UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Dashboard de visualizacion del uso y preferencias de los desarrolladores basado en las encuestas anuales de Stack Overflow

Presentado por Alba Bartolome Román
en Universidad de Burgos — 11 de junio de 2022
Tutores: José Ignacio Santos Martín
Virginia Ahedo García

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Gº en Ingeniería en Informática

D. José Ignacio Santos Martín, y Dña. Virginia Ahedo García, profesores del departamento de Ingeniería de Organización, área de Organización de Empresas

Exponen:

Que el alumno Dña. Alba Bartolomé Román, con DNI 12425540C, ha realizado el Trabajo final del Gº Ing. Informática titulado: Dashboard de visualización del uso y preferencias de los desarrolladores basado en las encuestas anuales de Stack Overflow

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a 9 de junio de 2022

Vº. Bº. del Tutor

Vº. Bº. del Tutor

D. José Ignacio Santos Martín

Dña. Virginia Ahedo García

Resumen

Stack Overflow realiza todos los años una encuesta a desarrolladores que permite conocer detalles sobre las diferentes tecnologías con las que trabajan los desarrolladores y sus preferencias. Los datos de esta encuesta son públicos y se encuentran disponibles bajo licencia OdbL.

El objetivo de este proyecto consiste en desarrollar una aplicación web que permita mostrar loa principales datos de esta encuesta de forma simple. Para ello se ha procesado los datos de la encuesta de Stack Overflow, se ha estudiado y seleccionado un conjunto de gráficos y se ha diseñado una página HTML que alberga un *dashboard* interactivo.

Descriptores.

Dashboard, aplicación web, Python, Dash, Plotly, gráficos interactivos, procesado de datos, dataframe.

Abstract

Every year, Stack Overflow conducts a developer survey that provides details about the different technologies developers work with and their preferences. The data from this survey is public and is available under the OdbL license.

The aim of this project is to develop a web application that allows the main data from this survey to be displayed in a simple way. For this purpose, the data from the Stack Overflow survey was processed, a set of graphs was studied and selected, and an HTML page was designed to host an interactive dashboard.

Keywords.

Dashboard, web application, Python, Dash, Plotly, interactive graphics, data processing, dataframe

Índice general

Índice de figuras

1. Introducción

Actualmente se está viviendo en una época muy centrada en el consumo de información, social, económica, política, cultural, etc. La vida ajetreada del público y la ingente cantidad de datos que inunda todos los días nuestras pantallas hace que, aquellos no procesados y mostrados a los usuarios de forma asequible y simple, sean simplemente olvidados.

Stack Overflow —la comunidad por excelencia que define a una gran mayoría de personas que están desarrollando código— está formada por un grupo de usuarios que interactúan a menudo entre ellos. Aunque la propia página de Stack Overflow publica un resumen cada año, hablando desde experiencia personal, apenas se ha prestado atención.

Este trabajo intenta mejorar este último aspecto más abandonado por parte de Stack Overflow para hacerlo más interactivo e interesante a nivel personal.

1.1 Estructura de la memoria.

La memoria se ha organizado de la siguiente forma:

- **Introducción:** breve descripción del contenido del trabajo y la estructura de los elementos enviados.
- **Objetivos del proyecto**. Definición concisa de las metas que se propone alcanzar con el proyecto. Se hace una separación por categorías: objetivos generales, técnicos y personales.
- **Conceptos teóricos**. Apartados teóricos del trabajo que, por su complexidad, necesiten ser explicados de forma más amplia para poder ser comprendidos correctamente.
- **Técnicas y herramientas**. Profundizar en las técnicas y herramientas usadas en el proyecto además de explicar las alternativas estudiadas que finalmente fueron rechazadas.
- **Aspectos relevantes**. Partes del proyecto o puntos del desarrollo que son lo suficientemente interesantes como para mencionarlos.
- **Trabajos relacionados**. Breve mención a trabajos ya realizados que han inspirado el proyecto.
- **Conclusiones.** Conclusiones y líneas de mejora del proyecto.

1.2. Anexos.

A continuación, se mencionará brevemente la estructura de los anexos:

- **Plan de proyecto software**: Incluye una planificación temporal y un estudio de viabilidad económica y legal.
- **Especificación de requisitos**: Objetivos generales, catálogo de requisitos y especificación de requisitos.
- **Especificación de diseño**: Se menciona el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Documentación técnica de programación**: Se trata la estructura de directorios, el manual del programador, la compilación, instalación y ejecución del proyecto, así como las pruebas del sistema.
- **Documentación de usuario**: Se cubren los requisitos de usuarios, la instalación y el manual del usuario.

1.3. Enlaces entregados.

Además de la memoria y el anexo, también se entregará la dirección web del despliegue de la aplicación en Heroku, un link donde se pueda ver el proceso de calidad de desarrollo del código y la dirección del repositorio.

- **Heroku (despliegue de la aplicación)**
  - https://tfg-dashboard.herokuapp.com/
- **Codacy (control de calidad).**
  - https://app.codacy.com/gh/albarrom/GII_O_MA_21.05/dashboard
- **Repositorio en GitHub.**
  - https://github.com/albarrom/GII_O_MA_21.05

2. Objetivos del proyecto

Los objetivos generales han sido marcados por los requisitos funcionales. Estos serán:

- Desarrollar un *dashboard* que muestre los datos recogidos de una encuesta en forma de gráficos.
- Desarrollar los componentes necesarios para mantener y gestionar el *dashboard*.
- Procesar gran cantidad de información y mostrarla al usuario de forma accesible.
- Permitir al usuario interactuar con los datos.

2.2 Objetivos técnicos.

- Estudiar extensivamente Python y los paquetes escogidos para el desarrollo del *dashboard*.
- Usar GitHub como sistema de control de versiones.
- Usar herramientas de software libre.
- Usar ZenHub y GitHub para implementar metodologías ágil.
- Hacer uso de Jupyter Notebooks y PyCharm para desarrollar el código.
- Usar la arquitectura Modelo-Vista-Controlador.
- Implementar una página web funcional.

2.3 Objetivos personales.

- Aprender metodología y conceptos de desarrollo web (HTML, CSS y Bootstrap).
- Poner en práctica conceptos aprendidos en la carrera como, por ejemplo:
  - Metodologías ágiles.
  - Herramientas de control de versiones.
  - Desarrollo de programas (arquitectura modular, modelo-vista-controlador, especificación de requisitos, etc.).
- Aprender a desarrollar de una aplicación para un servidor y realizar su posterior despliegue.

3. Conceptos teóricos

En este capítulo se van a tratar de forma resumida todos los conceptos teóricos que actuarán a modo de base para comprender el proyecto.

3.1 Plotly

Plotly Python es una librería que genera gráficos interactivos en HTML. Es interesante mencionar brevemente dos de sus módulos:

- Plotly Express (usualmente importada como px): API de alto nivel para visualizar datos.
- Plotly Graph Objects (usualmente importada como go): API de bajo nivel para figuras, trazos y diseño.

```
fig = px.treemap(df, path=['Age1stCode', nameColumns[orden]],values='count',)
fig.update_layout(title_text="Age of start developing career and top 5 of most used "+name[orden], fo
```

```
fig = go.Figure(data=[go.Sankey(node = dict(pad = 15, thickness = 20,
                                            line = dict(color = "black", width = 0.5),
                                            label = labels,
                                           ),
                                link = dict(
                                    source = source,
                                    target = target,
                                    value = df['count']
                                      ))])
fig.update_layout(title_text="Education level and top 5 of most used "+name[orden], font_size=12)
```

Ilustración 1 Ploty Express vs Plotly Graph Objects

Plotly Express tiene una sintaxis significativamente más sencilla, pero sus gráficos son menos llamativos visualmente.

3.2 Dash.

Dash, tal y como es definido en su documentación[1], es un marco para crear rápidamente aplicaciones de datos en Python. Está diseñado para ser usado con Plotly Python. Tanto es así que a veces es denominado Plotly Dash. Es ideal para crear e implementar aplicaciones de datos con interfaces de usuario personalizadas. Es particularmente adecuado para cualquiera que trabaje con datos y no sea muy experimentado en HTML.

A continuación, se describirán los componentes más importantes que se han usado para construir la aplicación:

3.2.1 Dash HTML y conceptos básicos en HTML.

Aunque no se ha programado directamente en web usando exclusivamente HTML o CSS, sí se han necesitado estudiar algunos conceptos básicos. Dado que la programación web no ha sido uno de los apartados más importantes en el proyecto, tan solo se va a mencionar muy brevemente las características comunes de HTML con el módulo Dash HTML Components – módulo en el que se ha apoyado para construir el *layout* de la página.

La mayoría de elementos HTML usados en este *dashboard* constan de tres partes muy diferenciadas: una etiqueta de apertura, un contenido y una etiqueta de cierre. En Dash se consigue el mismo efecto mediante el uso de corchetes y paréntesis.

```
<div>
  <h1>This is the heading</h1>
  <div>
      <p>Here is a paragraph.</p>
  </div>
</div>
```

```
html.Div([
    html.H1('This is the heading'),
    html.Div([
        html.P('Here is a paragraph')
    ])
])
```

Ilustración 2 Comparación HTML y Dash: conceptos básicos

La estructura al añadir propiedades es similar al usar componentes HTML y Dash, pero hay algunas diferencias clave[2]:

- La propiedad *style* es un diccionario.
- *class* es renombrada como *className*.
- Se pueden omitir los píxeles en las propiedades de *style*.

Dash tiene un componente capaz de imitar a cada uno de HTML. Por ejemplo, el componente Graph ha sido usado en la aplicación para renderizar todos los gráficos generados por Plotly.

Se puede ampliar esta información y ver una lista que profundice en detalle sobre cada uno de ellos en la documentación de Dash.

3.2.1 Dash Core Components.

Componentes que permiten crear interacción con la página. La documentación[3] puede nombrar varios ejemplos, pero en la aplicación se ha usado especialmente el componente *dropdown*.

```
dcc.Dropdown(id="opt1",  multi=False, value='2021',
          options=[
              {"label": "2021", "value": 2021},
              {"label": "2020", "value": 2020}],
              style={'width': '45vh'},
             ),
```

Ilustración 3 Dash Core Components: elemento dropdown

Para crear esa interacción, se usan las funciones *callback*. Se profundizará en ellas en el apartado 3.2.1 de este documento.

3.2.3. Dash Bootstrap Components.

Dash Bootstrap es una biblioteca de componentes de Bootstrap que facilita la creación de aplicaciones con diseños responsive[4]. Estos componentes se han usado especialmente para diseñar el *layout* del *dashboard*, pero no se ha profundizado demasiado en ellos.

3.2. Dash y el modelo - vista – controlador.

Dash, tal y como lo define su propia documentación, es un *framework* escrito sobre Plotly y React y es ideal para crear e implementar aplicaciones con interfaces personalizadas.

Todas las aplicaciones creadas con Dash, incluyendo la creada en este proyecto, siguen una estructura de tres partes consecutivas: manipulación de datos, creación del *layout* e interacción entre los componentes. Esta particular estructura puede asociarse fácilmente con el modelo – vista – controlador[5] de la siguiente forma:

Modelo, acorde con el modelo – vista – controlador (MVC), únicamente representa los datos. Si se tiene en cuenta la estructura de tres partes mencionada anteriormente se puede llegar a la conclusión que su equivalente en Dash sería la **manipulación de datos**.

```
def branchGraph (df):

    #normalizar todos los datos.
    df.loc[df["MainBranch"] == "None of these", "MainBranch"] = "Other"

    df= df.groupby(['MainBranch',],as_index=False).size()

    return df
```

Ilustración 4 Ejemplo manipulación datos/modelo.

Vista permite que el usuario pueda visualizar los datos del modelo. Se incluirá únicamente el aspecto visual de todos los botones, gráficos y demás elementos gráficos. En Dash corresponderá con la **creación del *layout***.

En el proyecto se han usado dos componentes de la librería Dash llamados Dash Core Components y Dash Bootstrap Components. Estos se verán con más profundidad en el apartado siguiente.

Controlador crea todas las interacciones entre los elementos del *layout*. En Dash y en la aplicación, se asociaría con la **interacción de los componentes**. Es decir: con la creación de las funciones *callback*.

3.2.1 Funciones callback.

Estas funciones son llamadas de forma automática cada vez que se cambia la propiedad de un elemento de entrada para actualizar alguna propiedad en un componente de salida.[6]

En el apartado anterior se mencionó como, en el *layout*, el componente Graph tomaba como input una figura generada con Plotly. Las funciones *callback* funcionan de forma similar.

Todas las funciones *callback* necesitan un output y un input. Estos a su vez necesitan un component_id y un component_property. Cada uno de estos componentes en el *callback* referenciará un componente en el *layout*.

Se explicará este concepto sobre dos fragmentos de código de la aplicación, el primero perteneciendo al *layout* y el segundo a su propia función *callback*:

```
dbc.Row([ #dropdown
    dbc.Col([
        dcc.Dropdown(id="opt1",  multi=False, value='2021',
                     options=[
                         {"label": "2021", "value": 2021},
                         {"label": "2020", "value": 2020}],
                         style={'width': '45vh'},
                        ),
    ])
]), #dropdown
dcc.Loading([ #spinner
dcc.Graph(id='primero', figure={}),
]), #spinner
```

```
@app.callback(
    Output(component_id='primero', component_property='figure'),
    Input(component_id='opt1', component_property='value'))
def update_graph(opt1):
```

Ilustración 5 Funcionamiento de callback en fragmento de código

El output de la función *callback* que se ve en la imagen es el gráfico generado en Plotly y el input es el valor del *dropdown*. El usuario puede interactuar con dicho valor y hacerlo cambiar, cambiando así el input del *callback* y actualizando el output en consecuencia.

4. Técnicas y herramientas

Este apartado de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.

4.1 Metodología

4.1.1 Scrum.

Scrum es un marco de desarrollo para crear software funcional en una estructura cíclica hasta finalizar el proyecto:

- Se selecciona uno o varios problemas que deban ser tratados y se dividen en tareas más sencillas que puedan ser realizadas en un marco temporal muy concreto, también denominados sprint.
- El equipo se reúne al finalizar un sprint para llevar un seguimiento de las tareas realizadas, ajustar expectativas y programar el siguiente *sprint*.

El principal objetivo de los ciclos es abordar problemas complejos, pudiendo tener así un mayor control sobre la versión final, teniendo la oportunidad de adaptar o cambiar el desarrollo de forma continua.

4.1.2 Kanban.

Es una metodología de desarrollo basado en observar de forma dinámica los cambios en el proyecto. *Kanban* es una palabra de origen japonés que significa “letrero”, término que resume muy bien su filosofía.

Las tareas se colocan en una pizarra o panel siguiendo 3 criterios: pendientes, en progreso o finalizadas. Los miembros del equipo son asignados diferentes tareas y es su responsabilidad cambiar la tarjeta de lugar conforme al estado de la tarea en particular.

En el proyecto se ha intentado seguir las mismas directrices, añadiendo una columna de revisión para aquellas *issues* que necesitaban pasar por un proceso previo antes de ser cerradas.

4.1.3 Técnica *pomodoro*.

Durante todo el desarrollo del proyecto se ha usado una interfaz web[7] para seguir la técnica *pomodoro*. Su objetivo es mejorar la concentración y tener una administración del tiempo óptima. Para ello divide la carga de trabajo en periodos de 25 minutos de trabajo intenso - denominados *pomodoros* - seguidos por 5 minutos de descanso. Tras 4 *pomodoros* el tiempo de descanso será de 30 minutos.

4.2. Herramientas

4.2.1 Git- GitHub.

GitHub es un software de gestión de versiones. Cuenta con una aplicación de escritorio (GitHub Desktop) que simplifica considerablemente la gestión del código. En este proyecto se ha usado como método principal de gestión de repositorio.

Haber tenido contacto previo con esta herramienta en el grado, su popularidad y la facilidad para encontrar documentación online hizo que no se consideran otras alternativas.

4.2.2 Zenhub.

Zenhub es un plugin de GitHub. Ayuda a gestionar el proyecto aplicando metodologías ágiles.

4.2.3 Python

Lenguaje de alto nivel. Se eligió Python por su versatilidad a la hora de tratar con grandes conjuntos de datos.

4.2.4 Jupyter Notebooks.

Jupyter es un IDE (Integrated Development Environment) basado en web que permite organizar el código en notebooks. Soporta Python y su característica más importante es su diseño basado en celdas que permite outputs de código, HTML, gráficos, imágenes o incluso LaTeX. El diseño modular hace posible crear flujos de trabajo con una alta personalización.

La flexibilidad que otorga Jupyter Notebooks y el hecho de tener amplia experiencia derivada de otras asignaturas en el grado fue lo que lo hizo destacar frente a otras alternativas como por ejemplo Google Colab. Este último está basado en un concepto muy similar al de Jupyter Notebooks.

Se descartó porque Jupyter tiene acceso local directo en lugar de almacenar el proyecto en Google Drive y tiene el beneficio de usar el hardware del equipo. Dado que esto va a ser un proyecto desarrollado por una sola persona, se prefirió Jupyter.

4.2.5 PyCharm.

IDE usado especialmente para Python. Requiere una licencia, pero usando la cuenta de la UBU se puede solicitar una licencia de estudiante[8]. Se incorporó en las fases más tardías del desarrollo para suplir las carencias de Jupyter Notebooks.

Se valoraron otras opciones como Visual Studio Code o, incluso, Sublime Text 3 que no es un IDE sino un simple editor de código. Sin embargo, PyCharm destacó por encima de cualquier otra opción por su compatibilidad con Jupyter Notebooks.

4.3 Librerías

4.3.1 Plotly

Librería gratuita y de código abierto caracterizada por su accesibilidad. Se ha hecho uso de multitud de herramientas que permiten visualizar grandes conjuntos de datos de forma visualmente atractiva para crear gráficos de forma sencilla.

4.3.2 Dash

Dash es un *framework opensource* de Python.

Una de las grandes ventajas de Dash es que se puede renderizar directamente en el navegador. Su aplicación original es crear aplicaciones web, en este proyecto se ha usado Bootstrap para poder generar el HTML.

Se barajó la posibilidad de usar Panel o Streamlit que fueron descartadas, principalmente porque ofrecían mucha menos personalización y documentación disponible que Dash.

4.3.3. Dash Bootstrap Components

Librería de componentes Boostrap para Dash que facilita enormemente la construcción de la página web.

4.4 Despliegue.

4.4.1 Heroku

Heroku permite hacer *deploy* de la web en la nube. A pesar de tener una versión gratuita, está muy limitada.

Su mayor ventaja frente a otras plataformas en la nube es su extensa documentación, simplicidad y logs de error. En etapas tempranas de desarrollo, especialmente en el primer *deploy*, estos mensajes ayudaron enormemente a encontrar errores.

Otras alternativas que fueron brevemente consideradas como Elastic Beanstalk (AWS) o Dokku antes de ser descartadas por la gran curva de aprendizaje que presentaban respecto a Heroku.

4.5 Documentación.

4.5.1 Zotero.

Gestor de referencias bibliográficas gratuito y de código abierto. Con un solo *click* permite almacenar cualquier referencia digital para su posterior uso usando un estilo de citado.

La versión de escritorio y *pluggins* son compatibles con la mayoría de sistemas operativos –Linux, Windows y MacOS– y navegadores web –Chrome, Firefox y Safari.

Mendeley es una alternativa muy similar a Zootero que se terminó descartando por su pobre importación de metadatos de videos de YouTube.

4.5.2. Microsoft Word.

Popular procesador de texto. LaTeX fue considerado como principal alternativa, pero se eligió Word por su rapidez de trabajo y sencillez.

5. Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto.

5.1. Formación.

Las diferentes asignaturas del grado han dado las herramientas necesarias para aprender a desarrollar software. También se ha hecho un especial énfasis en adquirir una capacidad lógica para enfrentarnos a los problemas que se encuentren y poder resolverlos con relativa facilidad.

La etapa formativa ha sido la más extensa de todo el proyecto. Ha durado desde su elección hasta, prácticamente, la entrega de este. Se ha tratado este proyecto como una herramienta más de aprendizaje ya que casi todos los recursos usados en esta aplicación requerían unos conocimientos previos que no se tenían en un inicio. El único aspecto donde se contaba con experiencia era en Python y el tratamiento de datos.

La documentación oficial de Dash y Plotly es demasiado breve y poco explicativa para una primera toma de contacto. Como punto a favor, la mayoría de elementos y funciones en estas librerías son muy similares. Es decir, una vez entendido el concepto y funcionalidad de sus elementos, como por ejemplo una función *callback*, es fácil adaptarlos a las nuevas necesidades.

Para suplir las carencias que se pudieron encontrar al inicio del proyecto, se consultaron numerosos recursos online. Se debe hacer una mención especial a los tutoriales de CharmingData en Youtube[9] y al propio foro de Plotly[10].

5.2 Problemas.

Durante la realización del proyecto se han encontrado dos problemas de forma persistente. El primero de ellos ha sido causado por falta de memoria. Cuando la aplicación tan solo contenía 4 o 5 gráficos no era demasiado notorio, pero tan pronto el *dashboard* comenzó a crecer rápidamente se convirtió en un serio inconveniente. Llegando al extremo donde la página comenzó a tener errores de carga.

Dash ofrece un sistema *debug* integrado con una interfaz muy accesible que permite identificar fallos rápidamente, pero en este caso en particular no fue demasiado específico ya que el problema parecía solucionarse simplemente recargando la página.

Ilustración 6 Error dashboard.

Además de una herramienta para ver errores, el *debug* de Dash tiene otras funcionalidades para monitorizar el *dashboard*. Una de ellas es el denominado Callback Graph.

La siguiente imagen fue tomada varios días después del error de la ilustración anterior:

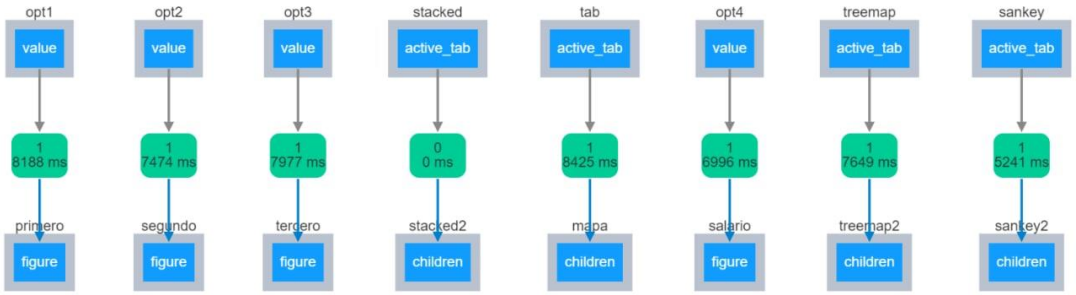

Ilustración 7 Error en callback graph

Esto permitió ver un posible origen del problema. Días más tarde, al tratar de volver a desplegar la app en Heroku para registrar los cambios realizados, se vio un amplio listado de errores R14 y un fallo en el *deploy*. El Dev Center de Heroku indica que, si se obtiene una gran cantidad de estos errores, es probable que el rendimiento de la aplicación se degrade gravemente[11].

Una vez identificado el problema, se trató de llevar a cabo varias acciones para solucionarlo:

Primero se construyó una función que pudiese reconocer cada columna del *dataframe* y convertir sus datos al tipo más apropiado. Esto probó resultar muy poco efectivo, por lo que se ha eliminado del producto final.

Se modificó la forma en la que los *dataframes* de la encuesta de Stack Overflow eran procesados. Tan solo se seleccionaron las columnas con datos relevantes para el

dashboard y se añadió el parámetro "*engine*". Este parámetro indica que *parser engine* se debe usar al leer el archivo csv y convertirlos a dataframe con pd.read_csv. Acorde a la documentación[12], el más rápido es C y pyarrow. Se eligió el primero ya que pyarrow requería hacer más modificaciones en el código que podían evitarse usando C.

Por último, se profundizó en Dash, Plotly y Python y se reescribió casi todas las funciones que manipulan los datos de los *dataframes* originales y los *callbacks*. Este fue el método que probó ser más efectivo, pero el que más tiempo necesitó para llevarse a cabo.

Con un nueva forma de cargar el *dataframe* y un mejor código, la página redujo su tiempo de carga a más de la mitad y pudo ser desplegada en Heroku de nuevo.

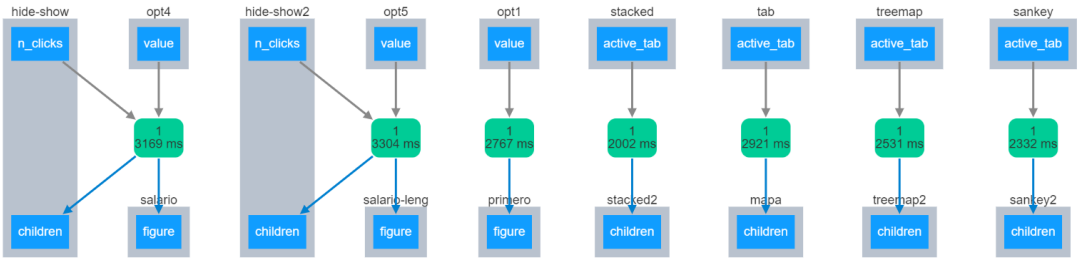

Ilustración 8 Callback graph mejorado

El segundo problema se deriva de la novedad de los conceptos y recursos que se han tratado. Especialmente en las primeras etapas de desarrollo, la curva de aprendizaje fue demasiado grande. No se contaba con conocimientos sobre Plotly, Dash, aplicaciones web o creación de estas, tan solo con un ligero entendimiento superficial sobre Python y sus estructuras.

Esto causó constantes retrasos iniciales en la planificación de los sprint y el desarrollo de la aplicación. Más tarde se empezó a dominar la técnica y se tomó un ritmo más ágil.

En un ejercicio de autocrítica se debe comentar que al inicio del proyecto se tomó una actitud muy analítica. Se trató de estudiar el mayor número posible de alternativas que ofrecían las librerías y recursos usados en el proyecto con el propósito de conocer la mejor forma de trabajar sin bloqueos. Tarea que resultó ser incompatible con el normal desarrollo de este.

Con este problema se aprendieron varias lecciones sobre el beneficio de una buena planificación. Las metodologías agiles, especialmente Kanban (y su implementación con la extensión de ZenHub), han sido muy útiles ayudando a organizar tareas en periodos temporales cerrados.

5.3. Metodología.

El software ha pasado por varias versiones hasta llegar a su estado final, pero siempre se ha seguido la misma metodología:

Primero se ha hecho una exploración de los datos de la encuesta para ver qué datos o columnas podrían ser significativamente interesantes para ser comparadas o ser visualizadas de forma gráfica.

Una vez elegidos los datos que iban a ser representados se ha estudiado el tipo de gráfico que podría representarlos mejor. Se buscó inspiración en el *dashboard* de Stack Overflow o mirando directamente en la documentación de Plotly.

Después se han inspeccionado en profundidad, especialmente si se quería añadir más de dos columnas del *dataframe*, cada columna para comprobar si eran o no compatibles.

Por ejemplo, el primer gráfico llamado "Edad Vs años desarrollando código" muestra en el eje x el número de años y el eje y la edad.

En el *dataframe* que recoge los datos de la encuesta del año 2021 esta edad se muestra en un rango. (Por debajo de 18 años, entre 18 – 24 años, etc.) mientras que el *dataframe* del año 2020 muestra la columna edad con un número entero. Para poder representar ambos datos usando los mismos rangos hubo que hacer una transformación en el *dataframe* del año 2020.

Tras tener un concepto del gráfico, se ha creado un notebook con la estructura que debe tener toda aplicación Dash. (Este concepto se puede ver más en profundidad en el apartado 3. Conceptos teóricos de este documento). Se genera una función que manipule los datos, un *layout* que muestre una funcionalidad básica y finalmente el *callback* que genere el gráfico.

Este Notebook se modificaba hasta estar conforme con el diseño y el procesamiento de datos y finalmente se copiaba en el archivo donde estaba el resto del *dashboard*. Se realizaban algunas pruebas para comprobar que el nuevo código funcionaba correctamente y se repetía el proceso de nuevo.

La mayoría de notebooks realizados fueron archivados para poder ser consultados más tarde. Se pueden encontrar en el repositorio bajo el directorio *"old"*.

5.3. Pruebas: Unittest.

Las pruebas de código se han añadido en las fases tardías del desarrollo del proyecto. Dado que el único parámetro del que depende la aplicación es el fichero que contiene los datos de la encuesta, se ha usado unittest para definir varios test que comprueban el estado del archivo csv y el *dataframe* resultante.

6. Trabajos relacionados

6.1. Página web de Stack Overflow de la encuesta del año 2021.

Stack Overflow hace todos los años una encuesta donde recoge información sobre los usuarios[13] que luego muestra en una página web[14].

Ilustración 9 Página web de Stack Overflow 2021

Se ha tomado como referencia a lo largo de todo el desarrollo del proyecto.

7. Conclusiones y líneas de trabajo futuras

7.1 Conclusiones.

Los objetivos fijados al iniciar el proyecto se han cumplido satisfactoriamente: al finalizar el desarrollo se ha desplegado una aplicación web que muestra datos en forma de gráficos interactivos.

Tener una línea de trabajo definida es algo que ayuda a tomar decisiones sobre la dirección que va a tomar el proyecto, pero inesperadamente, uno de los desafíos más grandes del proyecto ha estado relacionado con la planificación de tareas, especialmente su estimación de tiempo. En algunos momentos ha sido complicado calcular una duración aproximada de algunas actividades, especialmente cuando los conceptos eran muy novedosos y no se tenía demasiada información.

Otro de los desafíos del proyecto fue abandonar la fase formativa y comenzar a trabajar en la aplicación. Dividir el trabajo en pequeñas secciones (los notebooks con un único gráfico) ha sido una estrategia muy efectiva para ir, poco a poco, avanzando en el desarrollo sin llegar a sentirse incapaz.

Este trabajo ha sido el producto de muchas horas de investigación y trabajo. Ha sido útil para poder medir mis capacidades y los conocimientos adquiridos en el grado. En general lo finalizo con una sensación positiva.

7.2 Líneas de trabajo futuras.

Internacionalización.

Actualmente la aplicación está en un solo idioma, se tomó como punto de partida el inglés ya que los datos de la encuesta están en ese idioma.

Aceptar más datos.

La app puede ser ampliada con datos de encuestas de otros años, incluidos los de la encuesta del año 2022. Si no se desea añadir un nuevo botón en los gráficos ya generados, el *dashboard* se puede convertir en una *multipage* app.

Mejorar la app en dispositivos móviles.

La aplicación está realizada con dash-bootstrap-components y, aunque este no fue uno de los objetivos, si se accede a la dirección web del *dashboard* se puede visualizar con un dispositivo móvil. No obstante, los gráficos y las leyendas son demasiado

pequeños como para poder vistos correctamente. Un reajuste del *layout* sería una posible solución a este problema.

Recurrir a una base de datos.

En lugar de almacenar todos los datos de la encuesta en un *dataframe*, se podría guardar estos datos en una base de datos y acceder a ellos mediante consultas.

Bibliografía.

[1] ‘Dash Documentation & User Guide | Plotly’ <https://dash.plotly.com/> [accessed 2 June 2022].

[2] ‘Dash HTML Components | Dash for Python Documentation | Plotly’ <https://dash.plotly.com/dash-html-components> [accessed 1 June 2022].

[3] ‘Dash Core Components | Dash for Python Documentation | Plotly’ <https://dash.plotly.com/dash-core-components> [accessed 29 March 2022].

[4] ‘Dash Bootstrap Components’ <https://dash-bootstrap-components.opensource.faculty.ai/> [accessed 2 June 2022].

[5] Aly Sivji, ‘Interactive, Web-Based Dashboards in Python’ <https://alysivji.github.io/./reactive-dashboards-with-dash.html> [accessed 26 May 2022].

[6] ‘Part 3. Basic Callbacks | Dash for Python Documentation | Plotly’ <https://dash.plotly.com/basic-callbacks> [accessed 26 May 2022].

7 ‘Pomofocus’ <https://pomofocus.io> [accessed 27 April 2022].

8 ‘Free Educational Licenses - Community Support’, JetBrains <https://www.jetbrains.com/community/education/> [accessed 14 May 2022].

9 ‘Charming Data - YouTube’ <https://www.youtube.com/c/CharmingData> [accessed 9 June 2022].

10 ‘Plotly Community Forum - The World’s Largest Online Community for Data Science Apps and Data Visualization.’, Plotly Community Forum, 2020 <https://community.plotly.com/> [accessed 9 June 2022].

11 ‘Heroku Error Codes | Heroku Dev Center’ <https://devcenter.heroku.com/articles/error-codes#r14-memory-quota-exceeded> [accessed 9 June 2022].

12 ‘Pandas.Read_csv — Pandas 1.4.2 Documentation’ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html> [accessed 9 June 2022].

13 ‘Stack Overflow Insights - Developer Hiring, Marketing, and User Research’ <https://insights.stackoverflow.com/survey?_ga=2.189292843.1285052511.1645528337-438523718.1645528337> [accessed 9 June 2022].

[14] 'Stack Overflow Developer Survey 2021', *Stack Overflow* <https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021> [accessed 9 June 2022].