UNIVERSITAT DE BARCELONA

Trabajo final de grado

GRADO DE MATEMÁTICAS

Facultad de Matemáticas e Informática
Universidad de Barcelona

LIBRERIA DE VISUALIZACIONES ACCESIBLES

Autor: Miguel Victor Huayllas Choque

Directora:	Dra. Mireia Isabel Ribera Turro
Realizado en:	Departamento de Matemáticas e Informática

Barcelona, 16 de enero 2024

Resumen

A medida que la visualización de datos se vuelve cada vez más ubicua en diversos campos, desde la ciencia de datos hasta la comunicación de información en línea, es fundamental garantizar que estos gráficos sean accesibles para todas las personas, incluidas aquellas con discapacidades visuales o motoras. Mejorar la accesibilidad de los gráficos no solo cumple con principios éticos de inclusión, sino que también amplía el alcance y la utilidad de la información presentada, beneficiando a una audiencia más amplia y diversa. Este proyecto representa un avance significativo hacia la creación de un entorno digital más inclusivo y equitativo para todos.

El proyecto consistió en el desarrollo de una librería de visualización de datos llamada Altair-Easeviz, con el objetivo principal de mejorar la accesibilidad de los gráficos generados. Para lograrlo, se integraron heurísticas de accesibilidad y se implementaron funcionalidades específicas para personas con discapacidades visuales o motoras.

La librería complementa a Vega-Altair , utilizando Python como lenguaje principal de programación. Se emplearon tecnologías familiares y se siguió una metodología ágil para el desarrollo, lo que optimizó el tiempo y los recursos disponibles.

Además, se creó una página web de demostración que detalla cada tipo de gráfico disponible y proporciona ejemplos interactivos para los usuarios. La página web se desplegó de manera automatizada utilizando la plataforma Render, lo que simplificó el proceso de despliegue y garantizó su accesibilidad en diferentes plataformas.

Finalmente, la librería se publicó en PyPI y se mantuvo un repositorio en GitHub para que cualquier persona pudiera contribuir y expandir la biblioteca. Este proyecto representó un paso significativo hacia la creación de gráficos accesibles y comprensibles para todos los usuarios.

Resum

A mesura que la visualització de dades esdevé cada cop més ubicua en diversos camps, des de la ciència de dades fins a la comunicació d'informació en línia, és fonamental garantir que aquests gràfics siguin accessibles per a totes les persones, incloses aquelles amb discapacitats visuals o motrius. Millorar l'accessibilitat dels gràfics no només compleix amb principis ètics d'inclusió, sinó que també amplia l'abast i la utilitat de la informació presentada, beneficiant una audiència més àmplia i diversa. Aquest projecte representa un avanç significatiu cap a la creació d'un entorn digital més inclusiu i equitatiu per a tothom.

El projecte va consistir en el desenvolupament d'una llibreria de visualització de dades anomenada Altair-Easeviz, amb l'objectiu principal de millorar l'accessibilitat dels gràfics generats. Per aconseguir-ho, es van integrar heurístiques d'accessibilitat i es van implementar funcionalitats específiques per a persones amb discapacitats visuals o motrius.

La llibreria complementa a Vega-Altair, utilitzant Python com a llenguatge principal de programació. Es van emprar tecnologies familiars i es va seguir una metodologia àgil per al desenvolupament, la qual cosa va optimitzar el temps i els recursos disponibles.

A més, es va crear una pàgina web de demostració que detallava cada tipus de gràfic disponible i proporcionava exemples interactius pels usuaris. La pàgina web es va desplegar de manera automatitzada utilitzant la plataforma Render, la qual cosa va simplificar el procés de desplegament i garantir-ne l'accessibilitat en diferents plataformes.

Finalment, la llibreria es va publicar a PyPI, i es va mantenir un repositori a GitHub perquè qualsevol persona pogués contribuir i ampliar la biblioteca. Aquest projecte va representar un pas significatiu cap a la creació de gràfics accessibles i comprensibles per a tots els usuaris.

Abstract

As data visualization becomes increasingly ubiquitous across various fields, from data science to online information communication, it is crucial to ensure that these graphs are accessible to all individuals, including those with visual or motor impairments. Improving the accessibility of graphs not only aligns with ethical principles of inclusion but also expands the reach and utility of the presented information, benefiting a broader and more diverse audience. This project represents a significant advancement towards creating a more inclusive and equitable digital environment for everyone.

The project involved the development of a data visualization library called Altair-Easeviz, with the primary goal of enhancing the accessibility of generated graphs. To achieve this, accessibility heuristics were integrated, and specific functionalities were implemented for individuals with visual or motor disabilities.

The library complements Vega-Altair, using Python as the primary programming language. Familiar technologies were employed, and an agile methodology was followed for development, optimizing the available time and resources.

Additionally, a demonstration website was created detailing each available graph type and providing interactive examples for users. The website was deployed automatically using the Render platform, simplifying the deployment process and ensuring accessibility across different platforms.

Finally, the library was published on PyPI, and a repository was maintained on GitHub for anyone to contribute to and expand the library. This project represented a significant step towards creating accessible and understandable graphs for all users.

Agradecimientos

Quisiera expresar mi sincero agradecimiento a mi tutora la Dra. Mireia Isabel Ribera Turro, cuya guía y apoyo fueron fundamentales durante todo el proceso de este Trabajo de Fin de Grado. Siempre estuvo disponible para aclarar mis ideas, ofrecer sugerencias valiosas y orientarme en cada etapa del proyecto.

Además, deseo expresar mi agradecimiento a Rubén Alcaraz, cuyo trabajo e investigaciones proporcionaron una base sólida y contribuyeron significativamente a dar forma a la biblioteca desarrollada en este proyecto, a pesar de no conocerlo personalmente.

A ambos, gracias por su colaboración y dedicación, sin la cual este trabajo no habría sido posible.

Índice

1. Introducción

En la actual era digital, la generación masiva y el intercambio constante de información se han convertido en pilares fundamentales de la sociedad. Desde datos estadísticos de vital importancia hasta simples conjuntos de recibos, la información se genera y comparte en cantidades exorbitantes, con una diversidad de fuentes y finalidades. En este contexto, la capacidad de interpretar y analizar datos se vuelve crucial para comprender el mundo que nos rodea y tomar decisiones informadas.

1.1. Gráficos de visualización de datos

Ante la vasta cantidad de datos y las múltiples formas de interpretarlos, los gráficos de visualización de datos se erigen como una herramienta fundamental para la comprensión y el análisis. Al representar datos de manera visual, los gráficos permiten identificar patrones, tendencias y relaciones de manera más intuitiva y efectiva que la mera observación de números. Esta capacidad de traducir datos complejos en representaciones visuales comprensibles los convierte en una herramienta invaluable en campos tan diversos como la ciencia, la economía, la educación y más.

Sin embargo, la eficacia de los gráficos de visualización depende en gran medida de su accesibilidad. A menudo, estos gráficos no están diseñados teniendo en cuenta las necesidades de personas con discapacidades visuales o motoras, lo que puede limitar su capacidad para acceder y comprender la información presentada.

1.2. Gráficos accesibles

La importancia de la accesibilidad en la visualización de datos se refleja en el creciente número de regulaciones y normativas que exigen la inclusión de características accesibles en los gráficos. Organizaciones gubernamentales y entidades internacionales, como el parlamento europeo con la directiva (UE) 2019/882[1] y la World Wide Web Consortium (W3C)[2], han establecido directrices y estándares específicos para garantizar la accesibilidad en todos los ámbitos de la comunicación digital, incluida la visualización de datos. Además, con la creciente conciencia sobre la importancia de la inclusión y la equidad, la demanda de gráficos accesibles continúa aumentando día a día, impulsando a los desarrolladores y diseñadores a adoptar prácticas que promuevan la accesibilidad y la igualdad de acceso a la información.

La accesibilidad en la visualización de datos es esencial para garantizar que todos los individuos, independientemente de sus capacidades físicas o cognitivas, puedan participar plenamente en el proceso de interpretación y análisis de datos. Los gráficos accesibles deben ser diseñados de manera que sean comprensibles y utilizables por todos los usuarios, incluyendo aquellos que dependen de tecnologías de asistencia o dispositivos adaptativos.

Determinar qué elementos hacen que un gráfico de visualización de datos sea accesible puede resultar un desafío, dado que existen diversas investigaciones, pautas al

respecto y otros proyectos similares. Entre estas, para este proyecto optamos por implementar o basarnos en las siguientes:

- La investigación realizada por Ruben Alcaraz **Accesibilidad para personas con baja visión de los gráficos estadísticos en la prensa digital: una propuesta metodológica basada en indicadores heurísticos**[5], que propone 18 heurísticas para garantizar la accesibilidad de los gráficos para personas con baja visión. Estas heurísticas proporcionan un marco de referencia útil para evaluar y mejorar la accesibilidad de los gráficos, y han servido como base para el desarrollo de estrategias y herramientas destinadas a hacer que los gráficos sean más accesibles para un público diverso.
- La investigación titulada **Improving Colour Patterns to Assist People with Colour Vision Deficiency** [6]. Este estudio ofrece diversas técnicas para implementar patrones de relleno en los colores, con el objetivo de ayudar a las personas que padecen daltonismo a percibir y distinguir los distintos elementos visuales con mayor facilidad. La aplicación de estas técnicas puede contribuir significativamente a mejorar la experiencia de usuario y la comprensión de los gráficos para este grupo de personas.
- Las pautas establecidas en las **Web Content Accessibility Guidelines (WCAG) 2.2**[2]. Estas directrices proporcionan una serie de recomendaciones y estándares para garantizar que el contenido en línea sea accesible para una amplia gama de usuarios, incluidas aquellas personas con discapacidades visuales o motoras. La incorporación de estas pautas en el diseño y desarrollo de los gráficos nos permite asegurar que cumplan con los estándares de accesibilidad y sean utilizables por el mayor número posible de usuarios, independientemente de sus capacidades o limitaciones.
- Entre los proyectos similares en lo referente a la creación de gráficos accesibles, destacan HighCharts[3] y Plotty[4]. HighCharts ofrece una amplia gama de opciones para crear gráficos accesibles, que incluyen características como incluir texto alternativo a cada elemento del gráfico para los lectores de pantalla, seguir la guía de la WCGA2.1 para crear gráficos y usar la sonificación de los datos en el gráfico .Por otro lado, Plotty proporciona opciones más limitadas para la creación de gráficos accesibles, aunque aún así los gráficos generados cuentan con opciones en el gráfico.
- Chartability[12] es otra agrupación de heurísticas que aseguran que las visualizaciones de datos, sistemas e interfaces sean accesibles. Estas heurísticas están organizadas en principios con criterios de prueba, por lo que a lo largo del desarrollo se cuestiono si la librería los cumplía.

1.3. Motivación

El constante avance en el desarrollo web ha conducido a la creación de contenidos cada vez más atractivos e interactivos. Sin embargo, se percibe una necesidad insatisfecha en cuanto a la accesibilidad de estos contenidos para una audiencia más

amplia. La implementación de soluciones que mejoren la accesibilidad representa un desafío significativo, ya que implica comprometer la interfaz y la experiencia del usuario con el contenido web. No obstante, abordar este desafío adicional puede potenciar el impacto y la relevancia de los proyectos futuros.

En este contexto, esta librería aspira a servir como una herramienta que facilite la creación de contenidos web más accesibles. Reconocemos la importancia de garantizar que todos los usuarios, independientemente de sus capacidades o limitaciones, puedan disfrutar plenamente del contenido en línea. Al abordar la accesibilidad desde una perspectiva proactiva, esperamos fomentar la inclusión y la equidad en el ámbito digital.

Desde un punto de vista más personal, este proyecto representa un desafío estimulante para mí, ya que implica adentrarme en un área del desarrollo web que no había explorado previamente. Además, alinearse con mis intereses en el desarrollo web, esta iniciativa ofrece una oportunidad valiosa para expandir mis habilidades y probar los conocimientos adquiridos en mi tiempo en la carrera de informática.

2. Objetivos

2.1. Objetivo general

El objetivo fundamental de este proyecto es concebir y desarrollar una biblioteca que se integre con la reconocida librería Vega-Altair[15], con el fin de proporcionar recursos y herramientas destinadas a mejorar la accesibilidad de los gráficos generados con Vega-Altair tanto para el autor como para el usuario final.

Esta biblioteca se concibe como un complemento para Vega-Altair, una destacada librería de Python diseñada para la creación de gráficos de visualización de datos. A pesar de la versatilidad y potencia que ofrece Vega-Altair, se ha identificado una carencia en ciertas funcionalidades específicas relacionadas con la accesibilidad, lo cual ha motivado el desarrollo de esta nueva biblioteca.

2.2. Objetivos específicos

- Implementar temas de accesibilidad: A través de entry points[8], desarrollar cuatro temas que modifiquen la apariencia de los gráficos, mejorando su accesibilidad para diferentes usuarios, tales como aquellos con discapacidades visuales.Además, se ofrecerá la opción de modificar estos temas según las preferencias individuales de cada usuario, permitiendo una experiencia de visualización personalizada y accesible para todos.

- Generar descripciones textuales alternativas: Desarrollar una función que genere descripciones textuales alternativas o detalladas de los gráficos para facilitar su comprensión por parte de usuarios con discapacidades visuales o que dependan de lectores de pantalla.

- Crear interfaz HTML para personalización: Diseñar e implementar una interfaz HTML que permita a los usuarios personalizar la apariencia y funcionalidades de los gráficos, adaptándolos a sus necesidades individuales.

- Elaborar manual de usuario y documentación técnica: Crear un manual de usuario completo y documentación técnica detallada que proporcione a los desarrolladores las instrucciones necesarias para integrar y utilizar la biblioteca de manera efectiva.

- Desarrollar página web demostrativa: Diseñar y desarrollar una página web interactiva que sirva como demostración de las capacidades y funcionalidades de la biblioteca, proporcionando ejemplos prácticos y guías de uso para los usuarios.

- Implementar paletas de colores seguras: Desarrollar paletas de colores seguras para personas con daltonismo y ofrecer una variedad de opciones para adaptarse a diferentes preferencias y necesidades.

3. Planificación

Para la ejecución de este proyecto, se adoptó la metodología Agile, que se caracteriza por su enfoque iterativo e incremental, permitiendo adaptarse a los cambios y nuevas ideas a lo largo del proceso de desarrollo.

Se realizaron sprints con una duración de dos semanas y al final de cada sprint una reunión con la tutora, donde se discutían las tareas pendientes, se establecían nuevos objetivos y se evaluaba el progreso del proyecto.

En cuanto al tiempo dedicado al proyecto, desde la fecha de inicio hasta la fecha de entrega teníamos cerca de 18 semanas, por lo cual optamos por dedicarle 6 horas diarias.

3.1. Diagrama de Gantt

El Diagrama de Gantt adjunto muestra cómo se repartieron las tareas presentes en cada sprint. En esta tabla no indicamos el tiempo dedicado a cada tarea, ya que al trabajar de forma individual, se decidió enfocarse en cumplir con las tareas establecidas para cada sprint.

Tareas	Sprint1 13/09/2023 27/09/2023	Sprint2 27/09/2023 11/10/2023	Sprint3 11/10/2023 25/10/2023
Investigar necesidades accesibles en la web	x		
Aprender Vega-Altair	x	x	
Aprender Vega-Lite		x	
Investigar como crear y publicar una librería	x		
Buscar sobre información de las regulaciones de la accesibilidad web		x	
Buscar proyectos similares		x	
Investigar Entry Points		x	
Investigar sobre la Vega-Lite specification		x	
Crear modelos para albergar parámetros de la Vega-Lite specification			x
Crear tokens donde guardar variables de colore, fuentes y otros		x	
Crear variables especificas por medio de typing		x	
Crear modelos de Charts			x
Crear templates en jinja			x
Crear endpoints de flask		x	

Cuadro 1: Tareas realizadas para el desarrollo de la librería en los sprint uno, dos y tres

Sprints 1-3: Durante los primeros tres sprints, el enfoque se centró en tareas de investigación, destinadas a profundizar en el conocimiento sobre accesibilidad y las librerías relacionadas, como Vega-Altair[15], Vega-Lite[16] y Python. Se llevaron a cabo investigaciones exhaustivas sobre los principios de diseño accesible, así como sobre las mejores prácticas para crear y publicar librerías en Python.

Una vez establecidas las bases teóricas y técnicas, se procedió a la implementación de las tareas básicas de la librería, incluyendo la definición de la estructura de datos necesaria y el proceso de publicación de la librería. Paralelamente, se inició el diseño y desarrollo de la página web de demostración, enfocándose en la creación de endpoints y la elaboración de esquemas para la interfaz de usuario.

Tareas	Sprint4 25/10/2023 08/11/2023	Sprint5 08/11/2023 22/11/2023	Sprint6 22/11/2023 06/12/2023
Buscar información sobre patrones de colores		x	
Investigar sobre las heurísticas accesibles			x
Investigar sobre la Vega-Lite specification	x		
Crear modelos para temas	x		
Crear variables especificas por medio de typing			x
Aprender R basico		x	
Crear una conexion entre R y Python			x
Crear plantillas con Jinja2		x	
Hacer pruebas con Braille R			x
Crear plantillas de código R			x
Generar todos los tipos de graficos	x		

Cuadro 2: Tareas realizadas para el desarrollo de la librería en los sprint cuatro, cinco y seis

Sprints 4-6: Durante estos sprints dedicados al proyecto, nos concentramos en cumplir con los objetivos específicos establecidos, los cuales implicaban el desarrollo de herramientas y recursos destinados a mejorar la accesibilidad de los gráficos generados con Vega-Altair. Una de las tareas principales fue la creación de estructuras de datos que servirían como base para los temas de accesibilidad. Estas estructuras, diseñadas con flexibilidad y coherencia, permitirían a los desarrolladores crear y personalizar sus propios temas según las necesidades específicas de sus proyectos.

Además, surgió la necesidad de implementar funciones adicionales que enriquecieran la experiencia del usuario al interactuar con los gráficos. En este sentido, la idea de desarrollar una función para generar descripciones textuales y un HTML con opciones de personalización cobró fuerza. Si bien estas herramientas no estaban inicialmente contempladas en el plan, se consideró que agregarlas sería beneficioso para ampliar las posibilidades tanto del autor del gráfico como del usuario final. Así, se buscaba ofrecer una experiencia más completa y accesible para todos los usuarios, independientemente de sus capacidades o limitaciones.

Tareas	Sprint7 06/12/2023 20/12/2023	Sprint8 20/12/2023 03/01/2024	Sprint9 03/01/2024 17/01/2024
Documentar TFG	x	x	x
Documentar librería		x	x
Investigar SVG y como meter patrones de colores	x	x	
Deploy de la librería			x
Deploy pagina web			x

Cuadro 3: Tareas realizadas para el desarrollo de la librería en los sprint siete, ocho y nueve

Sprints 7-9: Durante estos últimos tres sprints, con la librería casi finalizada, el enfoque se dirigió a tareas relacionadas con la documentación y la promoción del proyecto. Se dedicó tiempo a completar la página web de demostración, que serviría como vitrina para las capacidades y funcionalidades de la librería. También se elaboró la documentación completa de la librería, que incluía instrucciones detalladas para su integración y uso. Además, se preparó la documentación del proyecto en sí, que abarcaba la metodología utilizada, los objetivos alcanzados y los resultados obtenidos. Finalmente, se llevó a cabo el despliegue de la librería para su uso público, asegurando su accesibilidad y disponibilidad para la comunidad de usuarios.

4. Costes

La estimación de los costos asociados al desarrollo de la librería se fundamenta en la asignación de horas dedicadas a tareas específicas y en el precio por hora de trabajo del equipo de desarrollo. Se presenta a continuación un desglose detallado de los costos:

Proyecto	Horas dedicadas	Sprints dedicados	Semanas	Precio por hora (€)	Total (€)
Librería	300	6	12	18	5400
Pagina web	150	3	6	18	2700
Total	450	9	18	18	8100

Cuadro 4: Tabla de coste por horas trabajadas

4.1. Hosting

Los servicios de alojamiento web, también conocidos como servicios de hosting, son servicios que permiten a individuos y organizaciones publicar su sitio web o aplicación en Internet. Estos servicios proporcionan el almacenamiento de archivos, la capacidad de procesamiento, la conectividad a Internet y otros recursos necesarios para que un sitio web sea accesible en línea.

Para este proyecto, se tomó la decisión de utilizar un servicio de alojamiento web gratuito Render.[11] Esta elección se fundamenta en el hecho de que las páginas web asociadas al proyecto son estáticas y requieren un mínimo de recursos de alojamiento para su funcionamiento adecuado. Al optar por un servicio gratuito, se busca minimizar los costos asociados al desarrollo y mantenimiento del proyecto, sin comprometer la calidad y disponibilidad de las páginas web relacionadas.

Servicios de alojamiento	Precio	Ventajas	Desventajas
AWS(EC2)	$0.0116 / Hora	Paga por lo que uses	Complicado de implementar
Heroku	5€/mes	Fácil implementación para proyectos de Flask	Ofrece servicios que no usaremos
Render	Gratis	Gratis, fácil implementación para proyectos de flask	Estar bajo un subdomino, baja velocidad de navegación

Cuadro 5: Tabla de precios de alojamiento web

4.2. Certificado SSL

Un certificado SSL (Secure Sockets Layer) es un tipo de certificado digital que se utiliza para asegurar la comunicación entre un navegador web y un servidor web. Estos certificados proporcionan cifrado de datos, autenticación del servidor y la integridad de los datos durante la transferencia entre el navegador y el servidor. Cuando un sitio web tiene un certificado SSL instalado, se puede acceder a él utilizando el protocolo HTTPS en lugar de HTTP. Esto indica que la conexión entre el navegador y el servidor está cifrada y es segura.

En este proyecto, se decidió no adquirir ningún certificado SSL debido a la elección de utilizar servicios gratuitos para el alojamiento web. El certificado SSL (Secure Sockets Layer) es una medida de seguridad importante que garantiza que la comunicación entre el navegador web del usuario y el servidor web sea segura y esté cifrada. Sin embargo, los servicios gratuitos de alojamiento web generalmente no incluyen la opción de proporcionar certificados SSL sin costo adicional.

Certificado SSL proveedor	Precio anual
Ionos.es	35€
Godaddy.com	50€
Ssl.com	37€
Hostinger.es	48€

Cuadro 6: Tabla de precios de certificados SSL

4.3. Dominio web

Un dominio de página web es una cadena de caracteres alfanuméricos que constituye la dirección única de un sitio web en Internet. Este dominio se compone generalmente de dos partes: el nombre de dominio específico, que identifica de manera exclusiva al sitio web, y el dominio de nivel superior (TLD), que indica la categoría o tipo del sitio web, como ".com", ".org", ".net", entre otros. Los dominios de páginas web son registrados y gestionados por registradores de dominios, y son utilizados por los usuarios para acceder a un sitio web en particular a través de un navegador web o mediante otros servicios de Internet.

Debido a que el servicio de alojamiento web nos proporciona una dirección web en el formato webname.on.render.com, no vimos la necesidad de comprar un dominio propio. Sin embargo, es importante destacar que Render ofrece la opción de integrar dominios personalizados, lo cual podría ser considerado en el futuro si se desea una mayor personalización o branding para el sitio web. Por el momento, la dirección proporcionada por Render cumple con las necesidades y objetivos para el proyecto.

Dominio(Hostinger)	Precio anual
accessible-theme-altair.es	7€
accessible-theme-altair.com	15€
accessiblethemealtair.info	24€
accessible-theme-altair.tech	49€

Cuadro 7: Tabla de precios de dominio web en Hostinger sin promociones

En cuanto al almacenamiento del código de la librería, no generó ningún costo, dado que tanto GitHub como un repositorio personal y el repositorio oficial de Python (PyPI) son servicios gratuitos y abiertos al público.

5. Análisis

Como parte de la fase analítica durante el desarrollo de la librería, es crucial examinar detalladamente las principales tecnologías involucradas, comprender su funcionamiento y su relevancia en el proyecto. Además, es fundamental describir las estrategias implementadas para abordar las discapacidades visuales y motoras, con el objetivo primordial de garantizar la accesibilidad en los gráficos generados.

5.1. Librerías de Python

Las librerías de Python constituyen elementos fundamentales dentro del ecosistema de desarrollo de software, ofreciendo una amplia gama de funcionalidades y herramientas para abordar diversas áreas de desarrollo de aplicaciones. Estas librerías se componen de módulos y funciones predefinidos que pueden ser importados y utilizados en programas Python, ampliando así su funcionalidad y eficiencia. Existen dos categorías principales de librerías en Python:

- **Librerías Estándar:** Estas librerías están incluidas en la instalación básica de Python y cubren una amplia variedad de funcionalidades, como manipulación de archivos, procesamiento de texto, manejo de excepciones, entre otros. Son mantenidas por el equipo de desarrollo de Python y se consideran seguras y estables.
- **Librerías de Terceros:** Son librerías desarrolladas por la comunidad de programadores de Python para abordar necesidades específicas en diversos campos, como ciencia de datos, desarrollo web, inteligencia artificial, entre otros. Estas librerías son ampliamente utilizadas y contribuyen significativamente a la versatilidad y potencia del ecosistema de Python.

En cuanto a la seguridad, es importante señalar que, a diferencia de otros lenguajes de programación, Python no cuenta con un proceso de verificación o certificación oficial al publicar una librería. Aunque existen medidas y prácticas recomendadas para garantizar la seguridad de las librerías, como las revisiones de código, las pruebas exhaustivas y la documentación adecuada, no hay un mecanismo estándar de comprobación por parte de Python en el proceso de publicación de una librería.

Esta falta de un proceso de verificación centralizado significa que la responsabilidad recae en los desarrolladores y en la comunidad de usuarios para evaluar la seguridad y calidad de una librería antes de integrarla en un proyecto. Es esencial realizar una evaluación exhaustiva de la reputación, el mantenimiento y la retroalimentación de la comunidad antes de adoptar una librería, además de mantenerla actualizada con las últimas revisiones de seguridad.

A pesar de esta limitación, el ecosistema de Python cuenta con una comunidad activa y comprometida que trabaja constantemente para mejorar la seguridad y calidad de las librerías disponibles. Adoptar buenas prácticas de seguridad y estar atento a las actualizaciones y recomendaciones de la comunidad son pasos importantes

para mitigar los riesgos asociados al uso de librerías en proyectos de desarrollo de software.

Para publicar una librería en Python[8], es necesario seguir una serie de pasos definidos. Primero, se debe preparar la librería para la distribución, asegurándose de que esté correctamente empaquetada y documentada. Luego, se procede a registrar la librería en el repositorio oficial de Python, PyPI (Python Package Index)[10], proporcionando metadatos como el nombre de la librería, la versión y una descripción detallada. Una vez completados estos pasos, la librería está lista para ser compartida y utilizada por otros desarrolladores en sus proyectos.

5.2. Vega-Altair

Vega-Altair[15], una librería de visualización de datos para Python, se erige como una herramienta fundamental en el ámbito de la representación gráfica de información. Su existencia y funcionalidad revisten una importancia significativa en el panorama de la ciencia de datos y la generación de gráficos visuales.Internamente, Vega-Altair traduce las especificaciones de gráficos generadas por los usuarios en Python a la sintaxis de Vega-Lite, lo que facilita la generación de gráficos complejos con una sintaxis más simple y amigable.

Vega-Altair simplifica el proceso de creación de gráficos en Python mediante los siguientes pasos formales:

- **Preparación de los Datos:** En esta etapa, se procede a la carga y estructuración de los datos destinados a la visualización. Vega-Altair admite una variedad de tipos de datos, que incluyen archivos CSV, arrays numéricos provenientes de bibliotecas como NumPy y pandas, diccionarios, así como direcciones URL.
- **Especificación del Gráfico:** Se selecciona el tipo de gráfico deseado y se asignan los datos a los canales visuales correspondientes, tales como el eje x, el eje y, el color, entre otros. Esta fase implica la definición precisa de la estructura y la disposición de los datos en el gráfico.
- **Visualización:** Finalmente, se genera el gráfico completo, ya sea como una imagen estática o un gráfico interactivo. Este resultado final será guardado en un JSON denominado Vega-Lite Specification o un dentro de un archivo HTML que usa el JSON para renderizar el gráfico.

5.3. Vega-Lite

Vega-Lite[16] es una biblioteca de visualización de datos de alto nivel, desarrollada en JavaScript, que simplifica la creación de gráficos interactivos en la web. Esta librería se basa en la gramática de visualización de Vega y está diseñada para ser fácil de usar y flexible. Vega-Lite permite a los usuarios crear visualizaciones sofisticadas con un mínimo esfuerzo, utilizando una sintaxis declarativa y un conjunto de reglas

simples para definir la apariencia y el comportamiento de los gráficos.Vega-Lite y Vega nacieron como sucesores de D3, una biblioteca más compleja y de bajo nivel, proporcionando una abstracción más intuitiva y accesible para la creación de visualizaciones de datos.

Entre las características principales de Vega-Lite tenemos:

- **Sintaxis Declarativa:** Vega-Lite utiliza una sintaxis declarativa que permite a los usuarios especificar el tipo de gráfico y los datos a visualizar de manera intuitiva y concisa. Esto simplifica el proceso de creación de gráficos y facilita la comprensión del código.
- **Interactividad:** Vega-Lite ofrece soporte para interactividad en las visualizaciones, lo que permite a los usuarios explorar y analizar los datos de forma dinámica. Esto incluye funciones como zoom, filtro y resaltado de datos, así como herramientas de navegación y exploración.
- **Compatibilidad con la Web:** Las visualizaciones creadas con Vega-Lite son compatibles con los estándares web modernos, lo que significa que se pueden integrar fácilmente en aplicaciones web y sitios web existentes. Esto proporciona una forma efectiva de comunicar información y análisis a través de la web.

5.4. BrailleR

BrailleR[17] es una biblioteca especializada diseñada para mejorar la accesibilidad de las visualizaciones de datos en el entorno de programación R. A través de diversas funciones y herramientas, BrailleR permite generar descripciones textuales detalladas de gráficos, facilitando su comprensión para usuarios con discapacidad visual o aquellos que dependen de tecnologías de asistencia.

En el contexto de este proyecto, hacemos uso de la funcionalidad proporcionada por BrailleR para generar descripciones textuales de los gráficos creados utilizando Vega-Altair en Python. Dado que BrailleR es una biblioteca destinada al lenguaje de programación R, requerimos un medio para integrar su funcionalidad con el entorno de desarrollo en Python. Para lograr esta integración, empleamos la biblioteca pyRserve[18] como un puente de comunicación entre Python y R. pyRserve facilita la ejecución de comandos de R desde Python y la transferencia de datos entre ambos entornos, permitiendo así el acceso a las capacidades de BrailleR dentro de este proyecto en Python.

Mediante esta integración entre BrailleR y este proyecto en Python, garantizamos la generación de descripciones accesibles para los gráficos creados, contribuyendo así a mejorar la inclusión y la accesibilidad en la visualización de datos.

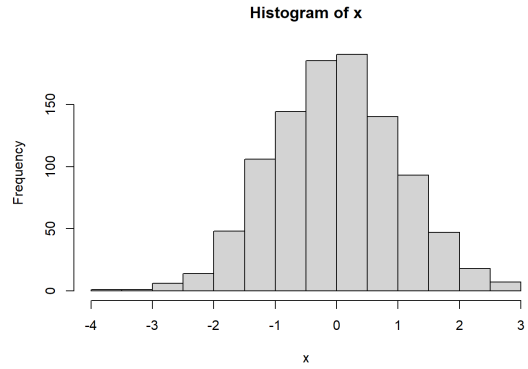

Figura 1: Histograma de 1000 valores aleratorios generado en R

```
## This is a histogram, with the title: with the title: Histogram of x
## "x" is marked on the x-axis.
## Tick marks for the x-axis are at: -4, -3, -2, -1, 0, 1, 2, and 3
## There are a total of 1000 elements for this variable.
## Tick marks for the y-axis are at: 0, 50, 100, and 150
## It has 14 bins with equal widths, starting at -4 and ending at 3 .
## The mids and counts for the bins are:
## mid = -3.75  count = 1
## mid = -3.25  count = 1
## mid = -2.75  count = 6
## mid = -2.25  count = 14
## mid = -1.75  count = 48
## mid = -1.25  count = 106
## mid = -0.75  count = 144
## mid = -0.25  count = 185
## mid = 0.25  count = 190
## mid = 0.75  count = 140
## mid = 1.25  count = 93
## mid = 1.75  count = 47
## mid = 2.25  count = 18
## mid = 2.75  count = 7
```

5.5. Jinja2

Jinja2 es una potente biblioteca de templates para Python, que permite la generación dinámica de documentos HTML, XML y otros formatos de texto.Jinja2 se basa en el concepto de plantillas, que son archivos con código mezclado con marcadores que indican dónde se deben insertar datos dinámicos. Estos marcadores, conocidos como expresiones de plantilla, pueden incluir variables, bucles, condicionales y otros constructos de programación. Jinja2 procesa estas plantillas y las combina con datos proporcionados en tiempo de ejecución para generar el contenido final que se enviará al navegador web o a otra salida.

En el proyecto, Jinja2 se ha utilizado para crear plantillas de HTML que posibilitan la generación de contenido dinámico al combinarlas con datos proporcionados en tiempo de ejecución. Estas plantillas se emplean para representar gráficos utilizando el Vega-Lite specification, así como otros datos relevantes para la visualización. Además de incluir los datos necesarios, las plantillas también contienen funciones vinculadas a botones y controles deslizantes en el HTML, lo que permite a los usuarios cambiar la apariencia del gráfico de manera interactiva. Esta integración de Jinja2 ha permitido generar páginas web dinámicas y personalizables que mejoran la experiencia del usuario al interactuar con los gráficos generados.

5.6. Análisis de accesibilidad

5.6.1. Análisis de accesibilidad

Para garantizar la accesibilidad en la creación de gráficos, el proyecto se baso en las heurísticas propuestas por Alcaraz. Estas heurísticas proporcionan una guía sólida para evaluar y mejorar la accesibilidad de las visualizaciones de datos. El análisis implica estudiar y comparar estas heurísticas con las funcionalidades proporcionadas por la librería Vega-Altair. A continuación, se describe cómo se realizó este análisis:

5.6.2. Heurísticas de Alcaraz:

Las heurísticas de Alcaraz[5] ofrecen criterios específicos que deben cumplirse para garantizar la accesibilidad de un gráfico para personas con baja visión. Estas incluyen consideraciones como el uso de textos alternativos, el contraste de colores adecuado y la compatibilidad con tecnologías de asistencia.

Evaluación Comparativa con Vega-Altair:

- **Heurísticas Incluidas**: Identificamos algunas heurísticas que ya están incorporadas en la funcionalidad de Vega-Altair, lo que nos permite aprovechar estas características sin necesidad de modificaciones adicionales.
- **Heurísticas Integrables con el TFG**: Identificamos otras heurísticas que pueden integrarse con esta librería para mejorar la accesibilidad de los gráficos generados. Esto implica desarrollar nuevas funcionalidades o ajustar las existentes para cumplir con estos criterios.
- **Heurísticas no aplicables**: Reconocemos finalmente la heurísticas que no son factibles de integrar en esta librería debido a limitaciones técnicas o de diseño. Estas heurísticas pueden requerir acciones adicionales por parte del desarrollador para garantizar su cumplimiento.

ID	Nombre	Definición	Estado
H1	Titulo	El gráfico debe contar con un título breve y descriptivo que ayude al lector a identificarlo y diferenciarlo del resto que aparecen en la misma página, permitiéndoles navegar así entre ellos.	Incluida
H2	Leyenda	El gráfico debe ofrecer leyendas que asocian unívocamente esquemas de color, patrones o formas utilizados en el gráfico con sus respectivas variables, siempre que sea necesario.	Incluida
H3	Ejes	Los ejes del gráfico deben ser visibles y sus etiquetas adecuadas concisas y claras.	Incluida
H4	Pie	El gráfico debe contar con un pie que contribuya a facilitar la compresión de la información que se comunica, a excepción de que se trate de un gráfico muy simple que no lo precise.	No aplicado
H5	Abreviaturas	Las abreviaturas utilizadas se desarrollan para facilitar su compresión.	No aplicado
H6	Fuente de datos	Junto al gráfico, preferiblemente cerca del pie, se ofrece información acerca de la fuente de los datos (institución, conjunto de datos, fecha y enlace)	No aplicado
H7	Version impresa	Se ofrece una versión del gráfico optimizada para impresión para aquellos usuarios que prefieran la consulta en este otro medio	TFG
H8	Alternativa textual	El gráfico cuenta con un texto alternativo que informa brevemente sobre el contenido del gráfico y ayuda a los usuarios a determinar si desean obtener más información al respecto. Si existe una descripción larga asociada al gráfico, se informa sobre ella.	TFG
H9	Descripción larga	Todo gráfico complejo cuenta con una descripción larga que ofrece información equivalente.	TFG
H10	Colores seguros	Los colores empleados en el gráfico para transmitir información pueden ser distinguidos por las personas con diferentes perfiles de VCD.	TFG
H11	Contraste	Entre el texto y el fondo existe un contraste mínimo de 4,5:1. Entre los elementos no textuales (líneas, barras, etc.) adyacentes del gráfico, existe un contraste mínimo de 3:1.	TFG
H12	Legibilidad	El diseño del gráfico se basa en criterios de composición de textos que aseguran una correcta legibilidad (fuente tipográfica, interlineado, espacio entre letras y palabras, uso comedido de mayúsculas y versalitas, etc.)	TFG
H13	Calidad de imagen	Todos los gráficos, en particular los gráficos en formato de mapa de bits cuentan con una calidad suficiente para su correcta visualización y admiten un zoom mínimo de 200 % sin desenfocarse, ni pixelarse.	Incluida

Cuadro 8: Heurísticas de Alcaraz del H1 a H13

ID	Nombre	Definición	Estado
H14	Redimensionado	El gráfico se puede ampliar hasta un 200 % sin la necesidad de contar con ninguna ayuda técnica diferente al navegador. Una vez ampliado, ningún elemento se superpone al gráfico y todas sus funcionalidades originales continúan estando disponibles.	TFG
H15	Sin obstáculos en la visualización	En ocasiones, los editores de contenido utilizan elementos como marcas de agua o de autoría que superponen a las imágenes de texto, SVG o Canvas, dificultando su lectura. El objetivo de este indicador es asegurar que ningún elemento externo al gráfico dificulta o impide parcial o totalmente su lectura	Incluida
H16	Foco visible	Se muestra un indicador visible para todos aquellos elementos del gráfico que puedan recibir el foco del ratón, teclado o táctil. Este criterio no aplica a los gráficos en formato de mapa de bits, sino sólo a los gráficos en formato SVG o implementados con bibliotecas de JavaScript.	No aplicado
H17	Navegación independiente del dispositivo	Tanto en dispositivos de sobremesa, como en dispositivos móviles es posible navegar mediante distintas interfaces (ratón, teclado o gestos táctiles). Para cumplir este indicador, los tipos de eventos o interacciones disponibles deben ser identificables y poder operarse mediante los diferentes tipos de interfaces disponibles. Este criterio no aplica a los gráficos en formato de mapa de bits, sino sólo a los gráficos en formato SVG o implementados con bibliotecas de JavaScript.	Incluida
H18	Personalizacion	Se permite a los usuarios personalizar las características de los gráficos (paleta de colores, contraste, familia tipográfica, tamaño del texto, interlineado, etc.). Cumplir con este indicador implica el uso de tecnologías compatibles con las ayudas técnicas pensadas para para este cometido. Alternativamente, es posible alcanzar el indicador, ofreciendo un sistema propio que permita personalizar la apariencia del gráfico.	TFG

Cuadro 9: Heurísticas de Alcaraz del H14 a H18

5.6.3. Estrategias Generales contra el daltonismo

Al considerar la heurística H10 de colores seguros para la accesibilidad de los gráficos para usuarios con daltonismo, se han identificado varias estrategias que pueden mejorar la legibilidad y la comprensión de los gráficos para este grupo de usuarios. Estas estrategias incluyen:

- **Cambio del Hue de la Paleta de Colores:** Ajustar los colores utilizados en los gráficos para evitar combinaciones problemáticas para personas con daltonismo. Esto implica cambiar el tono de los colores para mejorar la distinción entre ellos.
- **Cambio de la Iluminación entre Colores:** Modificar la luminosidad o la saturación de los colores para aumentar el contraste y hacer que los colores sean más distinguibles para las personas con daltonismo.
- **Integración de una Paleta de Colores con Patrones de Relleno:** Utilizar patrones de relleno en lugar de colores sólidos para representar diferentes

categorías o datos en los gráficos. Esto proporciona una forma adicional de diferenciar entre elementos visuales, independientemente de la percepción del color.

5.6.4. Métodos Específicos para crear patrones de relleno

Para la integración de patrones de relleno, existen varias herramientas y métodos disponibles. Además de varias investigaciones[6][9] que determinan la mejor forma de implementarlos Algunos de estas ideas en las investigaciones son:

- **ColourNames**: Una herramienta que asigna nombres a los colores y puede ser útil para identificar colores que pueden ser problemáticos para personas con daltonismo.
- **ColourMeters**: Una herramienta que permite medir y comparar la percepción de los colores por parte de personas con diferentes tipos de daltonismo.
- **ColourIconizer**: Una herramienta que convierte los colores en iconos o patrones de relleno para mejorar la distinción entre ellos.
- **ColourMix**: Un método que combina diferentes colores y patrones de relleno para crear paletas de colores accesibles y distinguibles para personas con daltonismo.

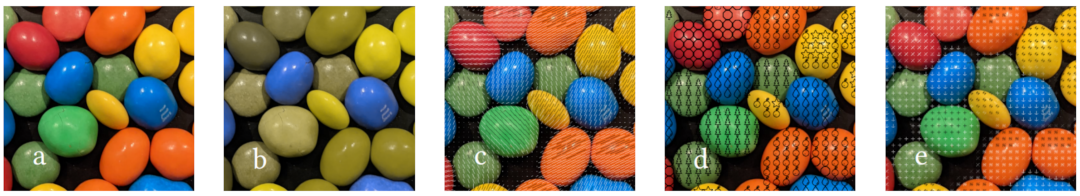

Figura 2: Figura a: Imagen normal. figura b: Imagen para alguien con daltonismo. Figura c: Imagen aplicando ColorMeters. Figura d: Imagen aplicando ColourIconizer. Figura e:Imagen aplicando ColourMix

De estas estrategias, se probaron todas; sin embargo, mientras todas resultaron efectivas, se observó que ColourNames saturaba el gráfico de letras, ColourMeters y ColourMix requerían una leyenda adicional o información previa para comprender el significado del patrón. Por lo tanto, se optó por utilizar ColourIconizer, ya que resultaba más intuitivo para los usuarios asociar un color con un objeto específico.

5.6.5. Navegación por teclado

Como estrategia para personas con discapacidad motora, se considera la navegación por teclado como una opción fundamental. De esta manera, al visualizar un gráfico

utilizando la función que genera un HTML, los usuarios tienen la posibilidad de navegar por la interfaz sin necesidad de utilizar el mouse. Entre las implementaciones empleadas se encuentran las siguientes:

- Inclusión del atributo tabIndex en los elementos del HTML: Este atributo permite la navegación utilizando la tecla de tabulación del teclado.
- Garantizar que los botones y elementos clicables sean también accesibles mediante el teclado: En particular, se ha configurado las teclas de dirección y la tecla Enter para ejecutar acciones sobre estos elementos.

5.7. Análisis para la pagina web de demostración

Dada la naturaleza del proyecto como una librería de visualización de datos, surgió la necesidad de desarrollar una página web que demostrara las capacidades de la librería y permitiera a los usuarios interactuar con los gráficos generados. Con este propósito en mente, se han establecido las siguientes decisiones de diseño y tecnología para la construcción de la página web:

- Backend con Flask: Se optó por utilizar Flask, un framework de Python para desarrollo web, para construir el backend de la página web. Flask ofrece una manera sencilla y flexible de manejar las solicitudes HTTP y servir el contenido dinámico de la página web.
- Frontend con Vanilla JavaScript y Bootstrap: Para el frontend de la página web, se decidió utilizar JavaScript puro (Vanilla JavaScript) para la lógica de interacción del usuario y Bootstrap para el diseño y la interfaz de usuario. Esto permitió crear una página web minimalista pero atractiva, con un diseño responsivo y fácil de usar.

La elección de estas tecnologías se fundamentó en el conocimiento previo adquirido durante la carrera, lo que facilitó el proceso de desarrollo al evitar la necesidad de aprender nuevas herramientas. Al optar por tecnologías familiares, se optimizó el tiempo y los recursos disponibles, permitiendo una mayor eficiencia en la ejecución del proyecto, porque cumplían el cometido deseado.

5.8. Consideraciones adicionales para el desarrollo de la librería

- **Compatibilidad y No Alteración de Vega-Altair**: Nos enfrentamos a la limitación de garantizar la compatibilidad y no alterar la funcionalidad de la librería Vega-Altair. Dado que Vega-Altair es una librería de código abierto y su desarrollo está en constante evolución, no podemos prever todas las posibles versiones y cambios que puedan surgir. Por lo tanto, optamos por trabajar con la última versión disponible en el momento del desarrollo y asegurar que esta librería funcione correctamente con ella.

- **Limitaciones con Canvas:** Dado que Vega-Altair emplea el elemento Canvas para la generación de gráficos, se tomó la decisión de implementar elementos SVG (Scalable Vector Graphics) en su lugar. Esta elección se fundamenta en la preferencia por SVG en términos de accesibilidad y desarrollo web[19]. Los gráficos SVG son más adecuados para la creación de elementos accesibles, ya que permiten la inclusión de etiquetas semánticas y texto descriptivo, lo que facilita su interpretación por parte de tecnologías de asistencia y mejora la experiencia de usuario para aquellos con discapacidades visuales.

- **Restricciones con BrailleR y Tipos de Gráficos:** En esta version del proyecto nos encontramos con limitaciones en la cantidad y tipos de gráficos que podemos pasar a R utilizando BrailleR. No todos los gráficos generados con Vega-Altair son compatibles con BrailleR, por lo que nos limitamos a trabajar con los gráficos básicos, como barras, pie charts, puntos de dispersión y líneas, que son los más compatibles y fácilmente interpretables por BrailleR.

5.9. Requisitos Técnicos

Librería Altair-Easeviz:

- **Plataforma y Entorno**: Altair-Easeviz debe ser compatible y utilizable en cualquier sistema operativo o entorno de desarrollo compatible con Python.

- **Integración No Invasiva**: La librería no alterará el código original de Vega-Altair, garantizando una integración no invasiva y preservando las funcionalidades originales.

- **Modelos de Datos Conformes**:Altair-Easeviz utilizará modelos de datos que cumplan con las propiedades especificadas en la Vega-Lite specification.

- **Edición de Propiedades**: Se implementarán funciones en los modelos de datos para la edición de propiedades, manteniendo la coherencia con la especificación Vega-Lite.

- **Documentación y Comentarios**:Los modelos de datos y sus funciones asociadas estarán debidamente escritos, documentados y comentados, facilitando su comprensión y extensión por parte de los desarrolladores.

- **Funciones en HTML**:Se incluirán funciones en el HTML generado para permitir la personalización del gráfico, mejorando su accesibilidad y legibilidad.

- **Generación de Descripciones**: Altair-Easeviz proporcionará funciones para la generación de descripciones textuales de gráficos, permitiendo también que los desarrolladores incluyan descripciones personalizadas.

- **Integración de Heurísticas de Alcaraz**:Se buscará integrar las heurísticas de accesibilidad propuestas por Ruben Alcaraz en la medida de lo posible en todos los modelos de datos y funciones.

- **Transparencia en Variables**: Evitaremos el uso de variables privadas u ocultas, permitiendo que los desarrolladores las modifiquen según sus necesidades.
- **Accesibilidad en HTML**: Los gráficos generados en HTML con funciones de Altair-Easeviz deben ser navegables por teclado y compatibles con lectores de pantalla.

Página Web de demostración:

- **Cumplimiento de Estándares WAI**: La página web se desarrollará siguiendo la guía de estilos y estándares de la Web Accessibility Initiative (WAI).
- **Configuraciones Visuales**: Mostrará configuraciones visuales aplicables a todos los gráficos generados por Vega-Altair.
- **Gráficos Accesibles**: Exhibirá gráficos que utilicen funciones de Altair-Easeviz para generar HTML accesible y comprensible.
- **Referencias a Recursos**: Se referenciarán los recursos utilizados, como paletas de colores, guías de estilo y estándares, para mayor transparencia y reconocimiento.

5.10. Requerimientos no técnicos

Librería Altair-Easeviz:

- **Diseño Responsive**: La librería Altair-Easeviz considera el diseño responsive en la medida de lo posible al integrarse con Bootstrap y hacer uso de sus capacidades de grid. La adaptabilidad en entornos móviles y de diferentes tamaños de pantalla se facilita, aunque se deja parte de esta responsabilidad a los desarrolladores.
- **Accesibilidad**: La librería sigue las heurísticas de accesibilidad de Alcaraz en la medida de lo posible para los gráficos. En cuanto a la web, se adhiere a las guías de estilo y estándares de la WAI, asegurando una experiencia accesible para todos los usuarios.
- **Escalabilidad**: Altair-Easeviz es altamente escalable al ser código abierto y exponer su modelo de datos. Esto permite su adaptabilidad a proyectos de diferentes complejidades, fomentando la expansión y contribución de la comunidad.
- **Licencia Open Source**: Altair-Easeviz se distribuye bajo una licencia de código abierto, permitiendo su libre uso, modificación y contribución. Esto favorece su integración en diversos proyectos y la colaboración entre desarrolladores.

Página Web de demostración:

- **Diseño Responsive**: La página web de ejemplos también aborda el diseño responsive al hacer uso de Bootstrap y su sistema de grid. Esto garantiza una presentación adecuada en diversos dispositivos y tamaños de pantalla.
- **Accesibilidad**: La página web sigue las pautas de accesibilidad web de la WAI, asegurando que la información y los gráficos sean accesibles para usuarios con diversas capacidades.

5.11. Usuarios

Durante el desarrollo de la librería de visualización de datos, identificamos dos tipos principales de usuarios, cada uno con diferentes necesidades y habilidades:

Desarrollador: Descripción: Este usuario es el encargado de crear y personalizar los gráficos utilizando esta librería. Tiene conocimientos avanzados en Python y matemáticas, y puede desempeñar roles como científico de datos, analista de datos o desarrollador web.

Características Clave:

- Experiencia en programación en Python.
- Conocimiento sólido en matemáticas y estadísticas.
- Familiaridad con la manipulación de datos y la visualización de datos.
- Capacidad para interpretar y analizar gráficos para la toma de decisiones.

Usuario Final: Descripción: Este usuario es aquel que visualizará y analizará los gráficos creados por los desarrolladores. Puede ser cualquier persona, desde colegas y superiores en una empresa hasta el público en general que accede a un sitio web. Consideramos tanto a los usuarios normales como a aquellos con discapacidades visuales o motoras.

Características Clave:

- Diversidad en habilidades y conocimientos, desde usuarios con experiencia en análisis de datos hasta aquellos con poca familiaridad con la visualización de datos.
- Posibles limitaciones físicas o cognitivas que requieren una consideración especial para garantizar la accesibilidad de los gráficos.

6. Diseño

6.1. Arquitectura de la librería

La arquitectura de Altair-Easeviz se basa en un enfoque modular, lo que facilita su integración con la librería Vega-Altair[21]. Esta estructura se organiza en capas, separando las funcionalidades de accesibilidad y manipulación de datos. La clara delimitación de responsabilidades entre estas capas promueve una arquitectura cohesiva y adaptable. Además, se emplean modelos de datos bien definidos para la interacción, lo que permite una fácil extensión y adaptación del sistema a nuevas versiones de Vega-Altair. Las funciones están meticulosamente separadas, lo que permite su tratamiento independiente y una mayor modularidad en el desarrollo

6.2. Estructura de código

La estructura del código de Altair-Easeviz se ha diseñado con claridad y modularidad, siguiendo las mejores prácticas de organización y documentación. A continuación, se detalla la estructura de directorios y archivos:

Assets, contiene recursos gráficos, como imágenes e iconos, utilizados en la documentación.

Docs, almacena archivos Markdown que componen la documentación detallada de la librería.

Altair_easeviz, esta carpeta encapsula el núcleo de Altair-Easeviz.

- **Models**, contiene descripciones detalladas de los modelos de datos esperados por la especificación de Vega-Lite. Estas definiciones facilitan la comprensión y manipulación de las propiedades necesarias para las configuraciones visuales de los gráficos generados por Altair-Easeviz.
- **Description_generator**, archivo Python que aloja la lógica encargada de generar descripciones textuales de los gráficos. Contribuye a la accesibilidad al proporcionar información descriptiva.
- **Themes**, archivo Python que inicializa los modelos de datos relacionados con temas visuales. Contribuye a la personalización de la apariencia de los gráficos
- **Tokens**, archivo Python que contiene variables, paletas de colores, tamaños de fuentes y demás configuraciones utilizadas en Altair-Easeviz.
- **Types_theme**, archivo Python que describe nuevos tipos y alias, ofreciendo sugerencias y asegurando la coherencia de los tipos de datos utilizados en la librería.
- **Utils**, archivo Python que alberga funciones esenciales para generar HTML accesible, proporcionando una capa de utilidades importante para la accesibilidad.

Requirements.txt, archivo que enumera las librerías utilizadas durante el desarrollo de Altair-Easeviz. No especifica las librerías necesarias para su funcionamiento.

Setup.py, fichero de configuración de la librería que describe las dependencias necesarias, meta-datos como el autor, versión y descripción, y la versión de Python compatible.

6.3. Diagrama de los modelos o estructura de datos

En la librería, el primer paso consistió en modificar la apariencia predeterminada de los gráficos para hacerlos más accesibles. Para lograr este objetivo, se crearon estructuras de datos que contienen las principales claves presentes dentro de la clave **config** del Vega-Lite specification[20]. Esto permite organizar de manera más efectiva qué atributos deseamos cambiar o mantener. El enfoque se centró en las siguientes claves:

ConfigModel

Representa la configuración visual general del gráfico, incluyendo ejes, fondo, encabezado, leyenda, rango, título y vista. Actúa como contenedor para otros modelos específicos.

AxisModel

Define las propiedades del eje en el gráfico, como el color de las etiquetas, el fondo, la fuente, las marcas y otros parámetros relacionados con los ejes.

HeaderModel

Contiene las propiedades del encabezado, especificando la fuente, el tamaño, el color y otros atributos relacionados con la presentación del encabezado.

LegendModel

Define las propiedades de la leyenda del gráfico, como la fuente, el tamaño, el color y otros parámetros relevantes para la visualización de la leyenda.

RangeModel

Gestiona las configuraciones de rango en el gráfico, incluyendo paletas de colores para mapas de calor, rampas, categorías y divergencias.

TitleModel

Especifica las propiedades del título del gráfico, como el color, el tamaño de la fuente, el peso y otros atributos relacionados con la presentación del título.

ViewModel

Representa la vista del gráfico, definiendo propiedades como el ancho continuo, el trazo y la altura continua.

MarkModel

Modelo adicional para gestionar propiedades específicas de las marcas en el gráfico (barras, líneas, puntos, textos, reglas, formas, texto), como el color, el trazo, el relleno y otros atributos relacionados con las marcas visuales.

Todas estas estructuras se utilizan dentro de otra denominada Config, lo que permite representar un modelo similar al del Vega-Lite specification.

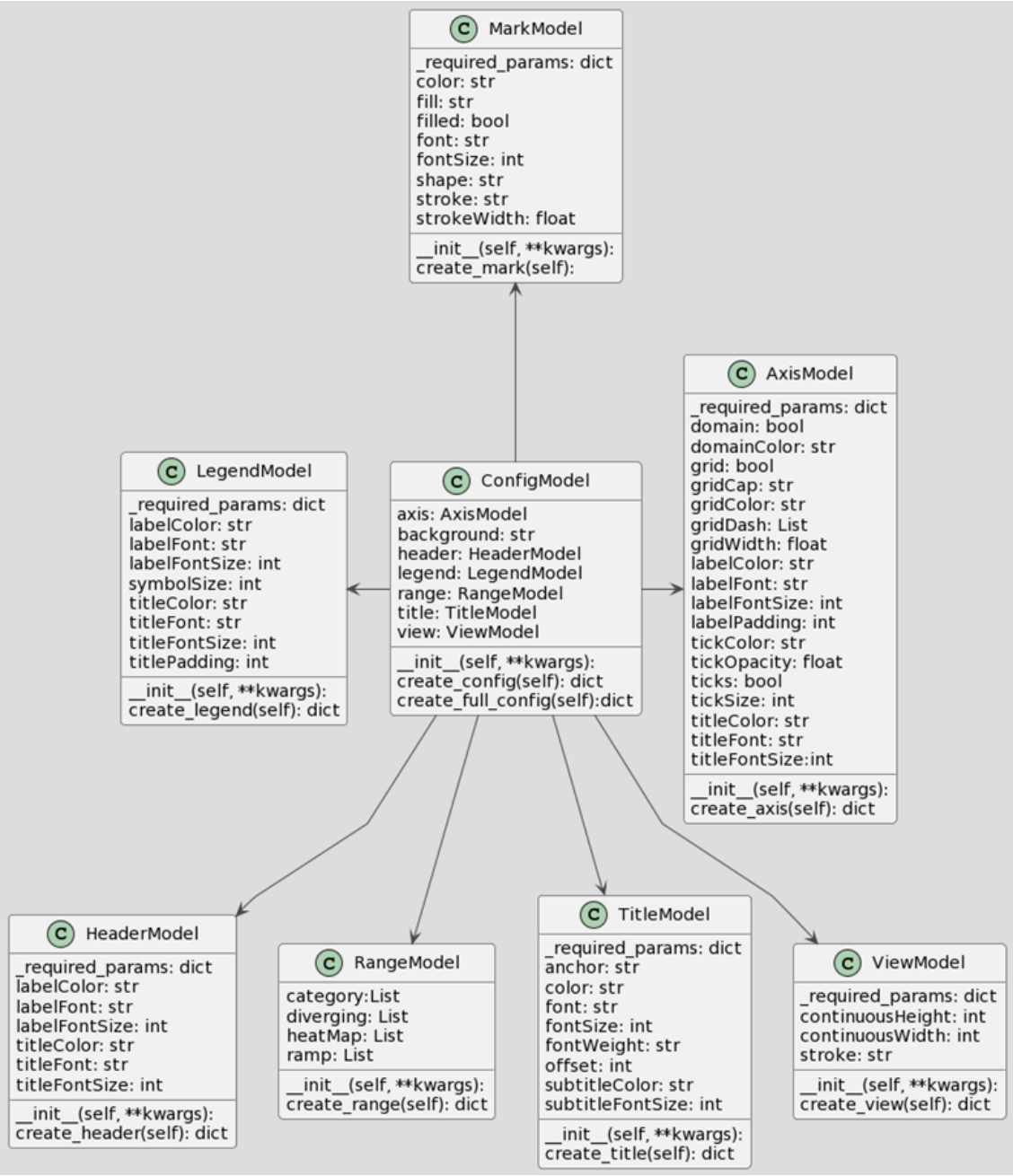

Figura 3: Representación de las estructuras de datos en PlantUML

Para iniciar estas estructuras y devolverlas de manera consistente, implementamos dos funciones principales:

- init: Este método se llama al instanciar la estructura y acepta solo kwargs. Todos los parámetros que se pasan son comprobados con variables de tipo específicas para garantizar que contengan las mismas claves que el tipo y sean del mismo tipo. Si alguna clave descrita no está en el tipo de variable, se

inicializa una variable con esa clave con un valor por defecto establecido por nosotros. Esto asegura que al devolver esta estructura, se garantice que las claves contenidas sean las esperadas.

- create_nombredeestructura: Esta función se encarga de devolver los atributos de la estructura que fueron inicializados en init. Aquí también se comprueba que las claves internas sean del mismo tipo que las establecidas en el tipo de variable.

Adicionalmente, creamos una estructura llamada ModelTheme que encapsula todas las configuraciones visuales de un gráfico. Esta estructura permite la creación de temas personalizados, como temas accesibles, oscuros, con patrones de relleno y amigables para la impresión. ModelTheme ofrece al desarrollador la opción de iniciar todas estas variables de manera más sencilla, permitiendo definir fácilmente los colores utilizados en el gráfico y las funciones que afectan al tamaño de la fuente, así como cambiar cualquier color de los elementos en el gráfico, ya sea el color de fondo o el de las letras.

En esta librería, se crearon cuatro modelos adicionales que heredan de ModelTheme, cada uno con funciones y atributos específicos:

- AccessibleTheme
- DarkAccessibleTheme
- FillerPatternTheme
- PrintFriendlyTheme

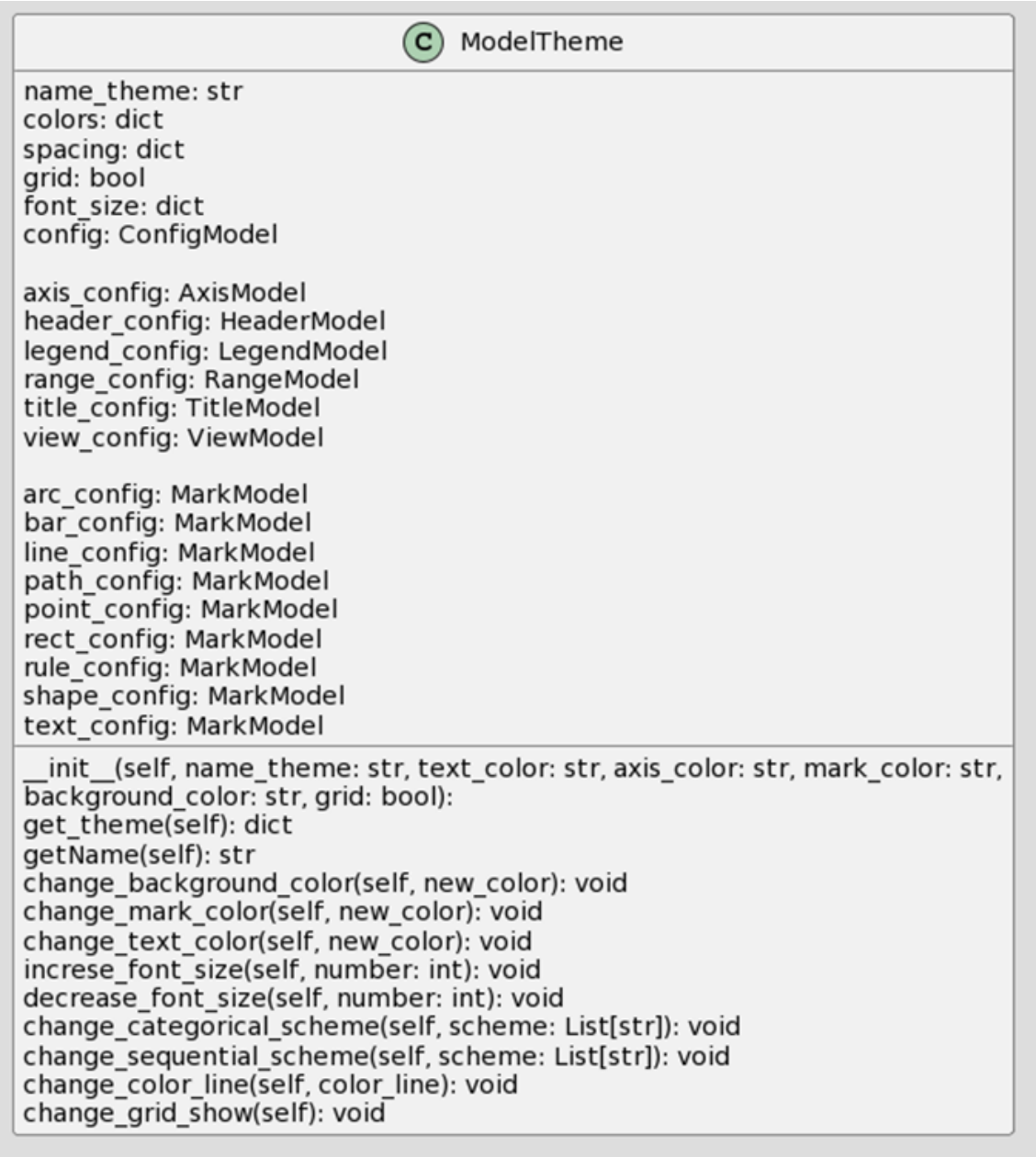

Figura 4: Representación en PlantUML de la estructura de ModelTheme

6.4. Typing

En este proyecto, se implemento varios tipos de variables personalizados utilizando la librería de typing[22]. Esta práctica permite establecer una estructura más sólida y consistente en los datos, garantizando que las estructuras de datos devueltas contengan las mismas claves internas.

Al utilizar tipos de variables personalizados, se evita la introducción de claves inesperadas durante el renderizado del gráfico, lo que contribuye a la robustez y confiabilidad del código. Además, el uso de typing mejora la legibilidad del código al

proporcionar información clara sobre los tipos de datos esperados en cada contexto. Esta implementación refuerza la integridad de los datos y facilita el mantenimiento del proyecto a lo largo del tiempo.

6.5. Tokens

En la librería, se han establecido varias variables globales que se utilizan en toda la extensión del proyecto. Estas variables incluyen :

- paletas de colores
- colores primitivos
- tamaños de fuentes
- tamaño de espaciado

Al definir estas variables como tokens, se garantiza una consistencia en la apariencia y el estilo en todas las visualizaciones producidas por esta librería. Además, al centralizar estas configuraciones en tokens globales, facilitamos la personalización y la adaptación del diseño según las necesidades específicas de cada proyecto. Esta práctica también promueve la cohesión y la mantenibilidad del código al proporcionar un punto de referencia único para ajustes y modificaciones en el diseño visual de los gráficos. En resumen, el uso de tokens contribuye a una experiencia de usuario más consistente, flexible y fácil de mantener en toda la librería.

6.6. Diagrama de la conexión a R

En la librería, se integro la funcionalidad de conexión con R para aprovechar la potencia de la librería BrailleR[17], la cual ofrece herramientas para generar descripciones de gráficos. Sin embargo, dado que BrailleR es una librería de R, su instalación no es posible mediante el uso convencional de pip. Como resultado, se requieren ciertos pasos previos para habilitar esta función en la librería.

Para utilizar la función de descripción de gráficos, los desarrolladores deben seguir los siguientes pasos:

- **Instalar R**: Se debe instalar el software R en el sistema donde se ejecutará la librería.
- **Instalar los paquetes Rserve, BrailleR y ggplot**: Estos son los paquetes de R necesarios para habilitar la funcionalidad de descripción de gráficos. Rserve actúa como un puente entre Python y R, mientras que BrailleR proporciona las herramientas de descripción y ggplot se utiliza para construir los gráficos en R.

- **Ejecutar R y la librería Rserve**: Una vez instalados los paquetes necesarios, se debe ejecutar R y cargar la librería Rserve para establecer la conexión entre Python y R.

En cuanto a los gráficos disponibles que implementados, se limito a utilizar los más básicos, incluyendo:

- Gráficos de barras
- Gráficos de puntos de dispersión
- Gráficos de líneas
- Gráficos circulares

6.7. Estructura del HTML con Jinja2

En esta implementación, se ha ampliado la forma en que se presenta el gráfico en el HTML. Mientras que Vega-Altair normalmente genera un gráfico dentro de un tag de Canvas y un pequeño script, se diseño un template más completo utilizando Jinja2.

Dentro de este template, se agrego una variedad de elementos adicionales que permiten personalizar la apariencia del gráfico de manera más completa. Por ejemplo, se incluiyo opciones para cambiar la paleta de colores del gráfico, con especial atención en paletas diseñadas para mejorar la legibilidad para personas con daltonismo. Además, proporcionamos opciones para ajustar el tamaño del texto en el gráfico y las dimensiones generales del mismo.

El uso de Jinja2 permitió crear un template HTML dinámico y flexible, que ofrece una experiencia de usuario más mejorada y personalizable en comparación con la visualización estándar de Vega-Altair.

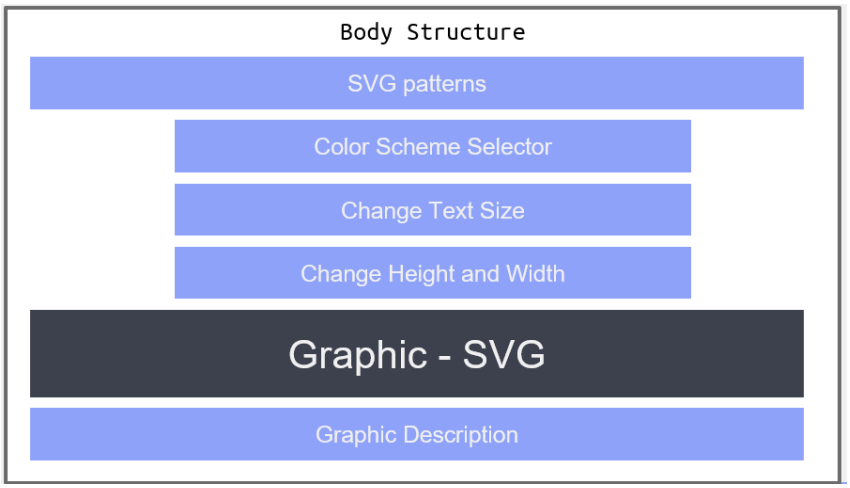

Figura 5: Estructura del HTML generado con Jinja2

6.8. Diagramas de los modelos para la página web

Para la página web de demostración, se ha aplicado el patrón de diseño Factory para gestionar la creación de los diferentes tipos de gráficos. Este patrón permite encapsular la lógica de creación de objetos y proporcionar una interfaz común para crear diversos tipos de gráficos.

En el corazón de este diseño se encuentra la clase ChartBase, que sirve como una plantilla base para todos los tipos de gráficos. Esta clase define dos atributos principales: el origen de los datos y la función defineChart(), encargada de establecer cómo se presentarán los datos en el gráfico.

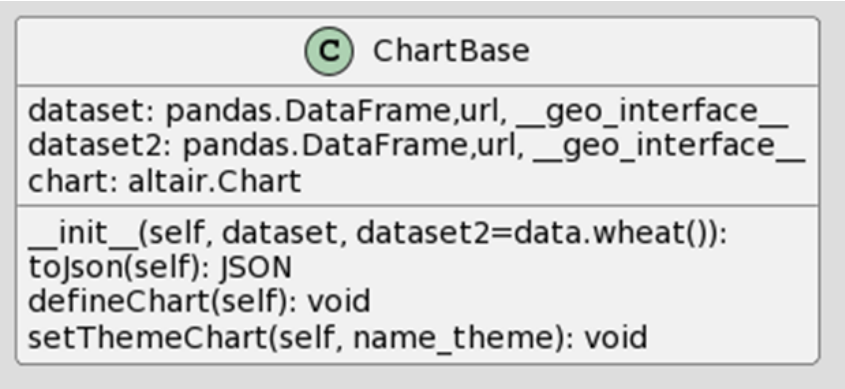

Figura 6: Estructura del modelo ChartBase

Luego, se han creado clases específicas para cada tipo de gráfico, como BarChart, LineChart, etc. Estas clases heredan de ChartBase y sobrescriben el método defineChart() para adaptar la presentación del gráfico según el tipo específico.

El uso del patrón Factory permite centralizar la lógica de creación de objetos y facilita la extensibilidad de la página web. Si en el futuro se desean agregar nuevos tipos de gráficos, simplemente se tendrán que crear nuevas clases que sigan el mismo patrón y no será necesario modificar el código existente.

6.9. Diagrama de la API de la página web

En la sección de la API de la página web, se implementaron una serie de endpoints que permiten generar gráficos dinámicos y acceder a diferentes visualizaciones de datos. Estos endpoints ofrecen una forma sencilla y eficiente de interactuar con la librería y explorar las opciones de personalización disponibles. A continuación, se detallan los principales endpoints disponibles:

- $Home: Este endpoint representa la página principal de la API. Durante el desarrollo, se accede a través de la IP 127.0.0.1:5000, mientras que en producción se accede mediante la URL del sitio https://accesible-theme-altair.onrender.com/.
- $Home/API/barchart: Endpoint para generar gráficos de barras. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.

- $Home/ API/linechart: Endpoint para generar gráficos de líneas. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/areachart: Endpoint para generar gráficos de áreas. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/circularplots: Endpoint para generar gráficos circulares. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/scatterplots: Endpoint para generar gráficos de dispersión. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/distributioncharts: Endpoint para generar gráficos de distribución. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/mapscharts: Endpoint para generar gráficos de mapas. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.
- $Home/ API/tablescharts: Endpoint para generar gráficos de tablas. Que espera dos variables una para escoger que gráfico usar y otro para que tema usar.

Los endpoints de la API esperan dos variables principales:

- type_chart: Esta variable determina el tipo de gráfico que se generará y puede tomar valores numéricos que representan diferentes tipos de gráficos, según el tipo de endpoint al que se accede. Por ejemplo, para el endpoint $Home/-barchart, los valores pueden ser 1 para gráficos de barras, 2 para gráficos de líneas, etc.
- theme_chart: Esta variable especifica el tema que se aplicará al gráfico y espera los nombres de los temas registrados en Vega-Altair. Algunos de los valores posibles incluyen "default", .ªccessible-theme", "dark-accessible-theme", "filler-pattern-theme", entre otros.

En caso de que el endpoint reciba valores incorrectos en estas variables, devolverá un error indicando el tipo de valor que se espera para cada variable. Esto garantiza una interacción clara y ayuda a los usuarios a proporcionar los datos correctos para generar los gráficos deseados.

6.10. Bocetos de la página web

En el diseño de la página web de demostración, se ha tomado inspiración del diseño de visual vocabulary para crear una interfaz que detalle cada tipo de gráfico de manera exhaustiva [23]. Cada sección de la página web está organizada para mostrar diferentes ejemplos de gráficos del mismo tipo. En la página principal, se proporciona una descripción del propósito de la página web y de la librería, junto con enlaces al repositorio de GitHub de la librería y a su documentación..

Además, se incluyeron enlaces a páginas con gráficos que ofrecen descripciones detalladas y opciones para personalizar los gráficos. Esta organización permite a los usuarios explorar y comprender mejor las capacidades de la librería, así como obtener ideas y sugerencias para la visualización de datos en sus propios proyectos.

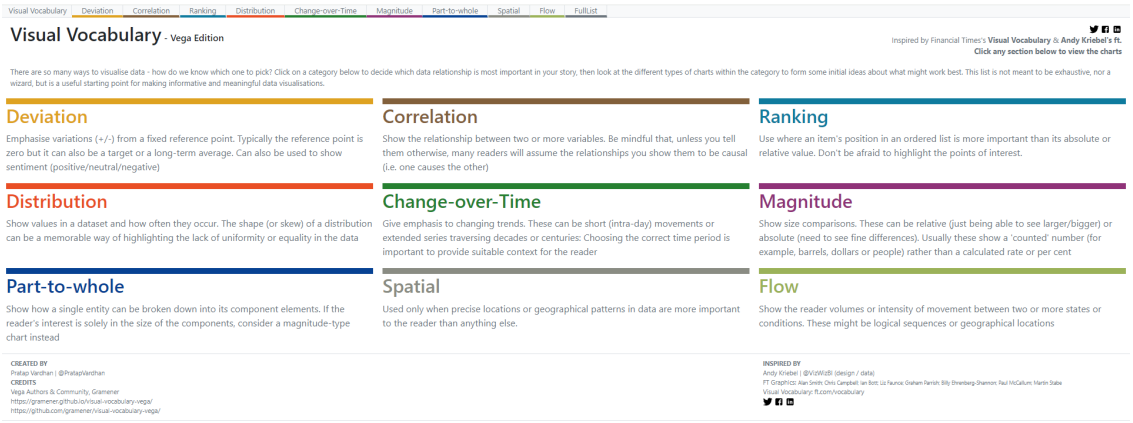

Figura 7: Página web de Visual Vocabulary

6.11. Casos de uso

Los siguientes son los casos de uso de la librería

Descripción	UC1. Creación de Gráficos en Python con Altair-Easeviz
Actores	Desarrollador
Precondiciones	Tener instalado vega-altairVega-Altair 5. * > Tener instalado la librería Altair-Easeviz con pip
Flujo básico	1.El usuario desarrollador carga el tema con altair.themes.enable('accessible_theme''ubaccesible') 2. El desarrollador crea un gráfico usando la librería de Vega-Altair
Flujo alternativo	Si el desarrollador encuentra algún problema durante la configuración, puede consultar la documentación de Altair-Easeviz para obtener orientación adicional.
Postcondiciones	Se generan gráficos interactivos en Python que cumplen con los principios de accesibilidad. El usuario ahora tiene configurado un tema accesible para mejorar la visualización de los gráficos

Descripción	UC2. Integración en una página web
Actores	Desarrollador
Precondiciones	Tener instalado vega-altairVega-Altair 5. * > Haber instalado la librería con pip Tener instalado la librería Altair-Easeviz con pip Tener un objeto tipo de Chart de la librería Vega-Altair
Flujo básico	1. El usuario desarrollador llama a la función create_accessible_scheme get_theme() de la librería del paquete temas Altair-Easeviz, donde paso por parámetros el objeto Chart, un nombre para el archivo HTML y una descripción del gráfico (opcional) 2. La función devuelve la configuración en formato dict de Python
Flujo alternativo	Si el desarrollador no pasa una descripción la función incluirá alternativamente el siguiente texto 'This is a graph made with vega-altair and altair-easeviz'
Postcondiciones	La función crea un archivo HTML con el nombre pasado donde se verá el gráfico con los datos del Objeto Chart, además de contar con opciones para modificar su apariencia y una descripción que se usa como texto alternativo al gráfico además de ser mostrado como texto en el HTML.

Descripción	UC3. Personalización de Temas Visuales
Actores	Desarrollador
Precondiciones	Tener instalado la librería Altair-Easeviz con pip
Flujo básico	1. El desarrollador hace una instancia de alguno de los 4 modelos temas accesibles disponibles AccessibleTheme, DarkAccessibleTheme, FillerPatternTheme, PrintFriendlyTheme 2. El desarrollador ejecuta alguna de las funciones que cambia los atributos de ese modelo
Flujo alternativo	Si el desarrollador mete algún parámetro mal Vega-Altair lanzará una excepción al momento de tratar de actualizar Si el desarrollador mete algún parámetro mal y no lanza una excepción de Vega-Altair, Vega-Lite lanzará una excepción al momento de tratar de visualizar el gráfico en HTML
Postcondiciones	Vega-Altair se actualiza con la nueva configuración cargada

Descripción	UC4. Generación de Descripciones Textuales
Actores	Desarrollador
Precondiciones	Haber instalado la librería con pip. Tener instalado la librería Altair-Easeviz con pip. Tener instalado R y las librerías Braille R, ggplot2, Rserve. Tener corriendo R y ejecutando la librería Rserve.
Flujo básico	1. Llamar a la función generate_description() donde pasamos por parámetro un objeto tipo Chart de la librería Altair, el tipo de gráfico, los datos del eje X y los datos del eje Y. 2. La función hace una conexión con pyRserve a R. 3. La función usa el tipo de gráfico para cargar un código R de acuerdo con el tipo de gráfico y pasa los parámetros del eje X e Y, también pasa parámetros del Chart como los títulos de los ejes y el título del gráfico. 4. En R se crea un gráfico acorde a los parámetros dados y es evaluado por BrailleR que genera una descripción.
Flujo alternativo	Si alguna excepción se genera en el flujo básico este será capturado por un try-catch y la función devolverá un texto con el error bajo la clave 'error'.
Postcondiciones	La función devuelve un texto descriptivo con los datos pasados bajo la clave 'res'.

Los siguientes son los casos de uso de la página web demostración

Descripción	UC1. Explorar Gráficos
Actores	Usuario
Precondiciones	El usuario está en la página de la web.
Flujo básico	1. El usuario navega por la lista de gráficos disponibles. 2. Selecciona un tipo de gráfico. 3. Observa el gráfico visualizado con configuración predeterminada.
Flujo alternativo	Si el usuario desea personalizar el gráfico, va al caso de uso ÜC2. Personalizar Gráfico".
Postcondiciones	El usuario ha explorado un gráfico en la web.

Descripción	UC2. Personalizar Gráfico
Actores	Usuario
Precondiciones	El usuario está visualizando un gráfico
Flujo básico	1. El usuario accede a las opciones de personalización. 2. Ajusta configuraciones como colores, tamaño de fuente y dimensiones del gráfico. 3. Visualiza el gráfico con las nuevas configuraciones aplicadas.
Flujo alternativo	El usuario vuelve a la configuración predeterminada actualizando la página
Postcondiciones	El usuario ha personalizado el gráfico según sus preferencias.

7. Implementación

7.1. Implementación de temas en librería de Vega-Altair

Para implementar los temas que modifican la apariencia de los gráficos en Vega-Altair[21], existen dos opciones disponibles:

- La primera opción común consiste en registrar el tema y luego activarlo mediante el uso de las funciones altair.themes.register(theme_name, theme()) y altair.themes.enable(theme_name), donde theme() debe devolver un diccionario con una estructura similar a la que se encuentra en la clave config de la especificación Vega-Lite.
- La segunda opción, que es la que se ha implementado, implica el uso de los entry points de la librería de plugins incluida en Vega-Altair. Esta opción permite evitar el paso de registrar los temas manualmente y simplifica el proceso a solo activarlos. Se ha optado por esta última implementación debido a que minimiza la cantidad de código que el desarrollador debe escribir. Además, la clase ModelTheme, al ejecutar alguna función para personalizar el tema, también se encarga de registrar automáticamente el tema en Vega-Altair, lo que facilita aún más el proceso de integración

7.2. Creación de temas

Para la creación de temas, la Clase ModelTheme adopta el patrón de diseño Factory. Esta clase es responsable de definir y generar los diferentes temas que modifican la apariencia de los gráficos. Los temas se caracterizan principalmente por sus variables de color de fondo, color de las letras, color de los ejes y marcas.

Siguiendo este enfoque, se desarrollo cuatro temas específicos destinados a mejorar la accesibilidad y la legibilidad de los gráficos:

- AccessibleTheme
- DarkAccessibleTheme
- FillerPatternTheme
- PrintFriendlyTheme

Cada uno de estos temas implementa una serie de ajustes visuales para adaptarse a diferentes necesidades y preferencias de los usuarios. El uso del patrón de diseño Factory permite gestionar de manera eficiente la creación y configuración de estos temas, asegurando una experiencia visual óptima para los usuarios finales.

7.3. Generador de descripciones

Una vez cumplidos los requisitos establecidos para la conexión con R, podemos utilizar las funciones de este entorno sin dificultad. Para la implementación de la función encargada de generar descripciones, se opto por emplear plantillas asociadas a cada tipo de gráfico. De esta manera, la función denominada generate_description espera recibir los siguientes parámetros:

Chart: Un objeto Chart de Vega-Altair que representa el gráfico que se desea describir.

- type_chart: El tipo de gráfico que corresponde al objeto Chart, pudiendo tomar los valores barchart, scatterplot, piechart o linechart.
- axis_x: Define los datos que se representarán en el eje X del gráfico.
- axis_y: Define los datos que se representarán en el eje Y del gráfico.

Posteriormente, la función selecciona una plantilla con código R basada en el valor de la variable type_chart y completa los datos con las variables de los ejes y títulos obtenidos del objeto Chart. Luego, este código se envía a R, que recrea el gráfico correspondiente, el cual será evaluado por BrailleR.

Al finalizar el proceso, la función devuelve un objeto tipo diccionario con las siguientes claves:

- Res: Un string que contiene la descripción generada.
- Error: En caso de que se haya producido alguna excepción durante la ejecución, se proporciona un mensaje explicativo que detalla el error ocurrido.

7.4. HTML con funciones de personalización

La función encargada de generar un HTML con opciones de personalización, denominada create_accessible_scheme, espera recibir los siguientes parámetros:

- Chart: Objeto Chart de Vega-Altair que representa el gráfico a personalizar.
- Filename: Nombre del archivo HTML a generar.
- Description: Descripción que sirve como alternativa al gráfico y descripción detallada del mismo.

Una vez recibidos estos parámetros, la función utiliza Jinja2 para emplear placeholders e insertar variables. Se extrae el JSON y los meta-datos del objeto Chart, como el tipo de gráfico y el título. Si el tipo de gráfico es uno compuesto, como ConcatChart, LayerChart, FacedChart o RepeatedChart, se muestra una advertencia en el HTML generado. Esta advertencia informa al usuario de que el gráfico puede no

ser completamente compatible con todas las opciones de personalización disponibles. Dado que estos gráficos compuestos pueden contener múltiples subgráficos, es difícil determinar con precisión cómo será su especificación de Vega-Lite. Además, en algunos casos, las claves de encoding pueden estar encapsuladas, lo que complica aún más la personalización.

7.5. Implementación de la página web con end points de flask

Gracias a Flask, la página web utiliza los siguientes endpoints, cada uno de los cuales muestra gráficos de diferentes tipos:

- $Home: Este endpoint representa la página principal de la API. Durante el desarrollo, se accede a través de la IP 127.0.0.1:5000, mientras que en producción se accede mediante la URL del sitio https://accesible-theme-altair.onrender.com/.
- $Home/barcharts: Este endpoint renderiza un HTML que muestra gráficos de barras.
- $Home/linecharts: Este endpoint renderiza un HTML que muestra gráficos de lineas.
- $Home/areacharts: Este endpoint renderiza un HTML que muestra gráficos de areas.
- $Home/circularplots: Este endpoint renderiza un HTML que muestra gráficos circulares.
- $Home/scatterplots: Este endpoint renderiza un HTML que muestra gráficos de puntos de dispersion.
- $Home/distributioncharts: Este endpoint renderiza un HTML que muestra gráficos de distribucion.
- $Home/mapscharts: Este endpoint renderiza un HTML que muestra gráficos de mapas.
- $Home/tablescharts: Este endpoint renderiza un HTML que muestra gráficos de tablas.

7.6. Despliegue

Para el proceso de despliegue de la biblioteca, inicialmente se empleó TestPyPI[24] como un entorno de desarrollo para verificar la instalación de la biblioteca en diversos entornos. TestPyPI es una instancia de PyPI (Python Package Index)[10] que sirve como entorno de pruebas para los desarrolladores de software en Python

Una vez superadas las pruebas, se procedió a la publicación de la biblioteca en PyPI, poniéndola a disposición del público en general. En este entorno, se garantiza la integridad del código de la biblioteca. Además, se mantiene una copia alternativa en un repositorio de GitHub propio, lo que permite que cualquier persona pueda contribuir y ampliar la biblioteca.

En lo que respecta a la página web, la gestión del despliegue se realizó de manera completamente automatizada mediante la plataforma Render [11]. Este proceso simplificado únicamente requiere proporcionar el repositorio correspondiente y el comando de ejecución apropiado para que Render se encargue del despliegue sin complicaciones.

8. Pruebas

8.1. Unitest

Para garantizar la calidad y funcionalidad de la librería, se ha implementado un conjunto completo de pruebas unitarias y de integración. Estas pruebas están destinadas a verificar el correcto funcionamiento de cada componente de la librería y la cohesión entre ellos.

En primer lugar, se han diseñado pruebas unitarias para cada estructura de datos y función principal de la librería. Estas pruebas se enfocan en confirmar que cada estructura se cree correctamente y que las funciones produzcan los resultados esperados. Se han desarrollado casos de prueba exhaustivos que abarcan una amplia gama de escenarios para asegurar la solidez de la implementación.

Además, dado el enfoque en proporcionar gráficos accesibles, se han integrado pruebas de accesibilidad en las pruebas de integración. Mediante herramientas como Selenium[25], se ha automatizado el proceso de generación de gráficos y se ha verificado que estos sean creados de manera adecuada.

8.2. Herramientas para poner a prueba la accesibilidad

En el proceso de evaluación de la accesibilidad de los gráficos, se han empleado diversas herramientas destinadas a simular las experiencias de usuarios con discapacidades visuales o motoras. Esta evaluación exhaustiva se llevó a cabo utilizando una combinación de plugins del navegador web Chrome y programas externos especializados, que incluyen:

- **Wave**: Este plugin específico para Chrome ha sido instrumental para identificar posibles problemas de accesibilidad en la página web, incluidos los gráficos. Proporciona sugerencias y recomendaciones para mejorar la experiencia del usuario, garantizando la conformidad con las pautas de accesibilidad.
- **Colorblindly**: Mediante este plugin, se han simulado distintos tipos de daltonismo, permitiendo evaluar cómo los gráficos son percibidos por personas con diferentes tipos de deficiencia en la percepción del color. Esto ha posibilitado ajustes en los esquemas de color para asegurar la legibilidad y comprensión adecuadas para todos los usuarios.
- **Funkify**: Esta herramienta ha sido de gran utilidad al simular diversas discapacidades y condiciones, como la dislexia o la parálisis cerebral. Esta simulación nos ha permitido comprender mejor cómo interactúan los usuarios con el sitio web y sus gráficos en diferentes contextos de discapacidad, facilitando así la optimización de la accesibilidad.
- **NVDA (NonVisual Desktop Access)**: Como software de código abierto, NVDA ofrece funcionalidades de lectura de pantalla para personas ciegas o con

discapacidad visual. Su utilización nos ha permitido evaluar la accesibilidad de los gráficos y asegurar su comprensibilidad y navegabilidad para todos los usuarios, independientemente de sus capacidades visuales.

- **VoiceOver**: Esta función integrada en los dispositivos de Apple proporciona lectura de pantalla para usuarios con discapacidad visual. Al verificar la accesibilidad de los gráficos en dispositivos iOS mediante VoiceOver, se ha garantizado una experiencia consistente en diferentes plataformas, mejorando así la accesibilidad general del proyecto.

9. Resultados

Los resultados se presentan visualmente a través de imágenes que comparan gráficos generados con la librería base de Vega-Altair y la librería mejorada con Altair-Easeviz.

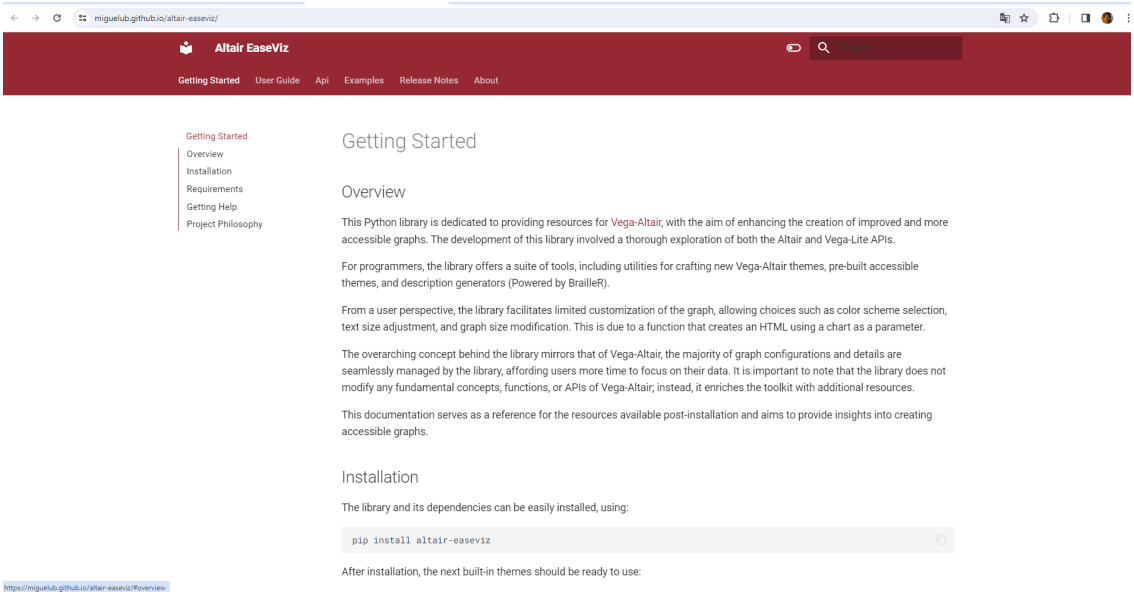

Figura 8: Documentación web de la librería Altair-Easeviz

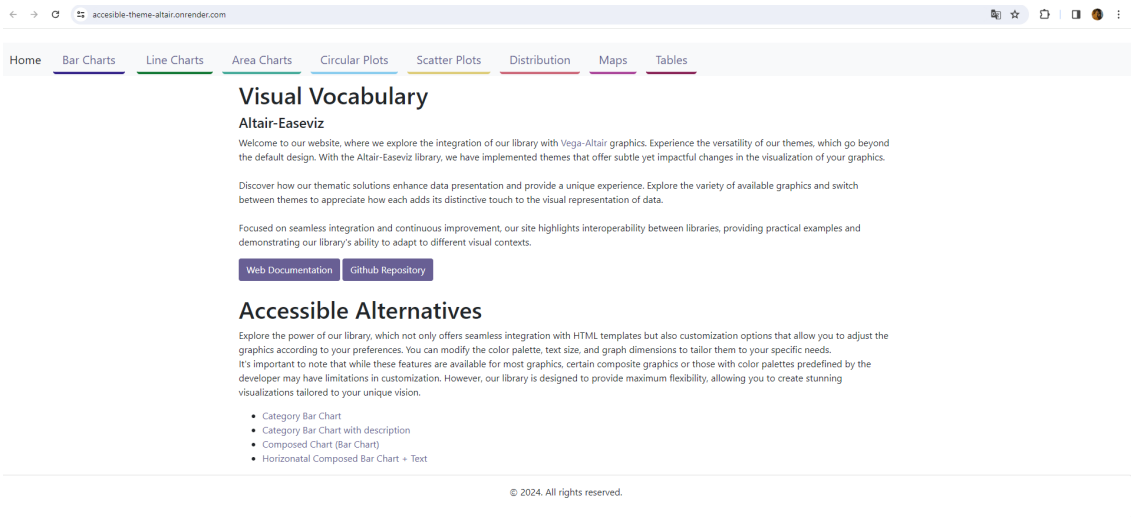

Figura 9: Página web con diferentes gráficos donde aplicamos la librería desarrollada

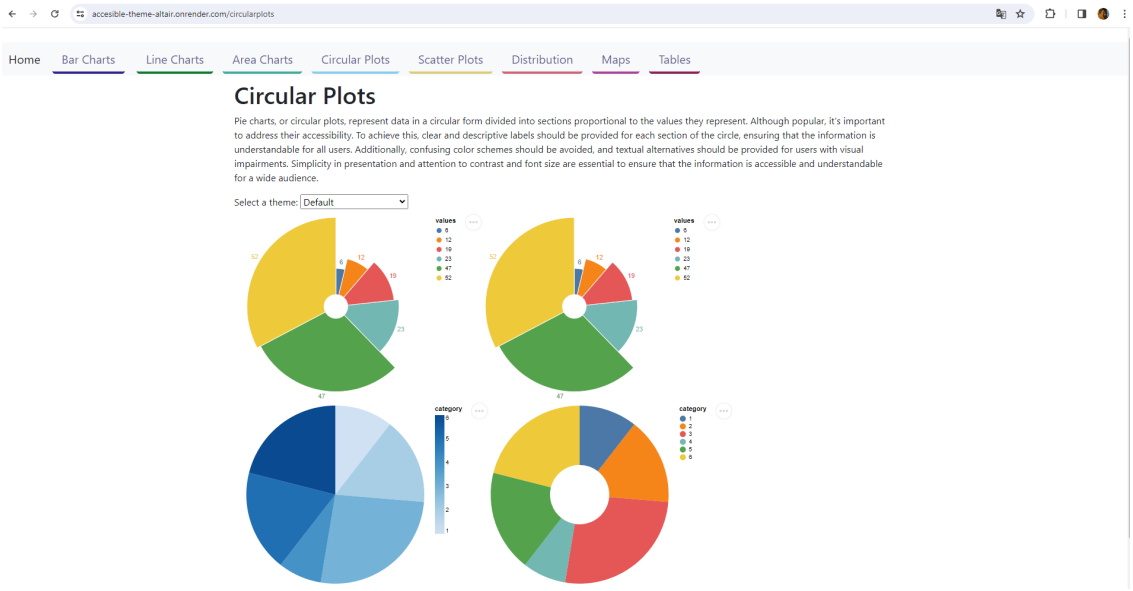

Figura 10: Gráficos circulares expuestos en la pagina web de demostración

Figura 11: Comparación entre el tema por defecto de Vega-Altair y los temas implementados por Altair-Easeviz. La primera imagen representa el aspecto predeterminado del gráfico sin la intervención de esta librería. La segunda imagen presenta el mismo gráfico con un tema más accesible, mejorando la legibilidad y la diferenciación de datos. La tercera imagen utiliza un tema oscuro para adaptarse a preferencias visuales específicas. La última imagen utiliza una paleta de colores de patrón, agregando variación visual y opciones de personalización a la representación gráfica

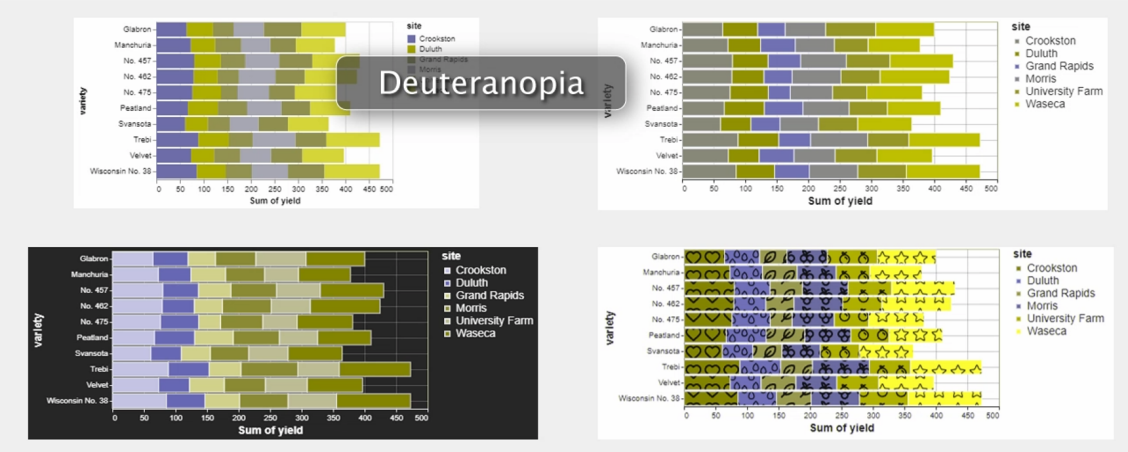

Figura 12: Representación de Figura 11 con un filtro de daltonismo de deuteranopia

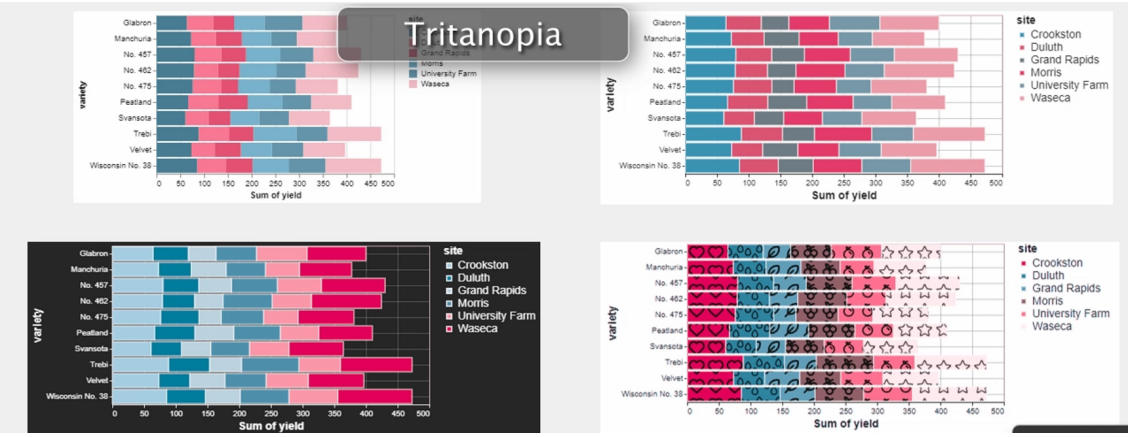

Figura 13: Representación de Figura 11 con un filtro de daltonismo de tritanopia

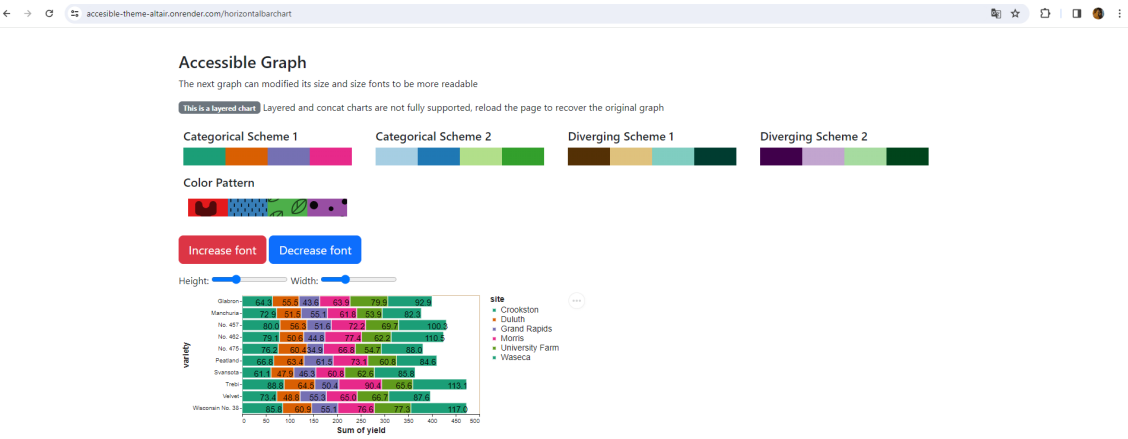

Figura 14: En esta figura, se presenta un ejemplo de un gráfico de barras que destaca las múltiples opciones de personalización que Altair-Easeviz ofrece a los usuarios.

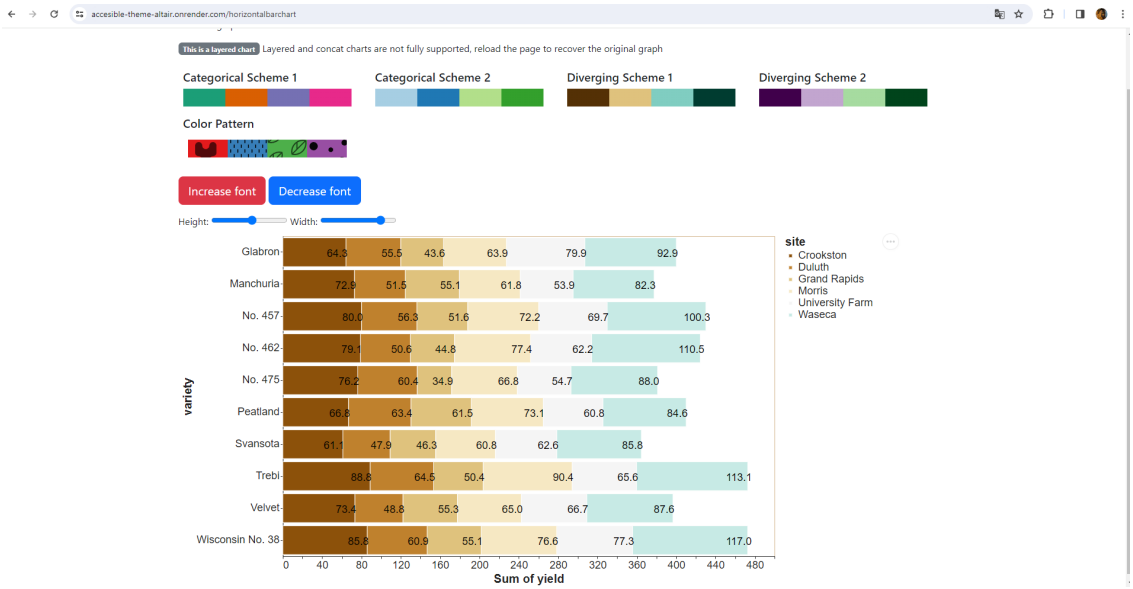

Figura 15: En esta figura, se presenta un ejemplo de un gráfico de dispersión que destaca las múltiples opciones de personalización que Altair-Easeviz ofrece a los usuarios. Donde se aplicaron cambios de tamaño y color

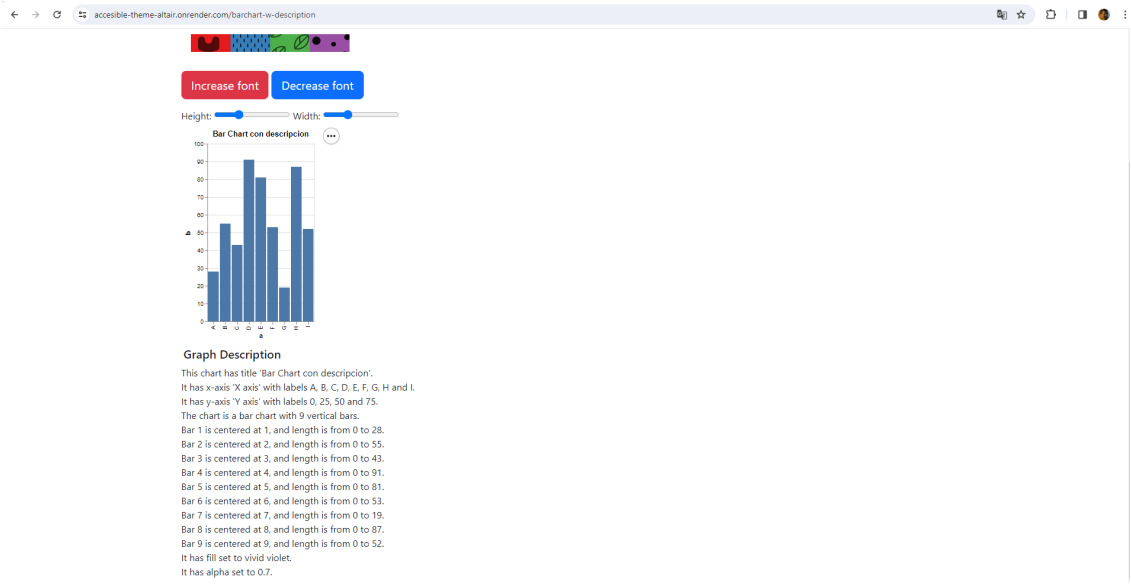

Figura 16: En esta figura, se presenta un gráfico de barras acompañado de su descripción textual generada mediante la integración con la herramienta BrailleR. La inclusión de descripciones textuales mejora significativamente la accesibilidad del gráfico

10. Conclusión y trabajo futuro

En conclusión, se han alcanzado con éxito los objetivos establecidos para este proyecto. La creación de la librería Altair-Easeviz y su documentación, en colaboración con Vega-Lite y Vega-Altair, ha mejorado significativamente la accesibilidad de los gráficos generados, proporcionando descripciones táctiles a través de la integración de BrailleR. La implementación eficiente en PyPI y GitHub garantiza su disponibilidad para la comunidad, y la página web accesible demuestra su aplicación en varios tipos de gráficos.

En las perspectivas futuras, tenemos la ambición de profundizar en la investigación de SVG y explorar su potencial para personalizar aún más los gráficos, buscando formas innovadoras de mejorar la accesibilidad y funcionalidades interactivas. Además, planeamos investigar alternativas a BrailleR, considerando posibles integraciones de inteligencia artificial, como ChatGPT, para ofrecer descripciones más adaptativas y multilingües. Estamos comprometidos con la continua evolución de Altair-Easeviz, buscando siempre nuevas oportunidades para hacer que la visualización de datos sea más inclusiva y accesible para una audiencia global.

Referencias

[1] EUR-Lex. Directiva (UE) 2019/882 del Parlamento Europeo y del Consejo, de 17 de abril de 2019, sobre los requisitos de accesibilidad de los productos y servicios. 2019. https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32019L0882&from=EN

[2] W3C. Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/

[3] Highcharts. Accessibility module. https://www.highcharts.com/docs/accessibility/accessibility-module

[4] Plotly. Getting Started with Plotly in Python. https://plotly.com/python/getting-started/

[5] Rubén Alcaraz-Martínez; Mireia Ribera; Adrià Adeva Fillol; Afra Pascual Almenara. 2022. Validation of a heuristic set to evaluate the accessibility of statistical charts. In Proceedings of the XXII International Conference on Human Computer Interaction (Interacción '22). Association for Computing Machinery, New York, NY, USA, Article 2, 1–9. https://doi-org.sire.ub.edu/10.1145/3549865.3549893

[6] Connor Geddes, David R. Flatla, Garreth W. Tigwell, and Roshan L Peiris. 2022. Improving Colour Patterns to Assist People with Colour Vision Deficiency. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22). Association for Computing Machinery, New York, NY, USA, Article 479, 1–17. https://doi-org.sire.ub.edu/10.1145/3491102.3502024

[7] W3C. Designing for Web Accessibility. https://www.w3.org/WAI/tips/designing/

[8] PyPA - Python Packaging User Guide. Creating and discovering plugins. 2020. https://packaging.python.org/en/latest/guides/creating-and-discovering-plugins/#using-package-metadata

[9] Matthew Herbst and Bo Brinkman. 2014. Color-via-pattern: distinguishing colors of confusion without affecting perceived brightness. In Proceedings of the 16th international ACM SIGACCESS conference on Computers accessibility (ASSETS '14). Association for Computing Machinery, New York, NY, USA, 245–246. https://doi-org.sire.ub.edu/10.1145/2661334.2661383

[10] Pypi. El Índice de paquetes de Python (PyPI) es un repositorio de software para el lenguaje de programación Python. https://pypi.org/

[11] Render. Build, deploy, and scale your apps with unparalleled ease. https://render.com

[12] Chartability. The Chartability Workbook. https://chartability.github.io/POUR-CAF/#howdoyouusechartability

[13] Organización Mundial de la Salud(OMS). La OMS presenta el primer Informe mundial sobre la visión. https://www.who.int/es/news/item/08-10-2019-who-launches-first-world-report-on-vision

[14] National Institute of Eye. Facts About Color Blindness. https://web.archive.org/web/20160728003639/https:/nei.nih.gov/health/color_blindness/facts_about

[15] VanderPlas et al., (2018). Altair: Interactive Statistical Visualizations for Python. Journal of Open Source Software, 3(32), 1057. https://doi.org/10.21105/joss.01057

[16] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. IEEE Transactions on Visualization and Computer Graphics 23, 1 (January 2017), 341–350. https://doi.org/10.1109/TVCG.2016.2599030

[17] A. Jonathan R. Godfrey. Package 'BrailleR'. https://cran.r-project.org/web/packages/BrailleR/BrailleR.pdf

[18] Ralph Heinkel. pyRserve. https://pypi.org/project/pyRserve/

[19] MDN. Accessibility concerns. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas#accessibility_concerns

[20] Vega-Lite. Top-Level Specifications. https://vega.github.io/vega-lite/docs/spec.html#top-level

[21] Vega-Altair.Vega-Altair - Getting started. https://altair-viz.github.io/getting_started/overview.html

[22] Python. Typing. Support for type hints. https://docs.python.org/3/library/typing.html

[23] Pratap Vardhan.Visual Vocabulary. https://gramener.github.io/visual-vocabulary-vega/

[24] Pypy. Entorno de pruebas del Índice de paquetes de Pythonhttps://test.pypi.org/

[25] Selenium. Selenium automates browser. https://www.selenium.dev/

[26] Miguel Víctor Huayllas Choque. Librería accesible Altair-Easeviz. https://github.com/MiguelUB/altair-easeviz

[27] Miguel Víctor Huayllas Choque. Documentación Altair-Easeviz. https://miguelub.github.io/altair-easeviz/

[28] Miguel Víctor Huayllas Choque. Visual Vocabulary Altair-Easeviz. https://accesible-theme-altair.onrender.com/

ChatGPT (GPT-3, modelo de generación de lenguaje a gran escala de OpenAI) se ha utilizado para mejorar el estilo de escritura de este artículo. El autor reviso, edito y modifico los textos generados por ChatGPT según sus preferencias y asumen la responsabilidad final del contenido de esta publicación.