



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática

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

Documentación Técnica



Presentado por Alba Bartolome Román
en Universidad de Burgos — 11 de junio de 2022

Tutores: José Ignacio Santos Martín
Virginia Ahedo García

Índice general.

Índice general.....	1
Índice figuras.....	3
Índice tablas.....	4
Apéndice A: Plan de Proyecto Software	5
A.1. Introducción.....	5
A.2. Planificación temporal.....	5
Sprint 0.....	6
Sprint 1.....	7
Sprint 2.....	9
Sprint 3.....	10
Sprint 4.....	11
Sprint 5.....	12
Sprint 6.....	13
Sprint 7.....	15
A.3. Estudio de viabilidad.....	15
Viabilidad económica.....	15
Viabilidad legal.....	18
Apéndice B: Especificación de requisitos.....	19
B.1. Introducción.....	19
B.2. Objetivos generales	19
B.3. Catálogo de requisitos.....	19
Requisitos funcionales	19
Requisitos no funcionales	19
B.4. Especificación de requisitos	20
Apéndice C: Especificación de diseño.....	23
C.1. Introducción.....	23
C.2. Diseño de datos.....	23
C.3. Diseño procedimental.....	23
C.4. Diseño arquitectónico.....	25
C.5 Diseño gráfico.....	25
C.5.1. Barra de navegación.....	25
C.5.2. Cuerpo.....	25
C.5.3. Pie de página.....	26

Apéndice D: Documentación técnica de programación.....	27
D.1. Introducción.	27
D.2. Estructura de directorios.....	27
D.3. Manual del programador	28
D.4. Compilación, instalación y ejecución del proyecto.....	31
D.4.1 PyCharm.....	31
D.4.2 Anaconda.	32
D.5. Pruebas de sistema.	33
Apéndice E Documentación de usuario	35
E.1. Introducción	35
E.2. Requisitos de usuarios e instalación.	35
E.3. Instalación.....	36
E.4. Manual del usuario	36
Bibliografía	39

Índice figuras

<i>Ilustración 1 Gestión del proyecto usando la extensión ZenHub</i>	<i>6</i>
<i>Ilustración 2. Gráfico burndown sprint 0</i>	<i>7</i>
<i>Ilustración 3. Gráfico burndown sprint 1</i>	<i>8</i>
<i>Ilustración 4. Gráfico burndown sprint 2</i>	<i>9</i>
<i>Ilustración 5 Gráfico burndown sprint 3</i>	<i>11</i>
<i>Ilustración 6 Gráfico burndown sprint 4</i>	<i>12</i>
<i>Ilustración 7 Gráfico burndown sprint 5</i>	<i>13</i>
<i>Ilustración 8 Gráfico burndown sprint 6</i>	<i>14</i>
<i>Ilustración 9 Impuestos Seguridad Social: página oficial.</i>	<i>16</i>
<i>Ilustración 10 Diagrama de casos de uso</i>	<i>22</i>
<i>Ilustración 11 diagrama secuencial de la aplicación.</i>	<i>24</i>
<i>Ilustración 12 barra de navegación dashboard.</i>	<i>25</i>
<i>Ilustración 13 Cuerpo del Dashboard.....</i>	<i>25</i>
<i>Ilustración 14 pie de página del dashboard.....</i>	<i>26</i>
<i>Ilustración 15 documentación PyCharm.</i>	<i>29</i>
<i>Ilustración 16 Heroku: crear una nueva app.</i>	<i>30</i>
<i>Ilustración 17 Deploy Heroku: instrucciones</i>	<i>30</i>
<i>Ilustración 18 Ejecutar barra superior.</i>	<i>31</i>
<i>Ilustración 19 Ejecutar fichero.</i>	<i>31</i>
<i>Ilustración 20 Link a la aplicación: PyCharm.</i>	<i>32</i>
<i>Ilustración 21 Anaconda.</i>	<i>32</i>
<i>Ilustración 22 Barra superior Jupyter Notebook.....</i>	<i>33</i>
<i>Ilustración 23 Link a la aplicación: Anaconda.....</i>	<i>33</i>
<i>Ilustración 24 Resultado test Coverage.</i>	<i>34</i>
<i>Ilustración 25 informe para test.py.....</i>	<i>34</i>
<i>Ilustración 26 informe funct.py.....</i>	<i>34</i>
<i>Ilustración 27 Informe general</i>	<i>34</i>
<i>Ilustración 28 aplicación en un dispositivo móvil</i>	<i>35</i>
<i>Ilustración 29 Menú Navbar.....</i>	<i>36</i>
<i>Ilustración 30 gráfico sunburst (Formative and work characteristics by DevType). ..</i>	<i>36</i>
<i>Ilustración 31 Controles gráfico.</i>	<i>37</i>
<i>Ilustración 32 Gráfico treemap.....</i>	<i>37</i>
<i>Ilustración 33 Gráfico sankey.</i>	<i>38</i>

Índice tablas.

Tabla 1 Costes indirectos 17

Tabla 2 Coste total 17

Tabla 3 Licencias usadas 18

Tabla 4 CU-01: Usabilidad del menú de cabecera 20

Tabla 5 CU-02: Interacción con los gráficos..... 21

Tabla 6 CU-03: Usabilidad de los links 22

Apéndice A: Plan de Proyecto Software

A.1. Introducción.

La planificación del proyecto es una parte fundamental del ciclo de vida de este. En este caso se han usado metodologías ágiles adaptadas a las necesidades y características particulares requeridas por la naturaleza del TFG, estas se verán con mayor profundidad en las siguientes secciones.

En el primer apartado de este apéndice se comentará la organización del proyecto usando como referencia los *sprints* creados en ZenHub, herramienta utilizada para realizar un seguimiento de tareas.

En el segundo apartado se profundizará en la viabilidad económica en el contexto de una empresa real para ver sus posibles aplicaciones.

A.2. Planificación temporal

Dado que el equipo responsable del proyecto es significativamente más pequeño de lo que sería habitual se han tenido que hacer algunos ajustes en las metodologías usadas, pero se ha mantenido su filosofía:

- Se fijó la duración de los *sprints* en dos semanas. Se planteó en un inicio *sprints* con una semana de duración, pero rápidamente se vio que se necesitaba más tiempo para poder profundizar y explorar la complejidad de las diferentes herramientas que se han usado.
- Cada vez que un *sprint* finalizaba se programó una reunión con el objetivo de comentar una breve evaluación del estado del proyecto hasta ese momento y planificar el siguiente sprint.
- La monitorización del proyecto se hizo con las herramientas que facilita la extensión ZenHub. Siguiendo la filosofía de Kanban se crearon tres pipelines en las que se han ido etiquetando todas las tareas y una extra para tratar asuntos de revisión:
 - *Backlog*, el equivalente a “por hacer”, donde se han ido colocando todas las tareas que iban a ser tratadas en ese Spring.
 - En progreso, donde se han ubicado las tareas en las que se estaba trabajando en ese momento concreto.
 - *Review*. Pipeline añadida para hacer una revisión de las tareas antes de cerrarlas.
 - *Closed*, similar a “finalizadas”. Tras ser cerradas, todas las tareas se colocan automáticamente en este pipeline.

Se evaluó una duración máxima de cada tarea basándose en *story points*. Cada *story point* se ha estimado como días activos de trabajo. De esta forma se calcula la finalización de una *tarea* con 2 *story points* en 2 días de trabajo que pueden o no ser consecutivos.

Al no crear una correspondencia fija entre los *story points* y la duración de las tareas, se permite crear un espacio de trabajo mucho más flexible donde las prioridades de las diferentes tareas pueden ser modificadas en todo momento.

ZenHub ofrece una distribución predeterminada de *story points* siguiendo la secuencia Fibonacci, por lo que no siempre se ha podido elegir un *story point* adecuado. Para evitar infravalorar el tiempo requerido para finalizar una tarea y hacer hueco para imprevistos, en el caso de dudar entre dos valores, siempre se ha elegido el número superior de la secuencia.

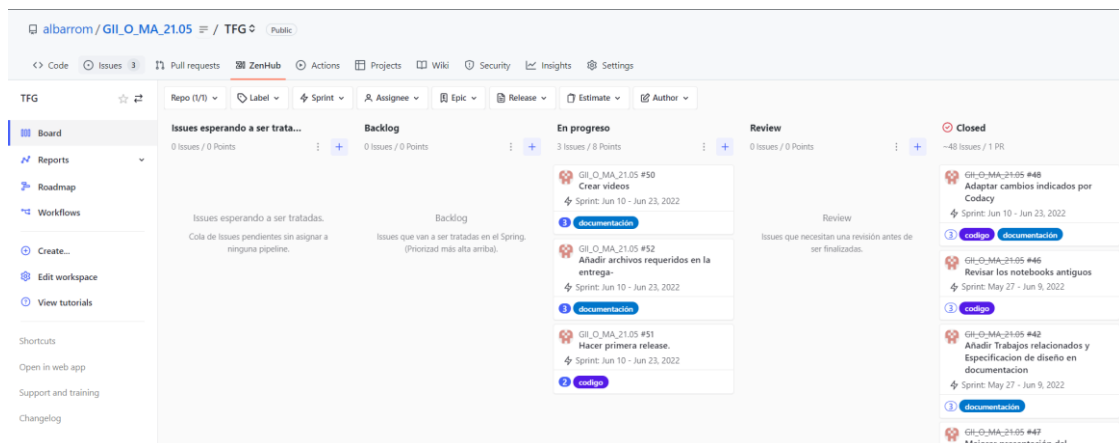


Ilustración 1 Gestión del proyecto usando la extensión ZenHub

A continuación, se expondrá un estudio detallado de cada sprint.

Sprint 0.

[Link.](#)

Fecha inicio y fin: 11 marzo - 17 marzo

Objetivo: Preparación básica: Formación e instalación de herramientas básicas necesarias.

Story points: 9 (no completados).

En ese primer sprint se realizó una labor de investigación sobre las diferentes herramientas y software que iban a utilizarse. Aún no se controlaba bien la metodología ágil y, como se puede ver en la Ilustración 2 o en el enlace al propio *sprint*, había problemas para asignar tareas o estimar tiempos. De hecho, a pesar de que las tareas

fuesen completadas antes de finalizar el *sprint*, estas no fueron cerradas antes de que terminase.

Issues:

- 001- Familiarizarse con conceptos teóricos de metodología agile básicos.
- 002- Instalación y primera toma de contacto con varias herramientas software.
- 003- Familiarización con normativa y especificaciones del TFG.
- 004- Primera toma de contacto con librerías Python.

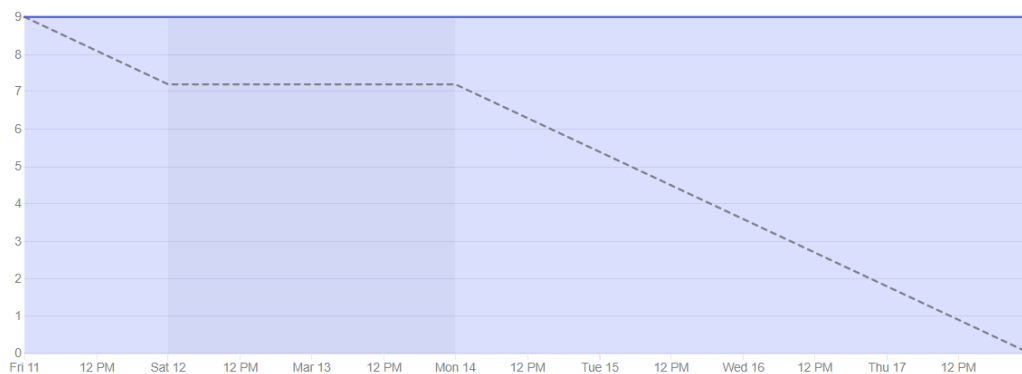
Gráfico *burndown*:

Ilustración 2. Gráfico burndown sprint 0

Sprint 1.

[Link](#)

Fecha inicio y fin: 18 marzo - 31 marzo

Objetivo: Crear un notebook que pueda mostrar un gráfico.

Story points: 35.

Este sprint se enfocó en familiarizarse con la librería SQLite de Python y crear una base de datos, con la idea de poder enriquecer el *dashboard* usando encuestas de otros años en futuras fases del proyecto.

Hubo también un cambio de planificación: se decidió crear *sprints* de 15 días en lugar de 7, que era la idea original.

Las tareas:

- Algunos campos numéricos en el csv contienen texto.
- No se puede abrir el HTML en localhost a través de Jupyter.
- Error visualizando los datos en el gráfico.

fueron creadas debido a un error de concepto con la etiqueta “problema”. Esto ha sido corregido y en futuros *sprints* no se crearán tareas con estas mismas características.

Las *milestones* cumplen la misma función que los *sprints*¹, debido a un error de concepto se estaban usando ambos. Se cesará el uso de *milestones* a partir de este sprint.

Issues:

- Cargar csv + usar librería SQLite en notebook
- Organizar todos los elementos necesarios para empezar a escribir la documentación.
- Algunos campos numéricos en el csv contienen texto.
- Crear un notebook que muestre el csv de alguna forma.
- Profundizar en Dash.
- No se puede abrir el HTML en localhost a través de Jupyter.
- Error visualizando los datos en el gráfico.

Gráfico burndown:

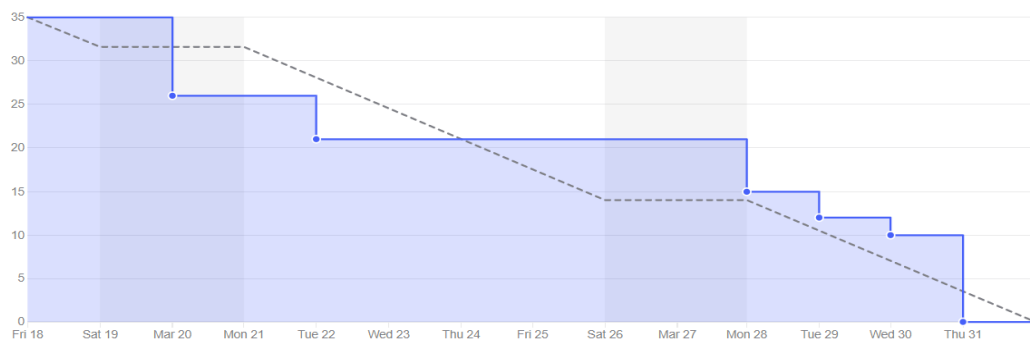


Ilustración 3. Gráfico burndown sprint 1

Sprint 2.

[Link.](#)

Fecha inicio y fin: 1 abril - 14 abril

Objetivo: Familiarizarse con la creación de varios gráficos y charts.

Story points: 39

El sprint se ha basado principalmente en la familiarización con la librería Plotly Express. Para ello se han creado diferentes gráficos y se ha investigado en profundidad cómo funcionan: añadir diferentes colores en la leyenda, usar pestañas, explorar nuevas formas de que el usuario introduzca input e introducir porcentajes.

Issues:

- Añadir información en bibliografía. Apartado Técnicas y herramientas
- Elegir gráficos
- Preprocesar datos
- Generar gráficos.
- Documentar el sprint 18/Mar - 31/Mar.
- Generar un notebook que genere una página con más de un gráfico.

Gráfico burndown:

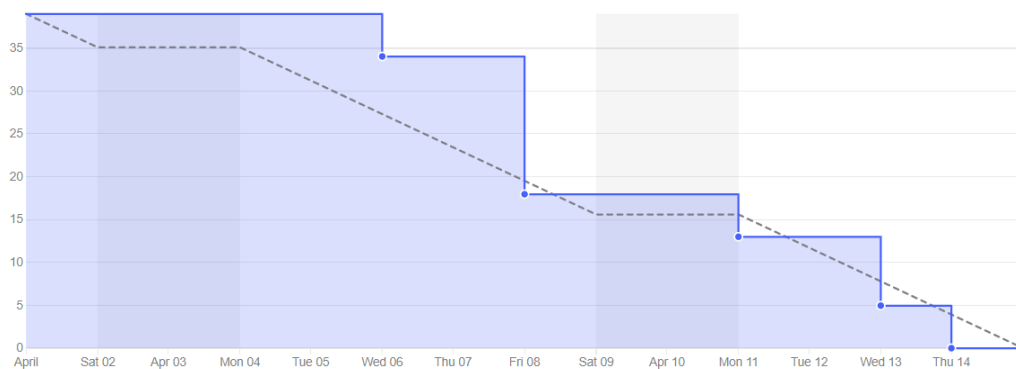


Ilustración 4. Gráfico burndown sprint 2

Sprint 3.

[Link.](#)

Fecha inicio y fin: 15 abril – 28 abril

Objetivo: *Deploy* app en Heroku.

Story points: 32

Este sprint se ha basado en continuar profundizando en la librería de Plotly Express y en realizar un primer *deploy* de la app en Heroku. Por otro lado, hubo un problema creando las tareas “Generar un gráfico que pueda procesar más de un csv” y “Crear un único archivo con varios csv”.

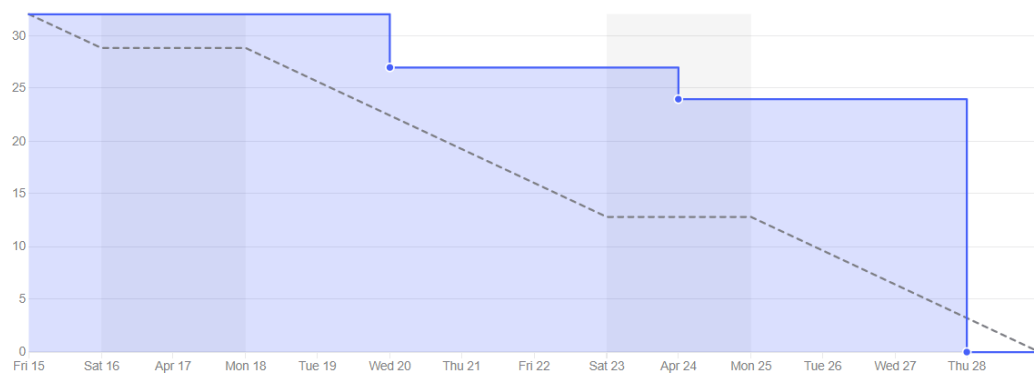
Ambas se crearon con un objetivo similar: usar encuestas de varios años para generar un único gráfico usando datos de todas las encuestas en el caso de la primera tarea y usar encuestas de varios años para generar varios gráficos, cada uno de ellos haciendo referencia a un único año en el caso de la segunda-. Tras una reevaluación de prioridades se decidió simplemente completar la primera tarea ya que la segunda no iba a aportar información o conocimientos relevantes.

Este notebook que procesaba dos csv diferentes obligó a hacer un entendimiento profundo de Dash (estructura, funciones *callback*, *inputs*, etc.) y probó ser un desafío más grande de lo que se había planeado originalmente, causando un retraso significativo en la planificación original del sprint.

También se comenzó a conectar las tareas con los *commits* a los que hacen referencia para llevar un registro más completo del proyecto.

Issues:

- Usar un path para cargar los csv.
- Familiarizarse con Heroku.
- Crear un histograma.
- Añadir documentación.
- Generar un gráfico que pueda procesar más de un CSV
- Crear un único archivo con varios csv.
- Deploy app con Heroku

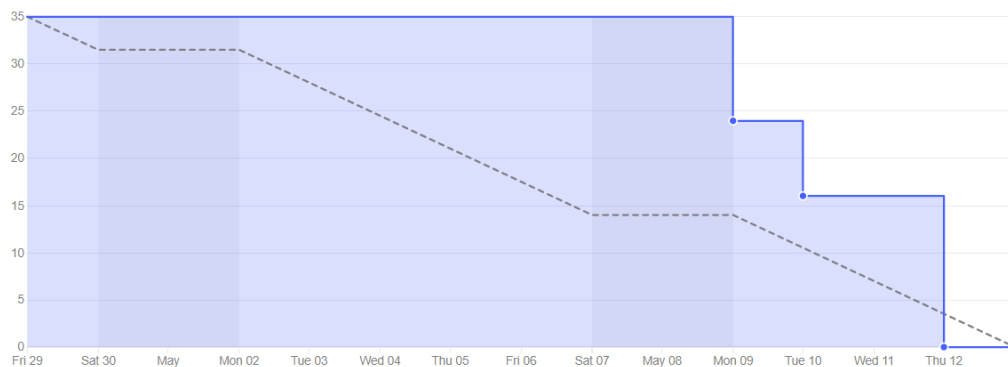
Gráfico *burndown*:*Ilustración 5 Gráfico burndown sprint 3***Sprint 4.**[Link.](#)**Fecha inicio y fin:** 29 abril – 12 mayo**Objetivo:** Crear *layout* definitivo.**Story points:** 35

El foco principal de este sprint ha sido en continuar familiarizándose con conceptos teóricos, en concreto sobre la exportación HTML interactiva en Python y la librería de componentes Bootstrap para Plotly Dash, con el fin de crear un *layout* definitivo de la app.

Este sprint cuenta con una gestión temporal particularmente mala debido a que nunca se ha tratado con conceptos web (HTML, CSS, Bootstrap, etc.). La curva de aprendizaje ha sido mucho más elevada de lo que se estimó en un primer momento y esto ha causado un retraso en el desarrollo.

Issues:

- Añadir nueva información a la documentación.
- Familiarizarse con conceptos más avanzados de Dash.
- Mejorar la navegación de la página.
- Hacer otro deploy en Heroku de la página mejorada.
- Elegir gráficos definitivos
- Crear *layout* definitivo del dashboard

Gráfico *burndown*:*Ilustración 6 Gráfico burndown sprint 4***Sprint 5.**[Link.](#)**Fecha inicio y fin:** 13 mayo – 26 mayo**Objetivo:** Crear gráficos definitivos.**Story points:** 25

Este sprint se ha enfocado en comenzar a añadir elementos (ya sean gráficos o menús) que sean definitivos. La idea principal ha sido dejar de crear notebooks con gráficos que no iban a ser incluidos en el resultado final porque no cumplían con unos estándares mínimos de calidad, pero han sido excelentes herramientas de aprendizaje, y centrarse en crear algo definitivo que pueda presentarse.

En este sprint fueron añadidos los gráficos que actualmente están en el *dashboard* final bajo el título:

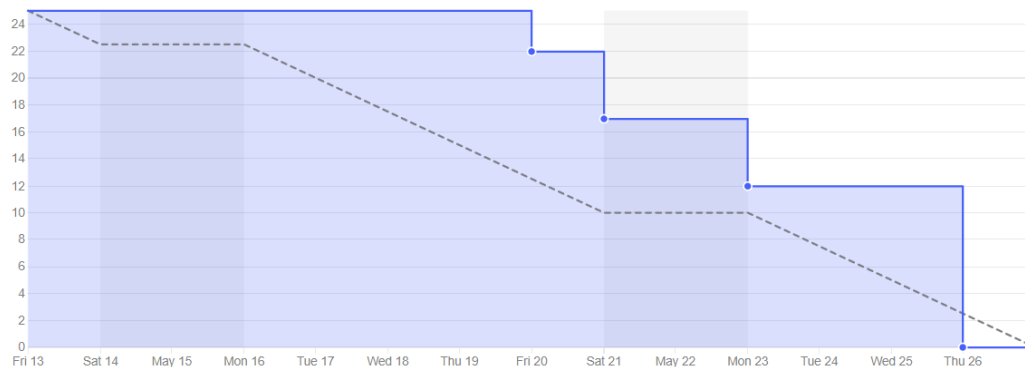
- Key territories.
- Love Vs Hate Vs Want.
- Salary and experience by developer type
- Age Vs Years Coding.

Aunque estos gráficos han sufrido alguna modificación menor tras haber sido añadidos en este sprint (cambiar colores, título o modificado etiquetas) el concepto sigue siendo el mismo que cuando se crearon.

También se ha actualizado el *layout* del *dashboard* para incluir menús, links y títulos. En general, prepararlo para que tan solo tengan que añadirse los nuevos gráficos.

Issues:

- Añadir documentación
- Organizar menús en el dashboard.
- Crear 4 gráficos definitivos.
- Añadir primera versión definitiva de 3 apartados en documentación
- Añadir primera versión definitiva de apartados en documentación.
- Añadir primera versión de conceptos teóricos

Gráfico burndown:*Ilustración 7 Gráfico burndown sprint 5***Sprint 6.**[Link.](#)**Fecha inicio y fin:** 27 mayo – 9 junio**Objetivo:** Puesta a punto de la primera versión.**Story points:** 41

Este sprint será el último sprint completo (2 semanas) antes de la entrega de primera convocatoria del TFG, por lo que tiene más tareas que los anteriores sprint. En general, se han realizado todas las actividades pendientes hasta el momento. Se puede ver un pequeño resumen de todos los cambios viendo las tareas. Haciendo una breve mención se han añadido:

- El resto de gráficos que se pueden encontrar en el dashboard ayudándose de notebooks realizados en sprint anteriores.
- Una primera versión de la documentación definitiva.
- Pruebas (usando unittest y Codacy).
- Reescrito el código de algunas funciones en el dashboard para mejorar su eficiencia.

- Mejorado la presentación del repositorio (actualizado el fichero readme).

Issues:

- Revisar los notebooks antiguos.
- Añadir Manuales de usuario y programador en documentación.
- Añadir gráficos sankey y cajas.
- Refinar código.
- Actualizar documentación + añadir aspectos relevantes.
- Reescribir los gráficos con tabs.
- Informarse, preparar y añadir pruebas del sistema.
- Revisar el tiempo de carga página.
- Documentar código y reestructurar el repositorio.
- Mejorar presentación del repositorio.
- Añadir Trabajos relacionados y Especificación de diseño en documentación.

Gráfico *burndown*:

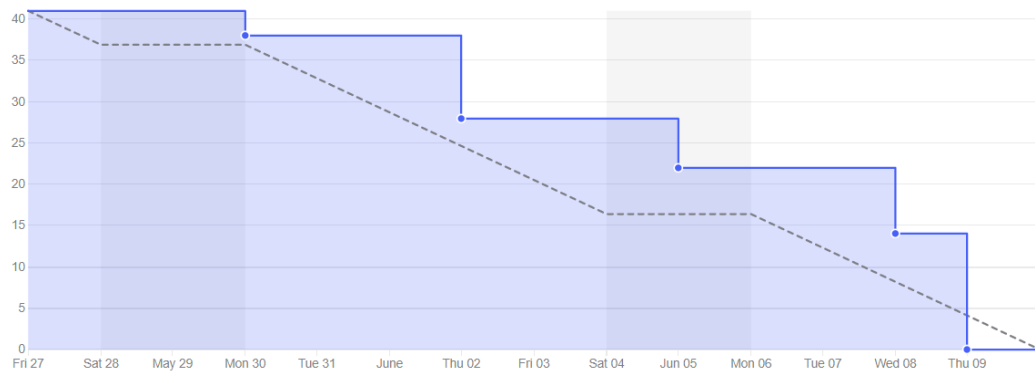


Ilustración 8 Gráfico burndown sprint 6

Sprint 7.

[Link.](#)

Fecha inicio y fin: 10 junio – 23 junio

Objetivo: Últimos detalles antes de la entrega.

Story points: 11

La fecha de entrega del TFG en primera convocatoria es el día 13 de junio, esto deja al sprint con tan solo 3 días antes de la fecha límite. Todo el trabajo ya estará hecho, tan solo es necesario hacer las últimas revisiones y cambios de última hora antes de dar el proyecto por finalizado.

Issues:

- Adaptar cambios indicados por Codacy.
- Crear videos.
- Hacer primera reléase.
- Añadir archivos requeridos en la entrega.

Dado que este *sprint* no va a estar completo, no se ha visto la necesidad de incluir una imagen con el gráfico *burndown*. Si se desea ver el estado del sprint, se puede acceder con el link añadido bajo el título.

A.3. Estudio de viabilidad.

Viabilidad económica.

En este apartado se realizarán cálculos para estimar el coste real de este proyecto en su estado actual.

Costes de personal.

Contando el inicio de este proyecto con el mismo día que inició el cuatrimestre (9 de marzo de 2022) y el fin como el día de la entrega (13 de junio de 2022), este proyecto ha tenido una duración de, aproximadamente, 3 meses.

Para simplificar los cálculos, se va a suponer que el desarrollo del proyecto lo ha llevado a cabo una sola persona, empleada a tiempo completo durante la duración de este con un salario bruto de 1.200€ mensuales.

La página oficial de la seguridad social establece unos impuestos que la empresa debe abonar², estos pueden verse en la imagen añadida a continuación.

CONTINGENCIAS SEGURIDAD SOCIAL				
Concepto	Empresa		Trabajador	Total
Contingencias comunes	23,60		4,70	28,30
Accidentes de trabajo y enfermedades profesionales	Tarifa Primas Disposición adicional cuarta, Ley 42/2006 de 28 de diciembre - P.G.E. 2007, en la redacción dada por la D.F. 5ª, del R.D.L. 28/2018 de 28/12/2018		No cotiza	

OTROS CONCEPTOS RECAUDACIÓN CONJUNTA				
Concepto	Empresa		Trabajador	Total
Desempleo	Tipo general	5,50	1,55	7,05
	Contrato duración determinada Tiempo completo	6,70	1,60	8,30
	Contrato duración determinada Tiempo parcial	6,70	1,60	8,30
Fondo de Garantía Salarial		0,20	No cotiza	0,20
Formación Profesional		0,60	0,10	0,70

COTIZACIÓN ADICIONAL HORAS EXTRAORDINARIAS				
Concepto	Empresa		Trabajador	Total
Cotización adicional horas extraordinarias	Horas extraordinarias fuerza mayor	12	2	14
	Resto horas extraordinarias	23,60	4,70	28,30

Ilustración 9 Impuestos Seguridad Social: página oficial.

Esto supondría un total de: 23.69% de contingencias comunes, 5.50 % de Fondo de Garantía Salarial y un 0,60% de formación profesional. El coste de personal mensual es

$$\frac{1200}{1 - (0.236 + 0.55 + 0.002 + 0.006)} = \frac{1200}{0.701} = 1711.84\text{€}$$

Costes hardware.

Todo el proyecto ha sido realizado con un único ordenador portátil. Este ha costado más de 300€, se considera un bien de inversión y puede amortizarse durante 4 años al 25%. Dado que este proyecto tan solo ha tenido 3 meses de duración, estos gastos no se van a añadir al estudio.

Costes software.

Su coste es virtualmente cero ya que todo el software usado para la generación de este proyecto tiene licencia de código abierto. Aprovechando los recursos que ofrece la UBU se ha podido acceder gratuitamente a software de pago (i.e. Microsoft Word o PyCharm).

Costes Indirectos.

En este apartado se detallarán costes que no están relacionados con el proyecto, pero que son necesarios para su realización. Estos serán gastos relacionados con electricidad e internet.

<i>Concepto</i>	<i>Coste</i>
<i>Internet</i>	50€
<i>Electricidad</i>	15€
<i>Total</i>	65€

Tabla 1 Costes indirectos

Coste total

<i>Concepto</i>	<i>Coste</i>
<i>Coste de personal</i>	1.711,84€
<i>Costes Software</i>	0€
<i>Coste Hardware</i>	0€
<i>Costes Indirectos</i>	65€
<i>Total (mensual)</i>	1.776,84€
<i>Total (trimestral)</i>	5.330,52€

Tabla 2 Coste total

La suma total del proyecto, habiendo estimado una duración de 3 meses, ascendería a 5.330,52€

Viabilidad legal.

En esta sección se comentará brevemente la situación legal del *dashboard*, para ello se hará una breve mención de todas las licencias usadas en la realización del proyecto.

<i>Nombre del componente</i>	<i>Descripción</i>	<i>Licencia.</i>
<i>Pandas</i>	Librería especializada en análisis de datos	BSD 3-Clause
<i>Plotly</i>	librería capaz de generar gráficos interactivos	MIT
<i>Dash</i>	Framework que ayuda a construir aplicaciones web.	MIT
<i>Dash Bootstrap Components</i>	Librería con componentes Bootstrap compatible con Dash.	Apache License 2.0
<i>Gunicorn</i>	Servidor que permite ejecutar aplicaciones Python.	MIT

Tabla 3 Licencias usadas

Todas las licencias usadas son de código abierto y permisivas. Lo único a tener en cuenta es que, si se modifica el código fuente de la librería, se pierden los permisos de distribución.

La licencia usada por este proyecto será la más restrictiva de todas. En este caso, la BSD 3-Clause.

Apéndice B: Especificación de requisitos

B.1. Introducción.

En este apéndice se recogerá la funcionalidad de la aplicación y sus componentes.

B.2. Objetivos generales

Los objetivos generales del proyecto son:

- 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 sencilla y útil.
- Permitir al usuario interactuar con los datos.

B.3. Catálogo de requisitos.

Requisitos funcionales

- RF-1. Navegación por la cabecera: el usuario debe ser capaz de interactuar con la barra de navegación.
 - RF-1.1 Usabilidad del menú de cabecera: El usuario debe poder usar el menú de cabecera para acceder a las diferentes partes del *dashboard*.
- RF-2. Interacción con los gráficos: el usuario debe ser capaz de interactuar con los gráficos mostrados en el *dashboard*.
- RF-3 Navegación del pie de página: el usuario podrá interactuar con el pie de página.
 - RF3.1 Usabilidad de los links: el usuario será redirigido a la URL correspondiente tras seleccionar los enlaces del pie de página.

Requisitos no funcionales

- RNF-1 No debe ser necesario el uso de un manual o directrices para navegar en la aplicación; ésta debe ser sencilla e intuitiva.

- RNF 2. La aplicación debe ser escalable y poder tener la opción de añadir nuevas opciones sin dificultad.
- RNF 3. La aplicación deberá mostrar la información en un tiempo razonable.
- RNF 4. La aplicación se debe poder desplegar en un tiempo razonable.
- RNF 5. La aplicación debe ser accesible mediante una URL.

B.4. Especificación de requisitos

En esta sección se verá de forma más exhaustiva los casos de uso cubiertos por los requisitos funcionales mencionados en el apartado anterior.

Primero se debe definir al único actor de la aplicación:

- Usuario: puede acceder al *dashboard* e interactuar con las diferentes opciones.

A continuación, se verán las tablas con los casos de uso que hacen referencia a los requisitos funcionales.

CU-01	Usabilidad del menú de cabecera
Actor	Usuario
Requisitos asociados	RF-1.1, RF-1
Descripción	El usuario debe poder usar el menú de cabecera para acceder a las diferentes partes del <i>dashboard</i>
Precondiciones	El menú debe ser visible en la barra de navegación
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la aplicación. 2. Se muestra el <i>dashboard</i> y la barra de navegación. 3. El usuario despliega el menú principal y accede a sus opciones.
Postcondición	La página web dirigirá automáticamente al usuario al título que haya sido seleccionado.
Excepciones	-
Importancia	Alta.

Tabla 4 CU-01: Usabilidad del menú de cabecera

CU-02	Interacción con los gráficos.
Actor	Usuario
Requisitos asociados	RF-2
Descripción	El usuario debe ser capaz de interactuar con los gráficos mostrados en el <i>dashboard</i> .
Precondiciones	Debe existir un gráfico con opciones de interacción.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la aplicación. 2. Se muestra el dashboard. 3. El usuario selecciona elementos de interacción.
Postcondición	El gráfico se actualizará acorde a la selección del usuario.
Excepciones	-
Importancia	Alta.

Tabla 5 CU-02: Interacción con los gráficos

(Continúa en la página siguiente).

CU-03	Usabilidad de los links
Actor	Usuario
Requisitos asociados	RF-3.1, RF-3
Descripción	El usuario será redirigido a la URL correspondiente tras seleccionar los enlaces del pie de página.
Precondiciones	Los links deben ser visibles en el pie de página.
Acciones	<div>1. El usuario accede a la aplicación.</div> <div>2. Se muestra el <i>dashboard</i> y el pie de página.</div> <div>3. El usuario selecciona uno de los links.</div>
Postcondición	La página web dirigirá automáticamente al usuario al link que haya sido seleccionado.
Excepciones	-
Importancia	Alta.

Tabla 6 CU-03: Usabilidad de los links

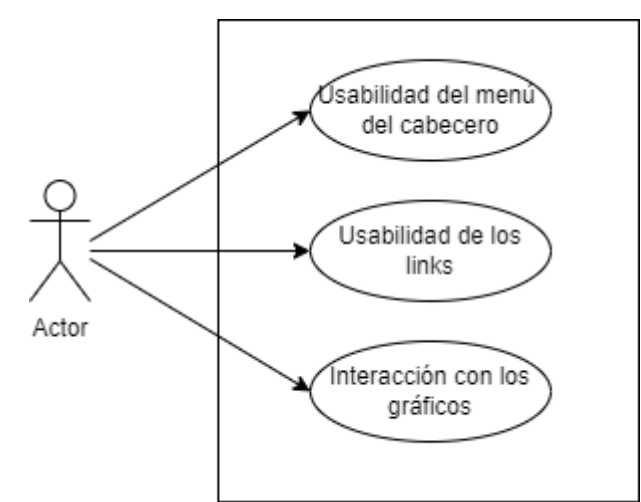


Ilustración 10 Diagrama de casos de uso

Apéndice C: Especificación de diseño.

C.1. Introducción.

Este anexo sirve para aclarar los puntos vistos con anterioridad. Define los datos de la aplicación y su arquitectura.

C.2. Diseño de datos.

El *dashboard* no cuenta con un almacén de datos fijo. Los *dataframes* basados en la encuesta de Stack Overflow son dos variables generadas al inicio del programa. El resto de funciones que procesan datos y generan nuevos *dataframes* con los que crear gráficos, generan una copia de las columnas que son relevantes, eliminan los valores nulos y envían el resultado a las funciones *callback* para generar los gráficos.

C.3. Diseño procedimental.

Este apartado contiene el diagrama secuencial por el que se recoge la aplicación.

(Continúa en la siguiente página)

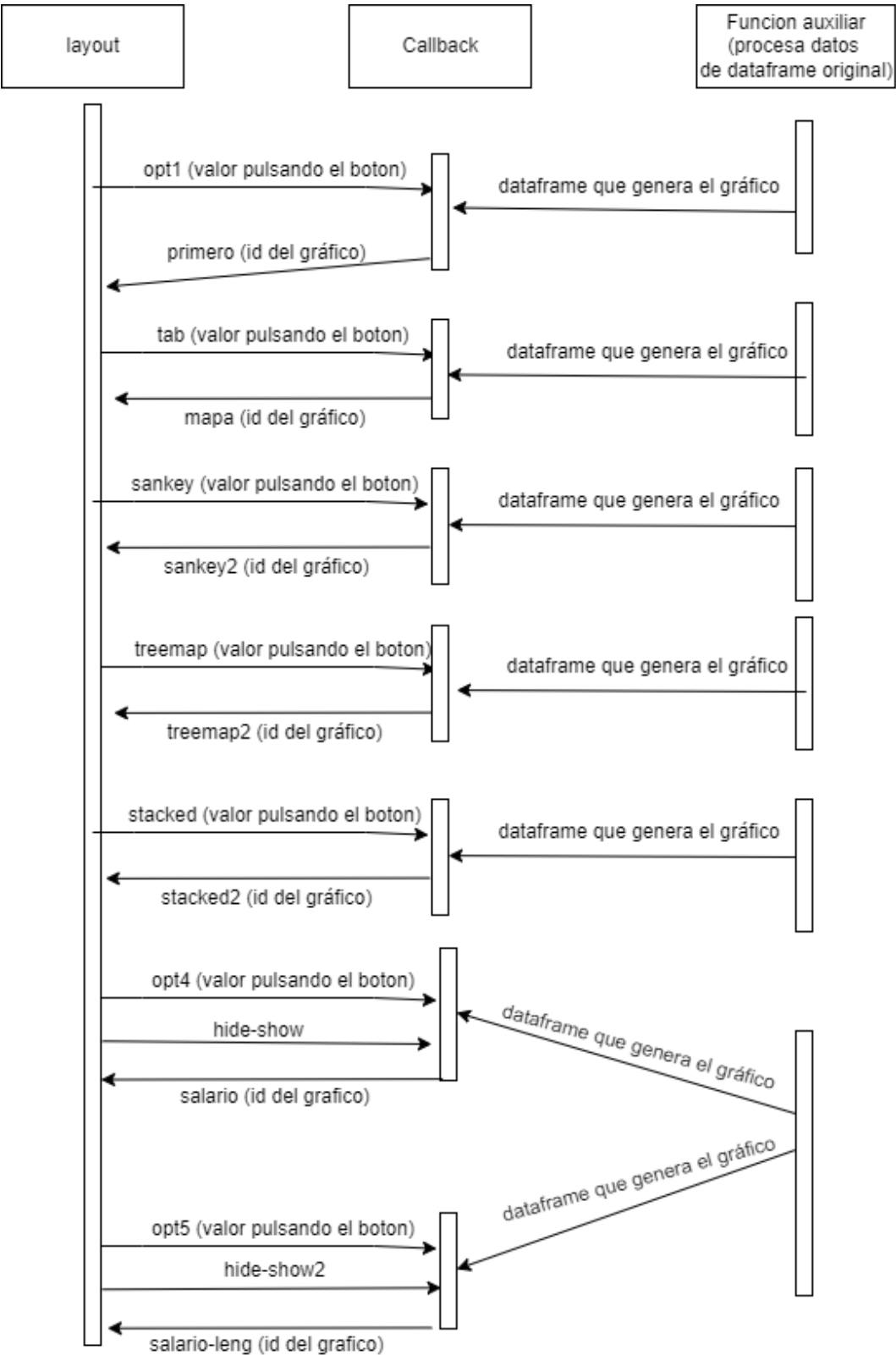


Ilustración 11 diagrama secuencial de la aplicación.

C.4. Diseño arquitectónico.

En esta sección se cubrirán los aspectos relacionados con el diseño arquitectónico de la aplicación. Se ha explicado en la memoria ampliamente el modelo – vista – controlador que se ha seguido durante la creación de la aplicación, para no repetir la misma información se va a referir al apartado 3 de la memoria.

C.5 Diseño gráfico.

En todo momento se ha intentado crear una interfaz fácil de usar e intuitiva. Se ha usado dash-bootstrap-components y dash-core-components para generar el *layout* de la página.

El *dashboard* tiene 3 partes muy diferenciadas. Se verán en profundidad en las siguientes secciones:

C.5.1. Barra de navegación.



Ilustración 12 barra de navegación dashboard.

Situada en todo momento en la parte superior de la página. A su izquierda tiene el título del *dashboard* y un icono con el logotipo de Stack Overflow que lleva a su página. A su izquierda se encuentra un menú *dropdown* con una lista de ítems. Estos llevarán a las diferentes partes de la página.

C.5.2. Cuerpo.

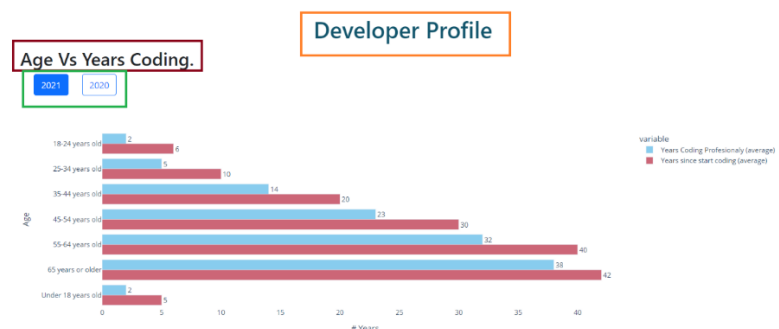


Ilustración 13 Cuerpo del Dashboard.

Aunque haya varios gráficos, todos siguen la misma estructura por lo que tan solo se va a comentar uno de ellos.

- El título en el recuadro naranja hace referencia a uno de los ítems del dropdown que hay en la barra de navegación. Hay 3 en total.
- El título en el recuadro granate es el título del gráfico. Además, otorga una pequeña descripción de las variables que van a encontrarse.
- Los botones del recuadro verde sirven para interactuar con el gráfico y hacer que cambie de valores.

C.5.3. Pie de página.

El pie de página tiene 5 elementos que sobresalen.

Stackoverflow survey data (by default, data from 2021) has been used to create this dashboard. See the links for more info	Made with	Interesting links	Contact
	Dash	Stackoverflow dashboard 2021	Github
	Plotly	Stackoverflow survey	Email
2022 TFG - GIL_O_MA_21.05			

Ilustración 14 pie de página del dashboard.

Está compuesto por dos filas, una más estrecha situada al fondo de la página con el título de la aplicación y una más ancha que contiene cuatro columnas más pequeñas. De izquierda a derecha estas columnas son:

- Un pequeño texto informativo sobre el dashboard.
- Links a las documentaciones de Dash y Plotly respectivamente.
- Enlaces al dashboard de Stack Overflow del año pasado y al conjunto de encuestas.
- Por último, información de contacto. Si se pincha sobre ellos llevan a la página correspondiente.

Apéndice D: Documentación técnica de programación

D.1. Introducción.

En este apéndice se verá en profundidad todos los aspectos relacionados con la documentación técnica de programación. Esto incluirá la estructura de directorios, el manual de programador, el formato de los archivos y la compilación, instalación y ejecución del proyecto.

D.2. Estructura de directorios

La estructura de directorios del proyecto es la siguiente:

- **/:** directorio raíz que contiene:
 - **/tfg_stackoverflow.py:** aplicación *dashboard* en formato *.py.
 - **/tfg_stackoverflow.ipynb:** aplicación *dashboard* en formato *.ipynb.
 - **/abbr.py:** archivo necesario para la ejecución de la aplicación.
 - **/requirements.txt:** fichero que contiene todos los módulos usados en la aplicación.
 - **/License:** licencia de la aplicación.
 - **/README.md:** fichero readme.
 - **/gitignore:** fichero gitignore.
- **/data:** Contiene los ficheros en formato csv que contienen información sobre la encuesta anual realizada por Stack Overflow en los años 2021 y 2020. Son los mismos ficheros que se pueden encontrar en el enlace3 oficial.
- **/docs:** Todos los ficheros relacionados con la documentación del proyecto.
 - **/docs/imagenes:** contiene todas las imágenes usadas en la memoria y anexos.
- **/pruebas:** contiene las pruebas unitarias de la aplicación.
 - **/pruebas/data:** ficheros requeridos para las pruebas.
- **/old:** Este proyecto ha sido incrementado gradualmente mediante gráficos completados. Es decir, una vez se ha decidido qué tipo de gráfico y que datos van a ser añadidos, se ha creado un Notebook donde se han hecho pruebas individuales sobre ese gráfico. Una vez el resultado ha sido satisfactorio, se ha añadido (o no) el nuevo gráfico al código. Esta carpeta contiene la mayoría de gráficos creados con esta metodología.
Estos ficheros han ido sufriendo modificaciones conforme el desarrollo de la aplicación avanzaba. Aunque hay un registro de todos los cambios en GitHub, en esta carpeta no se conservan las versiones de todos los notebooks, solo la más reciente.

Con el fin de ordenar la cantidad de notebooks en caso de que quieran ser

consultados en un futuro, se ha hecho una pequeña sub-clasificación. Dado que su contenido no es necesario o relevante para el producto final, esta carpeta (y, consecuentemente, sus sub-carpetas) están incluidas en gitignore.

- **/old/definitivo:** se encuentran aquellos notebooks cuyo código ha sido incluido de alguna forma en el *dashboard* final.
- **/old/versiones:** contiene un listado de notebooks con *dashboards* en diferentes procesos de desarrollo. Todos son versiones antiguas.
- **old/no definitivo:** se encuentran el resto de notebooks.

D.3. Manual del programador

Este apartado tiene como objetivo dar la información necesaria a futuros programadores que continúen trabajando y expandiendo la aplicación.

El proyecto está disponible en el [repositorio de GitHub](#). Se proporcionan dos tipos diferentes de archivo para trabajar con el proyecto:

- un fichero con extensión *.py que puede ser ejecutado en un IDE. (Se recomienda PyCharm).
- un Notebook Jupyter con extensión *.ipynb que puede ser ejecutado tanto en Jupyter Notebooks como en PyCharm.

Aunque ambos ficheros contienen la misma información se recomienda usar el archivo Notebook. Este ha sido el método preferido durante la mayor parte del desarrollo del proyecto y la instalación de las herramientas necesarias es también más sencilla. No obstante, se van a mencionar ambos métodos:

Instalación del entorno de desarrollo en IDE:

Se puede [descargar Python](#) desde su página oficial y seguir el asistente de instalación. También será necesario instalar los siguientes módulos:

- Pandas (versión 1.20.3)
- Plotly (versión 5.7.0)
- Dash (versión 2.0.0)
- Dash-bootstrap-components (versión 1.1.0)

Para terminar, se necesitará un IDE donde trabajar. Se recomienda [PyCharm](#), este ha sido el IDE usado para partes del desarrollo. Se puede resaltar la documentación del código en cualquier función

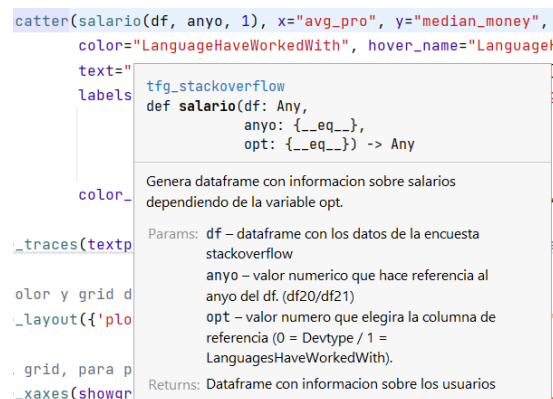


Ilustración 15 documentación PyCharm.

Instalación del entorno de desarrollo con Anaconda:

Se puede [descargar Anaconda](#) desde su página oficial y seguir el asistente de instalación. Al terminar se deberán instalar los siguientes módulos (la lista es más pequeña que en el apartado anterior porque Anaconda instala automáticamente algunas librerías):

- Plotly (versión 5.7.0)
- Dash (versión 2.0.0)
- Dash-bootstrap-components (versión 1.1.0)

Al terminar simplemente se abrirá el archivo en un Jupyter Notebook, también incluido en la instalación de Anaconda.

Requerimientos

En el directorio raíz del repositorio se ha añadido un archivo “requirements.txt” que contiene todas las librerías necesarias para su instalación. Si no se desea instalarlas una a una manualmente, se puede ejecutar:

pip install -r requirements.txt

Heroku.

Si se desea desplegar una versión propia de la aplicación se deben seguir los siguientes pasos:

Lo primero será crear una cuenta en Heroku y crear una nueva app. Se debe tener en cuenta que el nombre elegido será parte de su dirección. Si el nombre de la app es tfg-dashboard, su URL será tfg-dashboard.herokuapp.com.

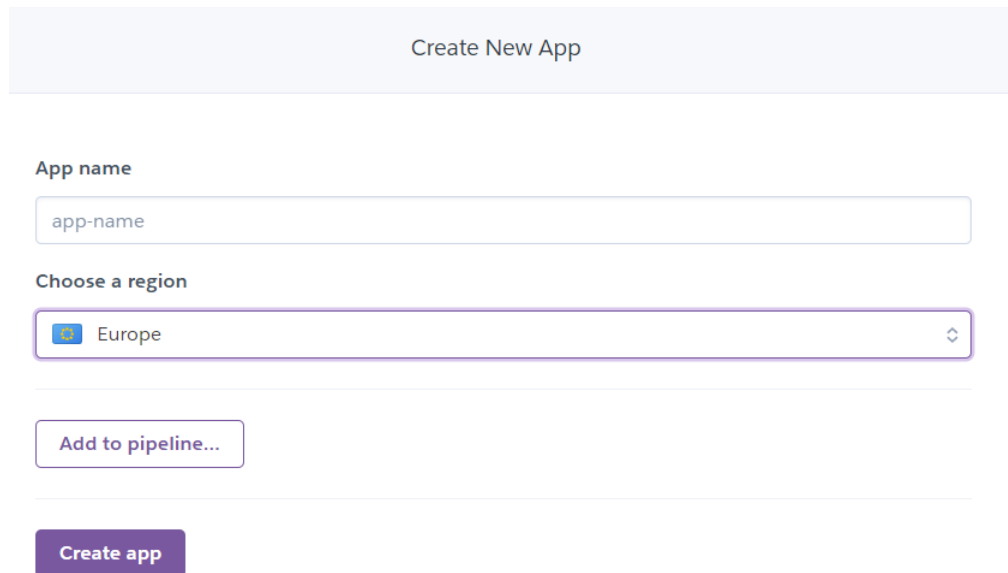


Ilustración 16 Heroku: crear una nueva app.

Uno de los puntos positivos de Heroku es que, una vez realizado estos pasos la página tiene una guía personalizada para la *app* que se acaba de crear que explica paso a paso como hacer el *deploy*:

Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Clone the repository

Use Git to clone tfg-dashboard's source code to your local machine.

```
$ heroku git:clone -a tfg-dashboard
$ cd tfg-dashboard
```

Deploy your changes

Make some changes to the code you just cloned and deploy them to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Ilustración 17 Deploy Heroku: instrucciones

Simplemente se deben seguir los pasos indicados:

- instalar [Heroku CLI](#).
- Hacer *log in* en Heroku desde la línea de comandos.
- Clonar el repositorio.
- Después de acceder al directorio donde está el repositorio, se introducen los tres últimos comandos y, tras esperar unos minutos, el *deploy* estará terminado.

D.4. Compilación, instalación y ejecución del proyecto.

Además de las librerías y un IDE a elección del usuario, no se necesita instalar nada más. Dependiendo del entorno elegido estos pasos pueden variar ligeramente.

D.4.1 PyCharm.

Una vez el proyecto se encuentre en PyCharm, simplemente se hará doble *click* en “tfg_stackoverflow.py” y se presionará en “ejecutar” en el botón de la barra superior.

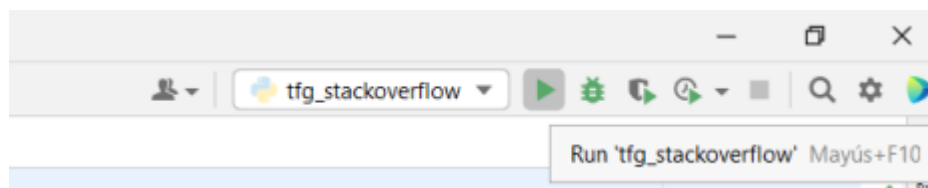


Ilustración 18 Ejecutar barra superior.

Una vez abierto el fichero “tfg_stackoverflow.py” también se puede hacer *click* con el botón derecho y ejecutar.

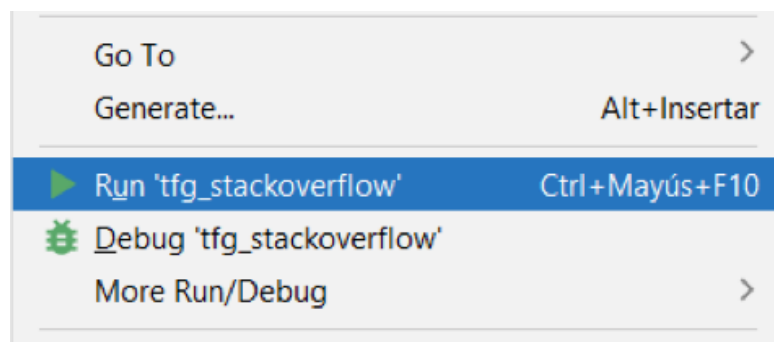


Ilustración 19 Ejecutar fichero.

O simplemente realizar la combinación de teclas Control+máyus+F10.

Independientemente del método elegido, al terminar, se encontrará un mensaje similar a este:



Ilustración 20 Link a la aplicación: PyCharm.

Simplemente se selecciona el enlace y se abrirá la aplicación.

D.4.2 Anaconda.

Cuando se abra la pantalla de Anaconda, se seleccionará Launch en el apartado Jupyter Notebooks.

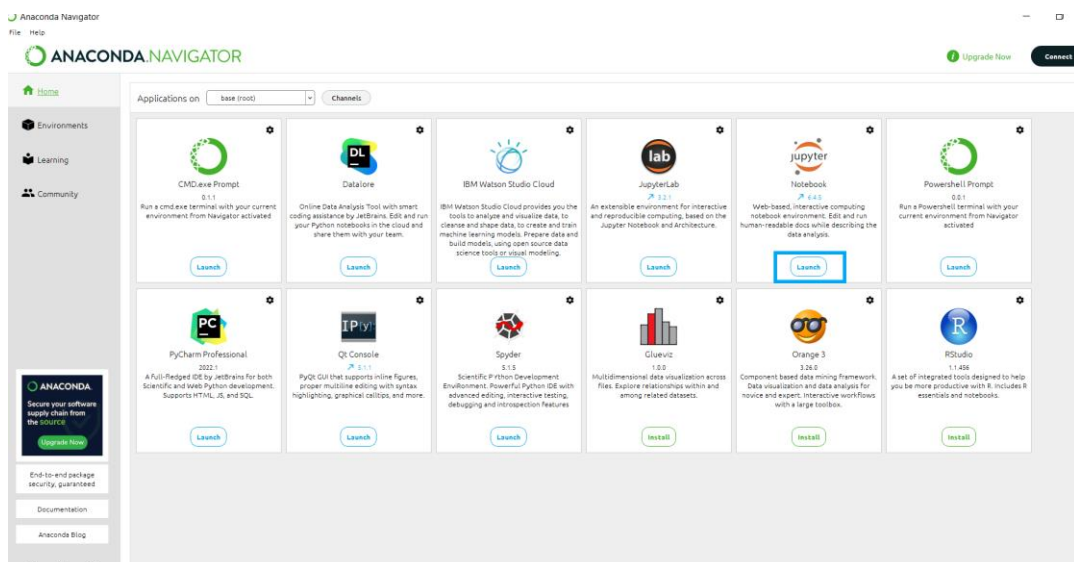


Ilustración 21 Anaconda.

Se navegará por el menú para abrir la carpeta que contiene el proyecto y después hacer doble *click* en “tfg_stackoverflow.py”.

Se prestará especial atención a la barra superior.

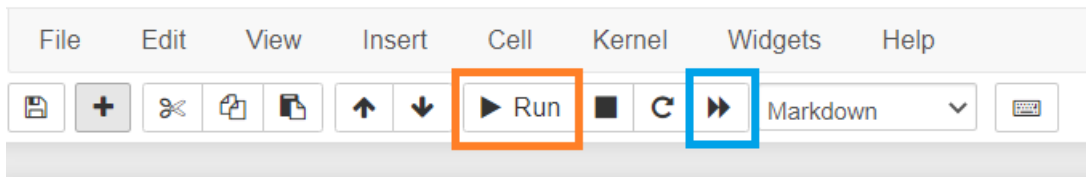


Ilustración 22 Barra superior Jupyter Notebook.

El botón que está dentro del cuadro naranja ejecutará una celda cada vez que se pulse. En el caso de este proyecto habrá que pulsar el botón varias veces hasta llegar al final de la página.

El botón que está dentro del recuadro azul ejecutará todas las celdas de una en una.

Dependiendo de las necesidades particulares del momento se puede pulsar uno u otro.

Una vez el notebook haya terminado de ejecutar se verá el siguiente mensaje en la última celda:

```
: 1 # Run the app
  2 if __name__ == '__main__':
  3     app.run_server(debug=True, use_reloader=False)
```

Dash is running on <http://127.0.0.1:8050/>

- * Serving Flask app "__main__" (lazy loading)
- * Environment: production
- WARNING: This is a development server. Do not use it in a production deployment.
- Use a production WSGI server instead.
- * Debug mode: on

Ilustración 23 Link a la aplicación: Anaconda.

Simplemente se hará *click* sobre el enlace y será redirigido a llevará a la aplicación en navegador web.

D.5. Pruebas de sistema.

En el apartado 5 de la memoria se mencionó que se ha usado unittest para definir varios test que comprueban el estado del archivo csv y el *dataframe* resultante. El control de calidad de código se ha hecho en Codacy.

La ejecución de las pruebas se puede hacer desde el propio PyCharm. Para generar el informe html y xml se debe usar la línea de comandos. La documentación de Coverage⁴ es bastante explicativa en cómo hacerlo.

Si no se tiene instalado coverage, se puede hacer con: *pip install coverage*

Luego se debe ejecutar:

- *coverage run --source= test.py*
- *coverage xml -i*
- *coverage html*

En el terminal se podrá ver el resultado del test

```
.....
-----
Ran 5 tests in 4.463s
OK
```

Ilustración 24 Resultado test Coverage.

El informe HTML se encontrará en el mismo directorio donde se hayan ejecutado los comandos.

Coverage for **test.py**: 100%

72 statements 72 run 0 missing 0 excluded

« prev ^ index » next coverage.py v6.4.1, created at 2022-06-09 03:07 +0200

Ilustración 25 informe para test.py

Coverage for **funct.py**: 100%

30 statements 30 run 0 missing 0 excluded

« prev ^ index » next coverage.py v6.4.1, created at 2022-06-09 02:49 +0200

Ilustración 26 informe funct.py

Coverage report: 100%				
coverage.py v6.4.1, created at 2022-06-09 03:07 +0200				
Module	statements	missing	excluded	coverage
funct.py	30	0	0	100%
test.py	72	0	0	100%
Total	102	0	0	100%
coverage.py v6.4.1, created at 2022-06-09 03:07 +0200				

Ilustración 27 Informe general

Apéndice E Documentación de usuario

E.1. Introducción

En este apéndice se detallarán los requisitos mínimos para que un usuario pueda ejecutar la aplicación.

E.2. Requisitos de usuarios e instalación.

Los requisitos mínimos para poder ejecutar la aplicación serán:

- Tener un ordenador personal.
- Disponer de un navegador web.

El usuario tan solo necesita acceder a la dirección proporcionada. Aunque la página sea responsive y, teóricamente, podría ser visualizada en pantallas de cualquier resolución, se recomienda evitar las pantallas demasiado pequeñas ya que los gráficos pueden volverse ilegibles en consecuencia.

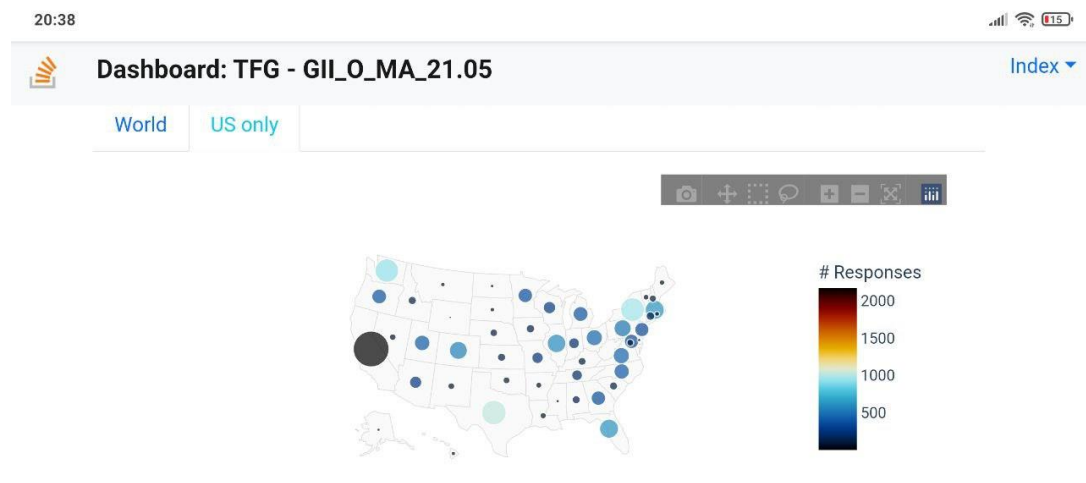


Ilustración 28 aplicación en un dispositivo móvil

E.3. Instalación.

El usuario no requiere instalar nada, tan solo debe tener acceso a internet y a un navegador.

E.4. Manual del usuario

Una vez el proyecto esté abierto en el navegador se podrá interactuar con sus diferentes menús y opciones.

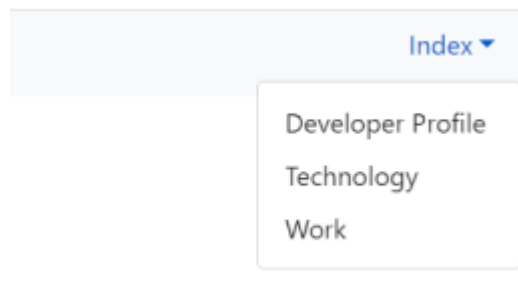


Ilustración 29 Menú Navbar.

El menú situado a la izquierda de la barra de navegación es un desplegable con un índice que contiene un enlace a las diferentes partes de la página. Si se selecciona uno de ellos, el usuario será desplazado directamente hacia esa sección.

La mayor interacción de la página será con los gráficos. Todos ellos son interactivos en mayor o menor medida. Por ejemplo, se puede pasar el ratón por encima de los elementos para ver diferentes estadísticas:

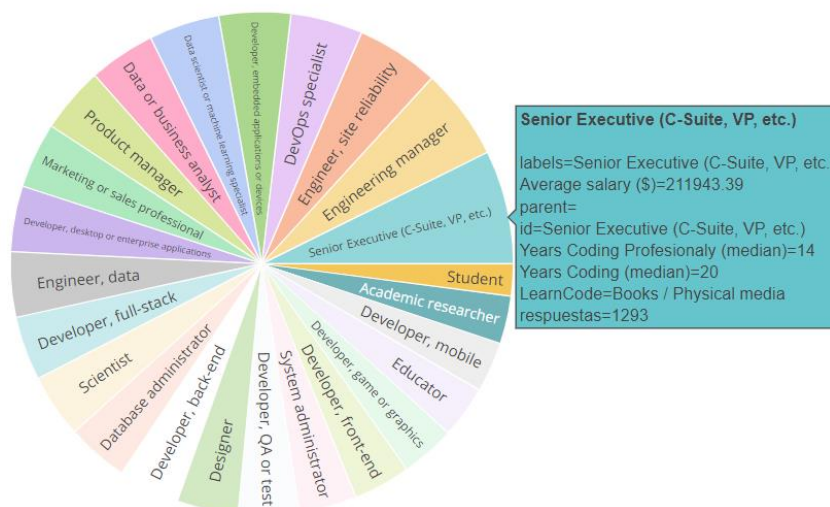


Ilustración 30 gráfico sunburst (Formative and work characteristics by DevType).

También se pueden seleccionar partes del gráfico arrastrando sobre él o usando la barra que aparece en el borde superior derecho en cada gráfico:



Ilustración 31 Controles gráfico.

De izquierda a derecha los símbolos significan:

- Realizar una captura de pantalla del gráfico en formato png.
- Hacer zoom
- Arrastrar el gráfico. (Doble *click* para resetear).
- Seleccionar elementos del gráfico (la herramienta de selección tiene forma cuadrada).
- Seleccionar elementos del gráfico (La herramienta permite hacer selección a mano alzada).
- Hacer zoom.
- Alejar el zoom.
- Auto escalado.
- Resetear ejes. (Volver a mostrar el gráfico tal y como estaba antes de modificarlo).

Es importante destacar que estas herramientas las genera Plotly automáticamente tras tener en cuenta que tipo de gráfico es y el tipo de dato con el que se está tratando. No todas estarán disponibles para gráficos que ya tienen una capa extra de interacción. Estos son:

- Treemap

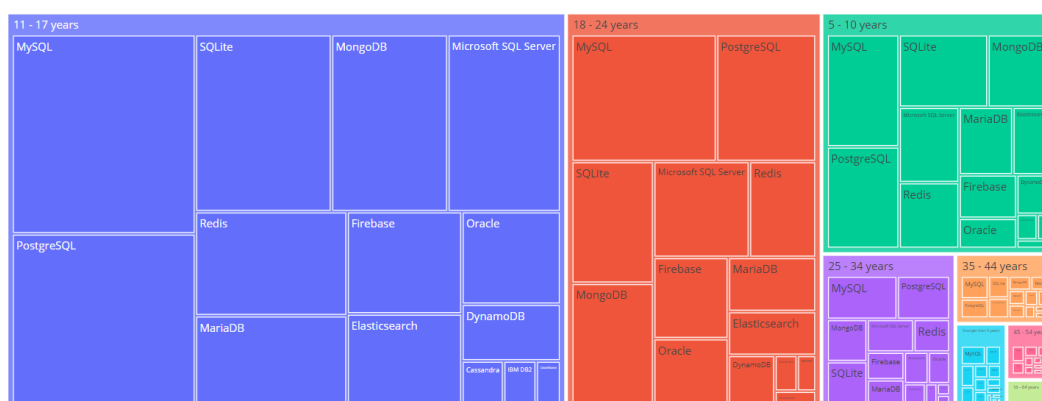


Ilustración 32 Gráfico treemap.

Se puede seleccionar cada cajita y abrir un nuevo panel cada vez.

- Sankey

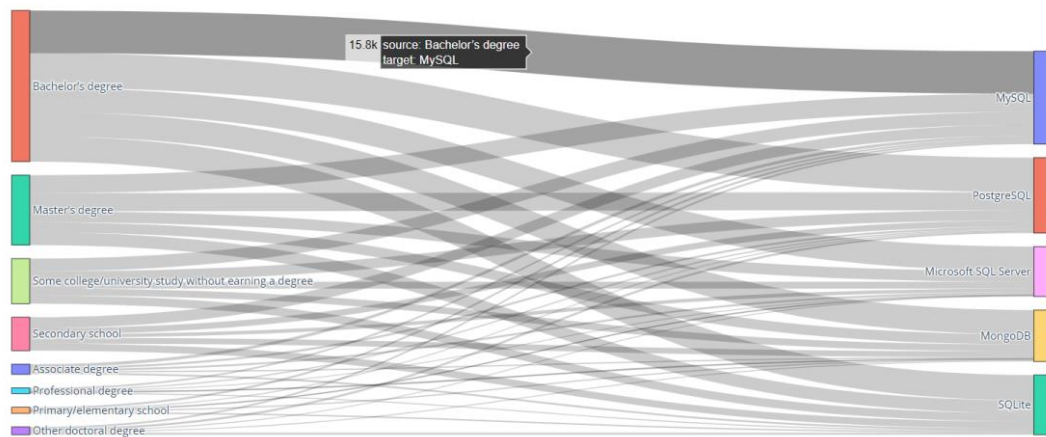


Ilustración 33 Gráfico sankey.

Se puede pasar el cursor por cada elemento visible en el gráfico para mostrar nueva información cada vez. Y también se pueden mover las diferentes etiquetas (barras laterales de colores) para reorganizar el gráfico.

Bibliografía

¹ ‘Introducing ZenHub Sprints: Automated Sprint Planning in GitHub’, Agile Project Management Best Practices & Guides | ZenHub Blog, 2021 <<https://blog.zenhub.com/introducing-zenhub-sprints/>> [accessed 11 April 2022].

² ‘Seguridad Social: Cotización / Recaudación de Trabajadores’ <<https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/10721/10957/583#580>> [accessed 9 June 2022].

³ ‘Stack Overflow Insights - Developer Hiring, Marketing, and User Research’ <https://insights.stackoverflow.com/survey?_ga=2.189292843.1285052511.1645528337-438523718.1645528337> [accessed 12 June 2022].

⁴ ‘Coverage.Py — Coverage.Py 5.0.3 Documentation’ <<https://coverage.readthedocs.io/en/coverage-5.0.3/>> [accessed 9 June 2022].