Muchas técnicas y tecnologías están luchando hoy en día por formar paquetes de herramientas para que los desarrolladores puedan utilizar cuando ponen en marcha cualquier tipo de proyecto. Las aplicaciones nativas para cada sistema operativo están siendo cada vez desplazadas a un segundo plano por las aplicaciones web, ya que éstas están disponibles desde el navegador de un ordenador, móvil o tablet, desde cualquier parte del mundo que tenga acceso a internet. Hay ciertos criterios, que un desarrollador debe tener en cuenta a la hora de crear una aplicación web, como por ejemplo:

* Una aplicación web ligera en vez de una standalone o GUI, ya que las aplicaciones web están de moda y las aplicaciones standalone o nativas ya no son populares. Esto se debe a que, las aplicaciones web tienen la ventaja de presentar la misma interfaz de usuario independientemente del sistema operativo; nos permiten realizar cambios o actualizar funcionalidades de la aplicación y desplegar las nuevas versiones de manera fácil, haciendo que los cambios realizados sean accesibles al los diversos usuarios en segundos.
* Tecnologías que tengan una curva de aprendizaje rápido y en tendencia. (React y Python)
* Que tenga las librerías más populares y más conocidas para realizar gráficos en Python (esto es porque entre más popular, más documentación tendremos disponible y más feedback podremos obtener de otros desarrolladores) (Framework Plotly)
* Que en entorno de producción tenga soporte para varios usuarios y de manera simultánea (escalabilidad). (Flask)
* Que nos permita integrar de manera sencilla BBDD tanto relacionales como no relacionales. (Flask)
* Fácil de realizar cualquier cambio y de realizar mantenimiento

A nivel de usuario:

* Que muestre los datos de manera rápida y fiable, sobretodo cuando la cantidad de datos es muy grande.
* Una página web con un diseño amigable y una interfaz sencilla de utilizar.
* Que permita realizar cambios en caliente de los datos mostrados en la gráfica.

En el amplio panorama de frameworks, bibliotecas y plugins desarrollados para la realización de esta práctica, encontramos los siguientes, donde algunos destacan más que otros:

# Pyxley:

# Proyecto con React + Flask + javascript, pero que ha sido abandonado (su último commit fue hace 4 años!) [1, 2]

# Bowtie:

# Proyecto con React + Flask + Socket IO + Eventos, gana en la parte de los eventos, ya que al parecer son más sencillos, como los que usa JQuery; pero no tiene el soporte, la comunidad y el mantenimiento que tiene Dash en estos momentos. [3, 4]

# Dash.

# Dash es un framework de código abierto desarrollado en Python para crear aplicaciones reactivas y analíticas basadas en la web [5]. Quienes se dedican al análisis, exploración, visualización y/o modelado de datos, encontrarán un uso inmediato de esta librería, ya que hace que sea muy sencillo construir una GUI codificando con sus herramientas. En su interior, además, lleva varios estilos que hacen que los elementos a crear en la aplicación sean customizables, además de poder implementar funcionalidad de librerías externas de estilos como Bootstrap, para que la aplicación aparte de verse moderna, sea también responsiva [5].

# Una de las gran cualidades que tiene, es que los componentes web se codifican y escriben en lenguaje Python, olvidándonos así de escribirlos en HTML, además, para enlazar las interacciones de usuario sobre dichos componentes y sus reacciones de procesamiento se realiza mediante simples decoradores, de esta manera nos olvidamos de utilizar JavaScript [5].

# Por tanto, podemos decir que tenemos presentes dos abstracciones:

# Componentes codificados con Python

# Reactividad mediante funciones decoradores [5]

# Dash extrae todas las tecnologías y protocolos necesarios para construir una aplicación interactiva basada en la web; es lo suficientemente simple como para vincular una interfaz de usuario mediante código Python en sólo una tarde [5].

Si hablamos de su arquitectura, los componentes de su front-end están adaptados en React [5]. React es una biblioteca diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y gracias a su comunidad de software libre, han participado en el proyecto más de mil desarrolladores diferentes [5, 6].

En Plotly han reescrito toda su plataforma web y editor de gráficos en línea con React. Dado que la comunidad de React es talentosa y prolífica, han publicado miles de componentes interactivos de alta calidad, desde combos, desplegables, controles deslizantes, selectores de calendario y tablas interactivas [5].

# En cuanto a la visualización de datos y gráficas, en Dash los gráficos se envían mediante el componente Graph, el cual representa gráficos con plotly.js. Plotly.js es ideal para Dash, dado que es declarativo, de código abierto, rápido y admite una gama completa de gráficos científicos, financieros y de negocios. Está construido sobre D3.js para exportación de imágenes vectorizadas con calidad de publicación y WebGL para visualización de alto rendimiento [5].

# Por otra parte, si hablamos de su back-end, está soportado mediante Flask. Conocido como el “micro” framework minimalista escrito en Python implementado en entornos de producción en todas partes [5, 7, 8]. Las propiedades configurables y su core es accesible para los desarrolladores de Dash. Para los desarrolladores más avanzados, las aplicaciones Dash también se pueden extender a través del variado conjunto de complementos de Flask [5].

# A pesar de esto, hay un framework el cual se puede considerar el rival más directo de Dash en estos momentos: Bokeh

# Bokeh

# Bokeh se define como una biblioteca de visualización interactiva para navegadores web, que proporciona una construcción elegante y concisa de gráficos de manera versátil. Ofrece interactividad de alto rendimiento sobre conjunto grande de datos e incluso de datos por streaming. Bokeh puede ayudar a cualquier persona que quiera hacer gráficos interactivos, dashboards y aplicaciones de datos de manera fácil y rápida [9].

# Para ofrecer tanto simplicidad y características potentes a la vez que flexibles, expone dos niveles de interfaz a los usuarios [9]:

# Modelos: interfaces de bajo nivel que brindan mayor flexibilidad a los desarrolladores de aplicaciones.

# Gráficas: interfaces de alto nivel centradas en componer visualizaciones de datos.

# Modelos.

# Uno de los principios centrales del diseño de Bokeh es que, independientemente de cómo se escriba el código en Python, el resultado será un objeto, que abarcará todos los aspectos visuales y de datos de la escena a mostrar. Este objeto será serializado, y es el que será utilizado por la biblioteca cliente BokehJS para representar el gráfico en el navegador. Los objetos de bajo nivel que comprenden un gráfico de Bokeh se denominan modelos [10].

# ../../Desktop/Captura%20de%20pantalla%202019-12-15%20a%20las%2014.01.45.p

# Ejemplo de una escena en Bokeh, donde cada una de las cajas son un modelo (plot, column, axis, etc). Al serializar, Bokeh lo hará como una única colección de objetos [10].

# Gráficas.

# La interfaz bokeh.plotting viene con un conjunto predeterminado tanto de herramientas como de estilos visuales, aunque permite también configurar el estilo visual de la gráfica a nuestro gusto. Nos permite dibujar cualquier tipo de gráficas, como gráfico de barras, gráficas con datos discretos, redes de nodos vecinos, gráficas con datos sobre mapas, entre otras [11].

Si hablamos sobre su arquitectura, podemos ver que en el back-end implementa el framework TornadoIO, aunque se puede utilizar un servidor externo si este no convence o se está familiarizado a usar otro servidor (como Flask, Django, etc) [12].

El propósito del servidor para Bokeh es facilitar a los usuarios de Python la creación de aplicaciones web interactivas, que puedan conectar eventos de interfaz de usuario del front-end con código Python real y en ejecución [12].

Lo que buscan con el servidor a groso modo son tres características:

* Que responda a eventos de la interfaz gráfica generados desde el navegador, realizando cálculos o consultas, utilizando toda la potencia de Python.
* Que envíe automáticamente actualizaciones a la interfaz de usuario (por ejemplo, widgets o diagramas en el navegador)
* Que de soporte a llamadas tanto periódicas, de tiempo de espera y asíncronas para manejar actualizaciones por streaming de datos [12].

Para el lado del cliente, existe una API para interactuar directamente con un servidor Bokeh. Esta se denomina **bokeh.client**. La API se puede utilizar para realizar modificaciones de ficheros con distintas sesiones abiertas en el servidor Bokeh [12].

# ../../Desktop/Captura%20de%20pantalla%202019-12-15%20a%20las%2015.04.54.p

# Por lo general, los navegadores web se conectan al servidor Bokeh, pero es posible conectarse desde Python usando el módulo bokeh.client [12]

# Esto puede resultar útil para realizar personalizaciones específicas de una aplicación Bokeh que está dentro de otro marco web como Flask o Django [12].

# En cuanto a la librería para gráficas del lado cliente, tenemos a BokehJS [13].

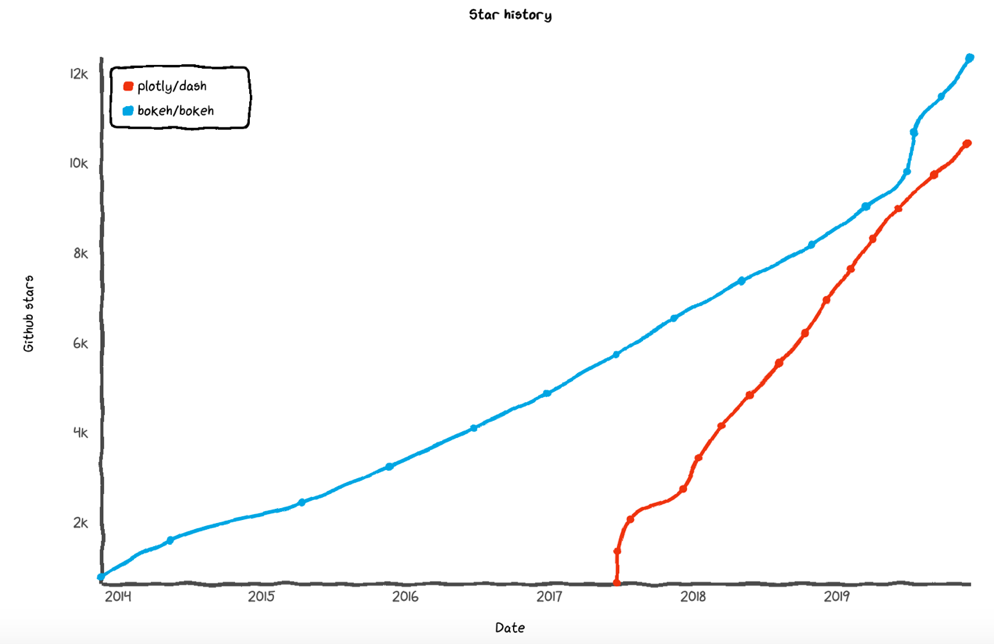
# BokehJS es la biblioteca en tiempo de ejecución del lado del cliente. Esta es finalmente con la que interactúan los usuarios desde sus navegadores. Esta biblioteca está escrita principalmente en TypeScript y es una de las cosas únicas sobre el sistema de graficación de Bokeh. La motivación para crear esta biblioteca fue la de proporcionar interacción personalizable desde fuera del propio código javascript. Además de tomar en cuenta que los gráficos pueden ser producto de una gran cantidad de datos o incluso streamings del lado del servidor, todo esto, realizándolo desde el mismo lenguaje Python [13].

# Dash vs Bokeh

# Si realizamos una comparación entre Dash y Bokeh, tendremos en cuenta distintos aspectos, como tecnologías que implementan, número de usuarios, proyectos activos en GitHub (para saber su popularidad y acogida), documentación disponible, velocidad de renderización de las gráficas y facilidad de codificación.

# Número de estrellas o puntuaciones en Github:

# En la siguiente gráfica observamos el nivel de crecimiento de votos en GitHub de las dos tecnologías:



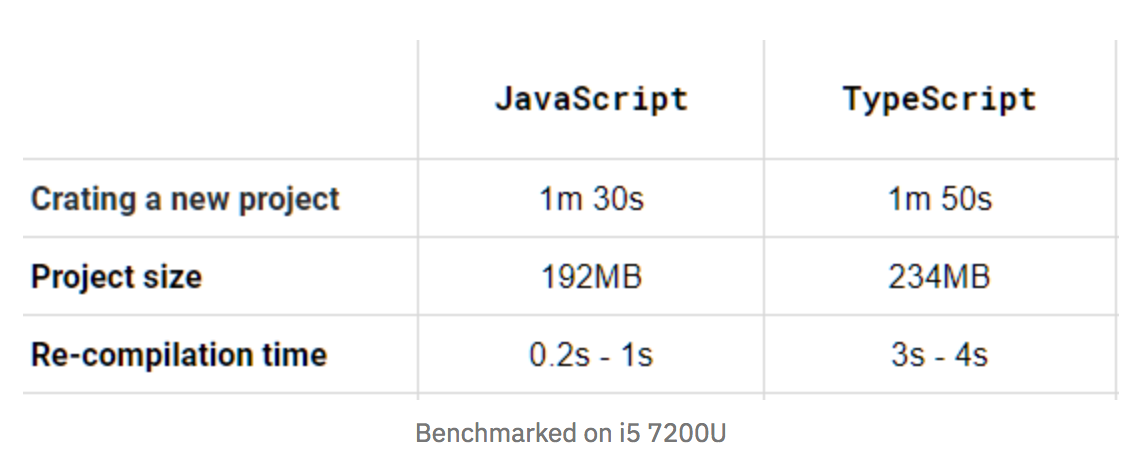
* *Representación de calificaciones por estrellas en Github para Bokeh (azul) y Dash (rojo)*[14]

# Podemos ver que, aunque Dash es mucho más reciente que Bokeh, el número de votos de desarrolladores es casi equiparable (10k frente a 12k). Lo cual indica la gran aceptación que ha tenido Dash desde que se publicó.

# No es casualidad que Dash tenga esta aceptación entre los desarrolladores Python, ya que esto se debe a diversos factores, como la facilidad de aprender y trabajar con él, muchos usuarios señalan que la curva de aprendizaje de Dash es menor que la de Bokeh, esto es debido a que Plotly les resulta mucho más intuitivo que Bokeh.js [14, 16]. Por otro lado, desarrolladores más avanzados pueden echar de menos algunas funcionalidades que en otros frameworks se hacen casi sin pensar, como por ejemplo la manera de controlar los clics sobre un elemento.

# En cuanto a la documentación, Bokeh a pesar de tener variada documentación, puede resultar un poco desordenada, incluso para encontrar solución a un problema pequeño tienen la sensación de perderse en la búsqueda, sin contar que la interacción de los foros para resolver una duda muchas veces no llega al camino de la solución; mientras que en Dash sucede todo lo contrario; la documentación, es extensa, pero a su vez concisa y ordenada, lo cual resulta fácil encontrar la solución a un problema, además, cuenta con documentación tanto offline como online, lo cual ayuda en situaciones que no se dispone de una conexión a internet rápida [14, 15, 16, 17]. La comunidad que hay detrás es otro punto a favor, ya que se preocupan por interactuar con los desarrolladores y ayudan a buscar la solución a cualquier tipo de problema. Además de escuchar las sugerencias de los usuarios para implementar nuevos componentes o revisiones de funcionalidad, cuyos cambios suelen ser publicados con las releases o versiones nuevas del framework[14, 15, 16].

# Si hablamos sobre la perspectiva de las gráficas, que es lo que nos interesa como científicos de datos, tenemos que Dash, debido a su implementación con Plotly y Flask, se renderizaban de una manera más rápida que en Bokeh, esto se debía a que funcionaban mediante diccionarios. Por ejemplo, generar una gráfica con 10x10 subplots toma alrededor de 4 segundos debido a esta característica, mientras que sin esto podría llegar a tardar alrededor de minutos mientras se llama a la función plotly\_append\_trace, lo cual sumado al tiempo que tarda del navegador en renderizar los gráficos por su cuenta, hace que se muestre de una manera más rápida en pantalla [17]. Esta característica de las gráficas con diccionarios la implementó Bokeh en versiones recientes, pero no se puede olvidar que Dash fue primero en trabajar con ella. En relación a la tecnología usada, dado que BokehJS utiliza Typescript y React utiliza Javascript, hay un tiempo de coste al momento de renderizar. Typescript requiere mucho mas tiempo de compilación que Javascript, lo cual podría afectar de manera significativa a la renderización de gráficas de para conjuntos de datos grandes [18]. En la siguiente gráfica se muestra los tiempos que pueden tardar uno y otro en distintas operaciones:



* *Tiempos de creación de un proyecto y recompilación usando JavaScript y TypeScript (Pruebas realizadas sobre un procesador Intel i5 7200 U)* [18]

Podemos observar que, el tiempo de creación del proyecto es inferior usando JavaScript (1 minuto 30 segundos frente a 1 minuto 50 segundos), aún así, para nuestro caso es insignificante; pero el tiempo de recompilación, que es el tiempo que tardará en aparecer el cambio en una gráfica, pasa aproximadamente de 1 segundo con JavaScript a 4 segundos con TypeScript, lo cual sumado a lo anteriormente mencionado, puede resultar más notorio si lo combinamos con gráficas realizadas con un conjunto de datos elevado [18].

# A pesar de esto, Dash pierde contra Bokeh en interacciones de las gráficas. Por ejemplo, Bokeh puede desplazar múltiples gráficos al mismo tiempo, lo cual no es posible en Dash en este momento. Esto se debe a que Dash apunta a solo ser de Python, y para realizar dichas interacciones necesitan algo de JavaScript para sincronizar. Por otro lado, algunas interacciones de Bokeh pueden llevar mucho tiempo en entenderlas, debido a su complejidad, lo cual entraríamos en un juego de complejidad y variedad, frente a simplicidad con algo de escases [15].

En referencia al back-end, Bokeh utiliza TornadoIO por defecto, mientras que Dash, algunos de los otros frameworks mencionados e incluso, algunos desarrolladores de Bokeh de manera externa, utilizan Flask [12].

Dado que hablamos sobre frameworks de back-end, no podemos olvidarnos de uno de los más conocidos desarrollados en Python, el denominado Django; lo cual llegados a este punto nos podemos hacer la pregunta ¿Por qué en Dash (incluso en Bokeh) se utiliza Flask en vez de Django, siendo el framework más conocido para back-end de la comunidad Python?. Repasaremos de una manera breve ambas tecnologías y finalmente contrastaremos los puntos fuertes y débiles de uno sobre otro contestando a la pregunta antes planteada.

**Django.**

Django es un framework de alto nivel que permite desarrollar aplicaciones web grandes de una manera rápida, segura y mantenible [19].

Se podría decir que Django ayuda a escribir software:

* Completo
* Versátil
* Seguro
* Mantenible
* Escalable

Completo, ya que sigue la filosofía “baterías incluidas”. Esto quiere decir que, provee todas las herramientas que necesitan los desarrolladores, siguiendo principios de diseño consistentes (como patrones de diseño), además de una amplia y actualizada documentación [19].

Versátil, porque puede funcionar con cualquier framework del lado cliente, pudiendo devolver contenido en cualquier formato (HTML, JSON, XML, entre otros) [19].

Seguro, ya que evita cometer errores de seguridad, proporcionando una manera fiable de administrar cuentas de usuario y contraseñas; o vulnerabilidades de forma predeterminada, incluida la inyección SQL, falsificación de solicitudes entre sitios y clickhacking (ataque de clics por capas) [19].

Mantenible, dado que el código está escrito siguiendo patrones de diseño, se fomenta la creación de código reutilizable, de fácil corrección o adaptación [19].

Escalable, ya que cada parte de la arquitectura es independiente de las otras, permite escalar para aumentar el tráfico al agregar hardware en cualquier nivel [19].

Pros:

* Funcionalidades de serie debido a que es “baterías incluidas” [19, 20].
* Estructura de proyecto autogenerado [19, 20].
* Reusable y ágil para el desarrollo bajo el patrón MVC [20].
* Panel de administración para gestionar base de datos [20].

Contras:

* La documentación puede resultar muy extensa y confusa, sobretodo para desarrolladores más experimentados en el framework [20].
* Conlleva algo más de dificultad el realizar APIs REST [20].
* Para un proyecto sencillo puede resultar costoso su despliegue (hay opciones mejores) [20].

**Flask.**

Flask es un “micro” framework minimalista escrito en Python que permite crear aplicaciones rápidamente bajo el patrón MVC, con un mínimo número de líneas de código. La palabra “micro” significa que mantiene un núcleo (core) simple, pero extendible, lo cual se podría traducir en un framework modular, dando la decisión de usar las herramientas que quiera el desarrollador e implementarlas usando módulos o extensiones compatibles con el núcleo. Es también conocido como “un superglue” [7, 20].

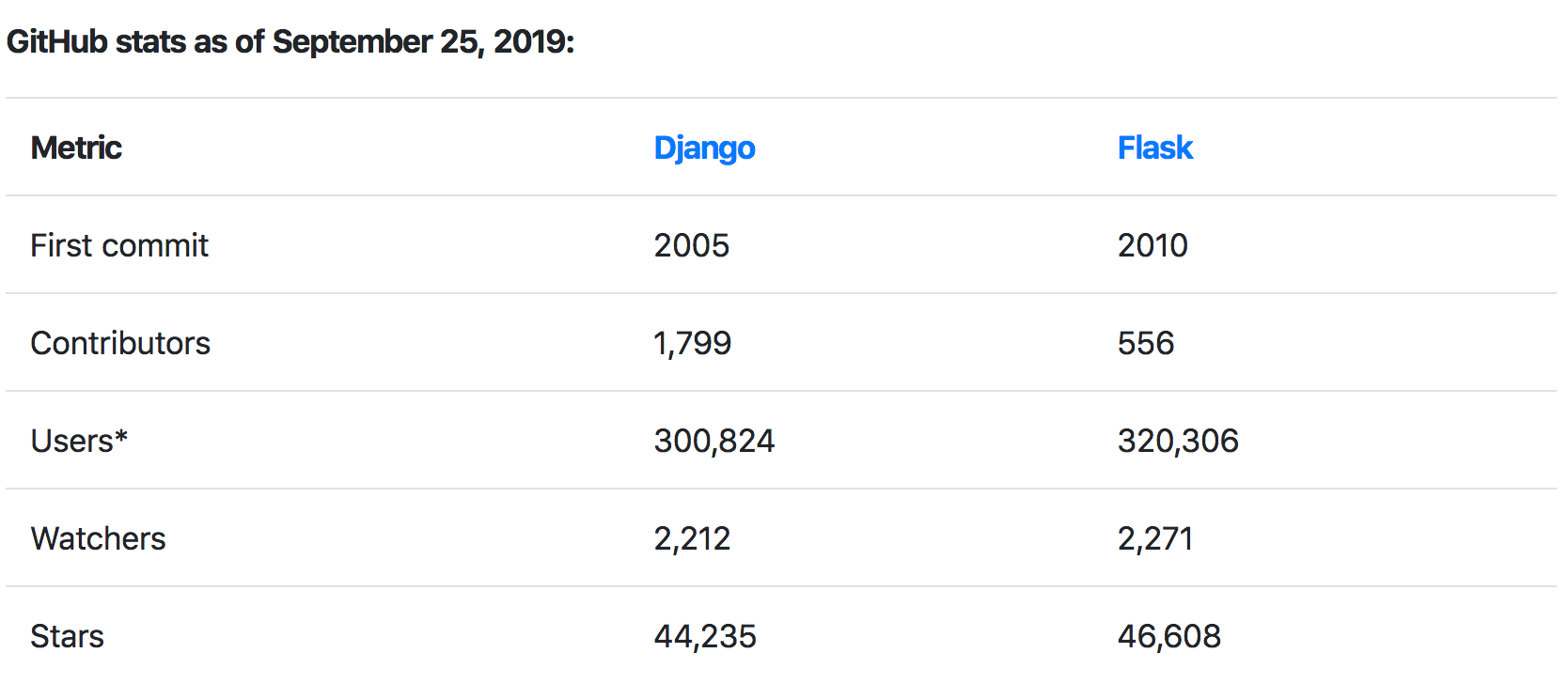
Pros:

* Sencillo de usar [20]
* Permite instalar extensiones de acuerdo al proyecto que se quiera realizar, facilitando el prototipado rápido de proyectos, o simplemente para potenciar su funcionamiento [20].
* Incluye un servidor web (WSGI Werzeug), lo cual no necesitamos instalar Apache o Nginx; además de un motor de plantillas (Jinja2) [20]
* Documentación fácil y amplia [19, 20].

Contras:

* Sistema de autenticación de usuarios es muy básico, el de Django es mucho mejor [20]
* Tiene pocos plugins, con respecto a Django, cual representación es más extensa [20].
* No tiene un ORM integrado (conexión con base de datos), es necesario instalar uno externo [20].

Algunas estadísticas acerca de los usuarios y calificaciones que han obtenido Django y Flask de acuerdo a GitHub de septiembre de 2019 [21]:



* *Se puede observar que, aunque Flask es un framework más joven, el número de usuarios es mayor al de Django.*

*Esto se debe a las características antes mencionadas, como su flexibilidad, su curva de aprendizaje rápida, la sencillez al configurar los recursos del proyecto, etc.*

Llegados a este punto, podemos responder a la pregunta ¿Por qué Dash usa Flask en vez del framework popular Django?. En conclusión desatacamos los siguientes puntos que han hecho inclinar la balanza por Flask:

* Flask permite la comunicación entre el front y back mediante objetos JSON.
* Flask se implementa de manera fácil en entornos de producción, además de facilitar la flexibilidad debido a su diseño de extensiones, esto significa dar más libertad a los usuarios de utilizar las herramientas que necesiten basados en sus gustos, necesidades de negocio y/o conocimientos.
* La curva de aprendizaje de Flask es mucho menor que con Django, debido a que es más explícito, lo cual es mucho mejor para empezar a desarrollar de manera instantánea. En Django se necesita realizar configuraciones previas de la estructura del proyecto, lo cual implica gastar tiempo en poner a punto el proyecto para empezar a desarrollar.
* Es más ligero y rápido al momento del renderizar que Django, lo cual es muy importante, sobretodo cuando las gráficas a representar son realizadas con una cantidad muy grande de datos.

# En resumen, Dash aprovecha el poder tanto de React, Plotly.js y Flask, dando a los científicos de datos un conjunto de herramientas para la creación de aplicaciones web, sin necesidad de que sean expertos en desarrollo web, y más aún, ahorrándoles el tiempo que se puede tardar en dominar lo mínimo de cada una de las distintas tecnologías utilizadas en desarrollo web.

# Bibliografía

[1] Pyxley, easy web applications with Flask and React.js: <https://www.youtube.com/watch?v=WG2HZeG6zxE>

[2] Pyxley repositorio GitHub***:*** <https://github.com/stitchfix/pyxley>

[3] Bowtie, interactive Dashboard Toolkit: <https://bowtie-py.readthedocs.io/en/latest/>

[4] Bowtie repositorio GitHub: <https://github.com/jwkvam/bowtie>

[5] Introducing Dash: <https://medium.com/plotly/introducing-dash-5ecf7191b503?>

[6] React: <https://es.wikipedia.org/wiki/React>

[7] Flask Foreword: <https://flask.palletsprojects.com/en/1.1.x/foreword/>

[8] ¿Qué es Flask?: <https://openwebinars.net/blog/que-es-flask/>

[9] Bokeh, quickstart: [https://docs.bokeh.org/en/latest/docs/user\_guide/quickstart.html - userguide-quickstart](https://docs.bokeh.org/en/latest/docs/user_guide/quickstart.html#userguide-quickstart)

[10] Bokeh. Models: [https://docs.bokeh.org/en/latest/docs/reference/models.html - bokeh-models](https://docs.bokeh.org/en/latest/docs/reference/models.html#bokeh-models)

[11] Bokeh. Plotting with Basic Glyphs: [https://docs.bokeh.org/en/latest/docs/user\_guide/plotting.html - userguide-plotting](https://docs.bokeh.org/en/latest/docs/user_guide/plotting.html#userguide-plotting)

[12] Running a Bokeh Server: <https://docs.bokeh.org/en/latest/docs/user_guide/server.html>

[13] BokehJS: [http://docs.bokeh.org/en/1.3.2/docs/dev\_guide/bokehjs.html - goals](http://docs.bokeh.org/en/1.3.2/docs/dev_guide/bokehjs.html#goals)

[14] Bokeh vs Dash – Wich is the best dashboard framework for Python: <https://www.sicara.ai/blog/2018-01-30-bokeh-dash-best-dashboard-framework-python>

[15] Creating Python Dashboards. Dash vs Bokeh: <https://www.activestate.com/blog/dash-vs-bokeh/>

[16] Bokeh vs Dash, wich is the alternative for R’s Shiny in Python?: <https://www.reddit.com/r/Python/comments/7u1jr9/bokeh_vs_dash_which_is_the_alternative_for_rs/>

[17] Plotly vs Bokeh vs …: <https://www.reddit.com/r/Python/comments/5nwk9r/plotly_vs_bokeh_vs/>

[18] Why i no longer use Typescript with React and why you shouldn’t either: <https://hackernoon.com/why-i-no-longer-use-typescript-with-react-and-why-you-shouldnt-either-e744d27452b4>

[19] Introducción a Django: [https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducción](https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n)

[20] Django vs Flask: <https://openwebinars.net/blog/django-vs-flask/>

[21] Django vs Flask in 2019: Wich framework to choose: <https://testdriven.io/blog/django-vs-flask/>