa-florenzano@ing.puc.cl**).

6. 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 o posterior a una extensión de plazo otorgada. En la eventualidad de entregar pasada la fecha de entrega, se aplicará una **reducción** a la nota máxima que podrás obtener en tu hito.

De haber atraso, la nota máxima a obtener se reduce en **1 punto (10 décimas)** por cada día de atraso, siendo la nota final del hito calculada mediante la siguiente fórmula:

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



Mientras que cualquier examen que sea entregado con más de 3 días (72 hrs) de atraso será evaluado con la **calificación mínima (1.0)**.

7. Anexo - Procesamiento de archivo TopoJSON

El formato de archivos TopoJSON es una extensión del formato GeoJSON, pero que en vez de guardar información geométrica de forma literal (mediante coordenadas y dimensiones absolutas) lo hace describiendo topologías o las formas de los objetos a describir.

No es necesario entender a cabalidad los detalles del formato para lo que incume esta evaluación, pero la gracia más importante de este formato es que elimina redundancias que suelen ocurrir en archivos

GeoJSON. Luego, al convertir un archivo desde un formato GeoJSON a TopoJSON usualmente hay una reducción de tamaño importante, lo que hace más barato cargar un archivo TopoJSON.

Es posible transformar de vuelta datos en formato TopoJSON de vuelta a GeoJSON en un programa escrito en JavaScript. Para eso, puedes incluir la librería `topojson-client` que tiene funciones de utilidad de manipulación en tu documento HTML. La **Figura 2** muestra como hacerlo mediante la etiqueta `script` de HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Evolucion de precipitaciones en Chile</title>
    <meta charset="UTF-8">
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <script src="https://unpkg.com/topojson-client@3"></script>
  </head>
  <body>
    ...
  </body>
</html>
```

Figura 2: Importar librería de cliente para trabajar con TopoJSON.

Luego, en un tu programa puedes llamar a la función `topojson.feature(topologia, objeto)` para convertir un objeto en una topología TopoJSON a un objeto equivalente en GeoJSON, y luego ocuparlo tal como se ha visto en ejemplos del curso.

El objeto `objeto` debe estar definido dentro del objeto `topologia`, y usualmente está contenido en una propiedad dentro de la propiedad `'objects'` del un objeto topología. En el caso del archivo `regiones.json` entregado, `objeto` correspondería a `topologia.objects.regiones`, si `topologia` es el objeto cargado mediante `d3.json`.

Por ejemplo, el código en la **Figura 3** que usaría datos GeoJSON para ajustar una proyección y definir una función de caminos geográficos, se convertiría al código en la **Figura 4** que haría lo mismo pero cargando datos TopoJSON.

```
d3.json("archivo_geojson.json").then((datos) => {
    const proyeccion = d3.geoMercator().fitSize([width, height], datos);
    const caminosGeo = d3.geoPath().projection(proyeccion);
    ...
}
```

Figura 3: Usar datos GeoJSON para ajustar una proyección y definir una función de caminos geográficos en D3.js.

```
d3.json("archivo_topojson.json").then((datosTopo) => {
    const datos = topojson.feature(datosTopo, datosTopo.objects.regiones);
    const proyeccion = d3.geoMercator().fitSize([width, height], datos);
    const caminosGeo = d3.geoPath().projection(proyeccion);
    ...
}
```

Figura 4: Usar datos TopoJSON para ajustar una proyección y definir una función de caminos geográficos en D3.js.