



Visualización de resultados

Grupo 1
ID de opera: 11

Miembros del grupo

[Amores Rodríguez, Manuel Jesús](#)

[De Tuero Barroso, Fernando](#)

[Jiménez Rodríguez, María](#)

[Teresa](#)

[Marín Cabrera, Marcos](#)

Enlaces de interés

[Repositorio de código](#)

[Sistema desplegado](#)

[Nuestro espacio en la wiki](#)

[Repositorio de incidencias](#)

[Opera](#)

Índice

1 Resumen	3
2 Introducción y Contexto	3
3 Descripción del Sistema	4
3.1 Descripción	4
3.2 Planificación	5
4 Entorno de Desarrollo	7
5 Gestión del código fuente	7
5.1 Gestión de ramas	7
5.2 Aprobación de cambios o parches	8
5.3 Roles en la gestión de código	9
5.4 Política de nombre y estilos de código	9
6 Gestión de la construcción e integración continua	10
6.1 Herramientas	10
6.2 Frecuencia de la construcción	10
7 Gestión del cambio, incidencias y depuración	10
7.1 Incidencias internas	10
7.2 Incidencias externas	11
8 Gestión de liberaciones, despliegue y entregas	11
8.1 Entregables	11
8.2 Liberación y despliegue	12
9 Mapa de herramientas	12
	13
10 Ejercicio de propuesta de cambio	13
11 Conclusiones y trabajo futuro	20
11.1 Conclusiones	20
11.2 Mejoras	20

1 Resumen

Como parte del trabajo a realizar en la asignatura de Evolución y Gestión de la Configuración el grupo de Visualización de Resultados es uno de los módulos de importancia presentes en el proyecto para la incorporación de una nueva mecánica en la página de congresos.

El grupo de visualización de resultados nos encargamos de que los votos que el usuario haya depositado en la base de datos, se muestren para que este pueda visualizarlos en cada momento y ver los resultados conforme a la votación. Puesto que es de importancia en toda votación dar a conocer al votante cual ha sido el resultado de la misma, nuestra labor como grupo es la de darles visibilidad e información lo más clara y visual posible para que no haya malentendidos. Nuestra opción ha sido el uso de elementos visuales típicos en la representación de votaciones como puede ser la visualización gráfica de un queso junto con sus porcentajes además de añadir algunas opciones más como la representación en las horas que ha habido más votantes entre otras.

2 Introducción y Contexto

Este proyecto se desarrolla en el marco de la asignatura de Evolución y Gestión de la configuración de la Universidad de Sevilla de cuarto curso con la intención de incorporar a una página de congresos la posibilidad de realizar votaciones. Y por otro lado, el aprender a llevar a cabo la gestión de un proyecto en otros tantos y la gestión propia en asuntos como el código, incidencias o tareas.

Elegimos la Visualización de Resultados ya que pensamos que era una rama que tocaría varios aspectos importantes no solo en temas de código como puede ser python o javascript sino también al ser un aspecto importante en el sistema de votación. Nuestra labor es llamar a los datos guardados en la base de datos y mostrarlos al usuario de una manera fácil y entendible basándonos en modelo/vista/controlador. Puesto que somos unos de los últimos módulos en el proceso de votación estamos en parte limitados por otros grupos de cara a la consulta de datos y debemos trabajar con datos introducidos directamente sin poder probar con la propia página.

Uno de los principales problemas encontrados es un poco de caos inicial en los grupos en general debido a no se tenía muy claro con qué se iba a trabajar. Otro de los problemas que tuvimos fue la falta de una clase necesaria para la gestión de

incidencias que se tuvo que recuperar de manera tardía siendo este aspecto importante en la asignatura y que nos retrasó de cara al trabajo. Una vez pasados estos problemas se llevó a cabo el trabajo sin ningún tipo de incidencia.

3 Descripción del Sistema

3.1 Descripción

El módulo de Visualización de Resultado es uno de los últimos grupos en la cadena del sistema de votación del proyecto dependiendo de las cabinas de votación y recuento de votos entre otros. A este módulo se accede tras el proceso de votación en donde el usuario decide ver el resultado de la votación. Para nuestro proyecto hacemos uso de Flask que es un framework minimalista escrito en Python que permite crear aplicaciones web de manera rápida y con un mínimo de número de líneas de código.

Nuestro módulo hace uso del sistema Modelo-Vista-Controlador que se define de la siguiente manera:

- Modelo: donde se encuentran los datos que va a manejar la aplicación ya sea mediante peticiones a otros módulos o manejando los propios.
- Controlador: encargado de la comunicación entre las vistas pasando la información necesaria para la visualización de votos y la comunicación con otros módulos dependientes.
- Vista: que mostrará de manera de visual los datos que el usuario ha solicitado de las encuestas como puede ser la visualización de las horas en las que se ha votado o el número de votos según las opciones.

Si bien no hemos requerido de la realización de cambios para la realización del trabajo ya que no partíamos de código heredado si hemos integrados otros sistemas.

Puesto que la integración no ha sido inmediata se ha recurrido a la creación de stubs para avanzar en el desarrollo del código. Los sistemas con los cuales nos hemos integrados son:

- Autenticación: Puesto que se requiere que un usuario esté autenticado para poder ver las votaciones, se llama a este sistema para comprobar que la persona solicitante está en el sistema.

- Recuento de votos: Hacemos uso de este sistema puesto que nuestra labor es la de visualizar los votos se ha acordado que para no realizar continuas llamadas a la base de datos se haga una llamada a recuento y obtener los votos necesarios a través de un json.

```
1 [
2   {
3     "title": "Ut possimus quo quidem alias sed quisquam ab.",
4     "description": "Nihil officia optio nobis odit numquam. Aut expedita consequatur ipsam assumenda animi ut optio nesciun",
5     "optional": true,
6     "multiple": false,
7     "questionoption_set": [
8       {
9         "description": "Saepe expedita quidem voluptas consequatur suscipit. Rerum quae nihil eos rerum eveniet iusto. Sint",
10        "result": {
11          "quantity": 4
12        }
13      }
14    ]
15  },
```

3.2 Planificación

Se partió al comienzo del proyecto de una planificación inicial principalmente para tener una referencia. Sin embargo debido al desconocimiento sobre hasta qué punto abarca el proyecto y la diferencia de horarios de los miembros del equipo no se ha ajustado tanto a la realidad.

El principal problema en lo referente al proyecto se ha debido en gran parte a la comunicación con la base de datos, MariaDB, y el uso y conocimiento de la tecnología Flask para coordinarlas entre sí. Una vez superado este problema que hiciera no ajustarnos a las tareas iniciales se ha proseguido sin ningún problema mayor y con un reparto de tareas más eficiente.

Las tareas se han repartido de manera dinámica aunque algunas como puede ser la realización de la documentación, diario o puesta en marcha del sistema se repartieron desde un principio para mantener constancia en dicha tarea. Por otro lado en la realización del código se ha dado total libertad para la realización de dichas tareas o en caso necesario de las incidencias o parches a aplicar. Por supuesto este reparto dinámico no ha sido un proceso caótico sino que se comunicaba de antemano vía Telegram sobre quien se asignaba que tarea además de la comunicación a través de correo ofrecida por GitHub.

La siguiente tabla representa el reparto inicial de tareas y unas fechas límites que se corresponden con los milestone que ha habido a lo largo del proyecto. Para ver el reparto de tareas real está reflejado en el diario.

Tareas	Miembro	Fecha Límite
Reunión de Planificación	TODOS	22/11/2017
Primer Milestone	TODOS	23/11/2017
Creación del entorno de desarrollo	Marcos Marín	23/11/2017
Documentación	Fernando De Tuero	14/12/2017
Elaboración del Diario	Manuel Amores	14/12/2017
Creación de diapositivas	Maria Teresa Jiménez	14/12/2017
Versión base	Marcos Marín	14/12/2017
Incremento 01-03	TODOS	14/12/2017
Reunión de Planificación	TODOS	14/12/2017
Segundo Milestone	TODOS	14/12/2017
Incremento 04-06	TODOS	21/12/2017
Integración Continua Travis	TODOS	21/12/2017
Integración del sistema	Marcos Marín	21/12/2017
Elaboración del Diario	Manuel Amores	21/12/2017
Documentación	Fernando De Tuero	21/01/2018
Tercer Milestone	TODOS	21/12/2017
Reunión de planificación	TODOS	18/01/2018
Documentación	Fernando De Tuero María Teresa Jiménez	18/01/2018
Elaboración del Diario	Manuel Amores	21/12/2017
Creación de tests unitarios	Marcos Marín	18/01/2018
Cuarto Milestone	TODOS	18/01/2018

4 Entorno de Desarrollo

Pues se han usado diferentes entornos de desarrollo tanto como para aprender de diferentes IDEs como por necesidad del trabajo.

En lo referente a entornos de desarrollo el jefe del proyecto Marcos Marín Cabrera ha optado por desarrollar directamente en el sistema operativo Ubuntu 17.04 para establecer el despliegue de Docker y la conexión con MariaDB mientras que el resto del equipo ha optado por una máquina virtual de Ubuntu donde tener la conexión con MariaDB funcionando sobre diferentes versiones de Windows.

Respecto a las herramientas utilizadas, se instalaron:

- Pycharm: Editor de Python (otros miembros del equipo usaron VSCode)
- Pip: Pseudomecanismo para la construcción de la build mediante comandos de instalación de módulos.
- Git/GitHub/GitDesktop: Programa para la gestión de versiones y del repositorio.
- Python 3.0: Lenguaje de programación usado para programar acompañado de Flask que es un framework minimalista escrito en Python para el desarrollo web.
- Chart.js: Para la introducción de gráficos basados en HTML5.

Para su correcto funcionamiento se ejecuta a través del archivo main.py y teniendo conexión con la base de datos de MariaDB para realizar las consultas necesarias.

5 Gestión del código fuente

¿cuál es el usage model del repositorio de código?

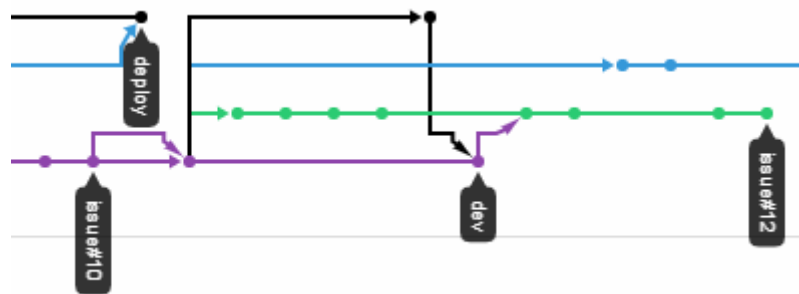
La herramienta para el uso de la gestión de código y procesos es Git en cada equipo para el control del código y GitHub para la gestión de los procesos como incidencias o tareas.

5.1 Gestión de ramas

En el proyecto interno de Visualización de Resultados se ha optado por gestionar las ramas de manera que cada una sea una tarea en relación con los incrementos a introducir además de la rama principal deploy.

- Rama deploy: Contiene en un principio el código base en funcionamiento y la versión finalizable a entregar.
- Rama repository: Contiene el contenido relacionado con la documentación y otros archivos de interés relacionados con el proyecto para tenerlos a mano en cualquier momento.
- Rama dev(Master): Contiene el código en desarrollo y las pruebas que se acaba volcando sobre la rama deploy al finalizar el proyecto. Esta rama tiene la posibilidad de tener otras ramas para la resolución de incidencias.

Igualmente se crean ramas adicionales sobre la rama development para resolver las incidencias que vayan sucediendo sobre el código que un miembro del grupo pueda solucionar sin necesidad de depender de otro.



5.2 Aprobación de cambios o parches

Como todo proyecto existe una posibilidad de que se produzca un cambio mientras se produce el desarrollo del proyecto y por ello tenemos unos pasos a seguir en caso de que se dé uno y basándonos en la documentación aportada por el equipo de integración contamos con el nuestro de manera muy similar.

Si se da la necesidad de pedir un cambio primero se consulta con el equipo donde se analiza dicha petición y la necesidad de realizar dicho cambio. Una vez que el grupo llega a un consenso sobre la necesidad de aplicar dicho cambio se asignará a una persona. Si es un cambio referente al entorno interno del equipo no es necesario que dicha petición salga del grupo de trabajo, por otro lado si se trata de un cambio que no solo afecte al equipo sino a otros módulos la petición se comunicará al grupo de integración vía Telegram para que estudie el caso lo antes posible.

En caso de que se acepte el equipo se adaptará al nuevo cambio creando en GitHub una nueva tarea con sus correspondientes incidencias. Si se deniega la petición y el grupo de integración no propone alguna alternativa, el equipo se

encargará de ofrecer una alternativa para que sea estudiada por el equipo de integración.

5.3 Roles en la gestión de código

En un principio cualquier miembro del equipo trabaja sobre la rama development y doc pudiendo crear ramas para las issues en la primera y subir los archivos necesarios sobre la segunda. Su rol se le asigna en la creación de la issue en GitHub siendo dicha persona el encargado de realizar dicha tarea.

Si requiere de ayuda para dicha tarea se asignará una segunda persona y en caso de no poder resolver la incidencia se recurrirá a la puesta en común del problema. Por otro lado el jefe del proyecto se dedicará de hacer los merge necesarios de la rama development y de la rama deploy procurando que el menor número de personas toque sobre la rama principal del proyecto.

5.4 Política de nombre y estilos de código

Puesto que se trabaja con lenguaje Python y con lo aprendido en la carrera se ha optado por usar el nombre de las variables y métodos en inglés y un estilo de código en python. Sería en minúscula y los espacios separados por barras bajas. Por ejemplo: metodo_parte2.

```
@app.route('/visres/<poll_id>/answers')
def answers(poll_id):

    poll = find_poll(poll_id)

    def votes_question(qo_id):
        votes = question_option_result(qo_id)
        return votes

    return render_template('answer.html', poll=poll, votes_question=votes_question)
```

6 Gestión de la construcción e integración continua

¿cada cuánto tiempo se realiza una construcción del proyecto? ¿Qué mecanismos IC se usan? ¿cómo?, etcétera.

6.1 Herramientas

Puesto que nuestro proyecto no usa Java no hacemos uso de la herramienta de construcción vista en clase, Maven. Sin embargo usamos Pip para la instalación y administración de paquetes necesarios para Python haciendo uso de instrucciones por terminal.

Para la integración continua se ha usado la herramienta vista en la asignatura: Travis, la cual, mediante un fichero llamado `travis.yml` se especifica el lenguaje, el nombre del archivo y la ruta del archivo que debe ejecutar cada vez que se realice un push sobre la rama `development`.

6.2 Frecuencia de la construcción

Tal y como se ha mencionado en el párrafo anterior, cada vez que se mergea con la rama `development` mediante un push venido de una petición `pull request` uniendo de esa manera el código habido en `development` con el de la rama de la `issue` correspondiente. Travis realiza la ejecución del fichero `test.py` para comprobar qué los cambios realizados son estables.

7 Gestión del cambio, incidencias y depuración

7.1 Incidencias internas

El proceso de depuración del código empieza comentándolo primero por el grupo de Telegram para que el equipo esté enterado para luego crear una incidencia en GitHub donde se explica en qué consiste el problema y como reproducirlo.

A continuación el equipo valor si el bug es prioritario o no, viendo si afecta a los servicios principales como podría la adecuada visualización de los votos o si viene a ser algo aislado como un bug meramente visual que no afecte al correcto funcionamiento de los demás servicios. Se les añade la etiqueta `bug` y si es uno importante se le pone la etiqueta `critical` además de las necesarias.

Se comentará por el grupo quien se encargará de encontrar y solucionar el bug siempre y cuando no sea aquel miembro que no la haya encontrado y se le asignará dicha incidencia aportándole ayuda en caso de que fuese necesario. Una vez que

bug sea corregido se informará a través del grupo y se dirá en la incidencia como se ha solucionado y se cerrará dicha incidencia.

7.2 Incidencias externas

En el caso de tener que informar de una incidencia a otro grupo de desarrollo primero el coordinador se pondrá en contacto vía Telegram con el coordinador de dicho grupo. Después se procede a crear la incidencia de la siguiente manera:

- Un título claro y breve sobre el asunto.
- Una descripción lo más detallada posible.
- Una lista con los pasos a seguir para la reproducción del bug.

Eso en cuanto se trata de una incidencia externa que no se deba a nuestro grupo. Respecto a incidencias externas que recibamos una vez llevado a cabo un lectura de la incidencia por el grupo se procederá a la asignación del miembro del equipo para que arregle dicho bug. Finalmente se cerrará la incidencia y se comunicará al grupo correspondiente.

8 Gestión de liberaciones, despliegue y entregas

8.1 Entregables

Los entregables del proyecto son los definidos en la wiki de la asignatura que consta de 4 hitos:

1. Ecosistema preparado.
2. Sistema de funcionamiento con incremento.
3. Taller de automatización.
4. Entrega y defensa de trabajos.

La entrega relacionada con la documentación y el código se gestiona a través de GitHub con las sus ramas pertinentes explicadas con anterioridad. La documentación incluirá lo pedido durante los diferentes hitos así como la petición en el pdf inicial de la asignatura. El código final requerido para el proyecto irá en la rama deploy hasta la fecha de su entrega. Además y como medida extra para evitar problemas también se subirá al apartado correspondiente del grupo en Opera.

8.2 Liberación y despliegue

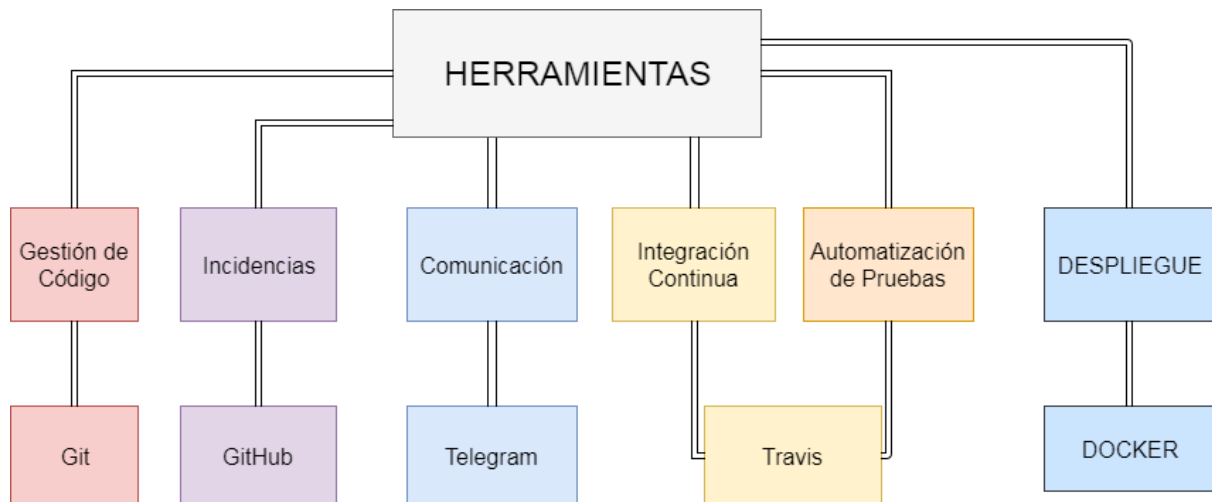
Una vez realizadas todas las mejoras propuestas y validadas sobre la rama development se realizará la unión con la rama deploy que contendrá el código estable de la aplicación. Se podrá acceder a esta versión a través tanto del enlace de GitHub puesto al inicio del documento así como el enlace del sistema desplegado.

9 Mapa de herramientas

Debe dar un esquema de cómo se conectan las herramientas que se usan en el proyecto, qué relaciones tienen o qué relaciones propondría añadir. No olvide explicar bien el mapa de herramientas. Se trata del mapa de herramientas de gestión de la configuración. El mapa de herramientas que se usen para el desarrollo (como bases de datos u otros) no es necesario que aparezca aquí.

El esquema de la imagen inferior representa las herramientas usadas a lo largo del proyecto.

- Para la gestión de código se realiza a través de Git con instrucciones de comando mediante consola.
- Las incidencias desde su creación hasta que se cierran se hacen mediante GitHub desde asignación de persona como etiquetas.
- Para la comunicación se ha usado un grupo de Telegram donde se habla sobre la realización de reuniones o algún posible problema teniendo un tiempo de respuesta rápida.
- Travis para integración continua y pruebas automatizadas.
- Docker para despliegue e integración.



10 Ejercicio de propuesta de cambio

Se propone el cambio de que en vez de gráficas estilo queso se quieren tener de estilo barra. Para ello se abre un incidencia y se crea una nueva rama con el código issue#número de la incidencia. En el ejemplo que vamos a realizar se corresponde con el 24 por lo que quedaría issue#24.

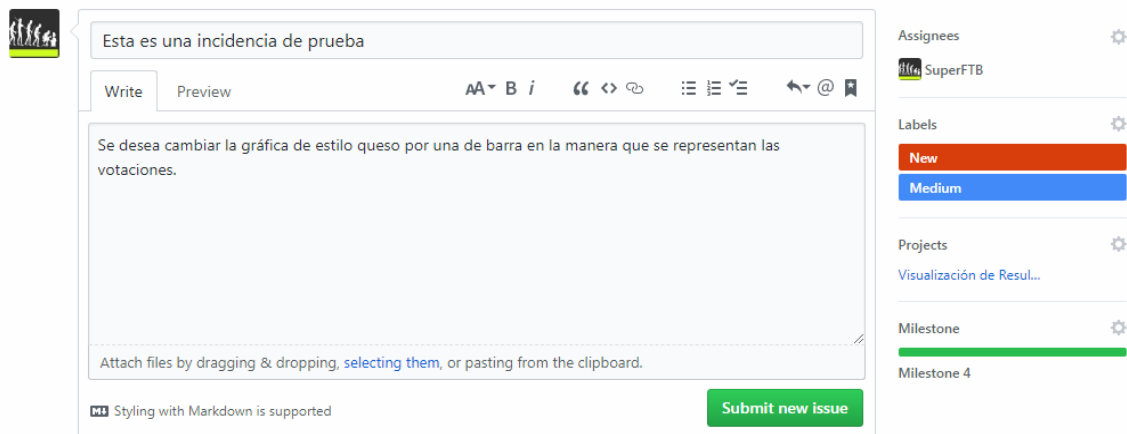
Paso 1: Vamos a nuestro GitHub y le damos a New Issue.

ProTip! Exclude your own issues with `-author:SuperFTB`.

Issue	Author	Labels	Projects	Milestones	Assignee	Sort
Encontrado un merge incorrecto	SuperFTB	Critical				
Error en nombre de archivo .sql en la configuración de Travis CI	kafok	Critical bug		Milestone 3		
Opción para guardar gráfica como imagen	kafok	Low				
Implementar la página principal de las encuestas	kafok	Medium		Milestone 3		
Implementar la página de las preguntas de una encuesta	kafok	Medium		Milestone 3		
Documentación	SuperFTB	Medium Started		Milestone 2		

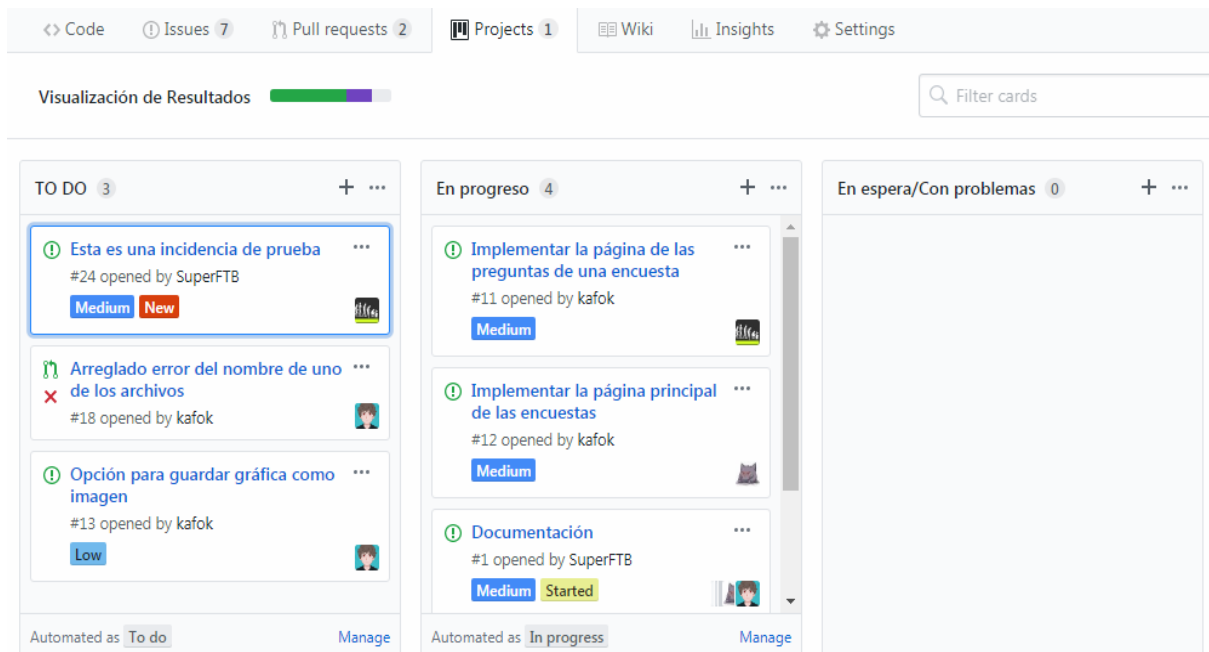
Paso 2: Entramos a la vista donde se crea la issue. Aquí tenemos para rellenar un título y una descripción siendo los campos más importantes puesto que requiere que

sean entendibles para cualquier otro miembro del equipo. Como podemos ver a la derecha tenemos varios campos a asignar. Persona a la cual se le asigna la issue, etiquetas para añadir y hacerla identificativa, proyecto al que quieres asociar dicha issue y el milestone al que pertenece.



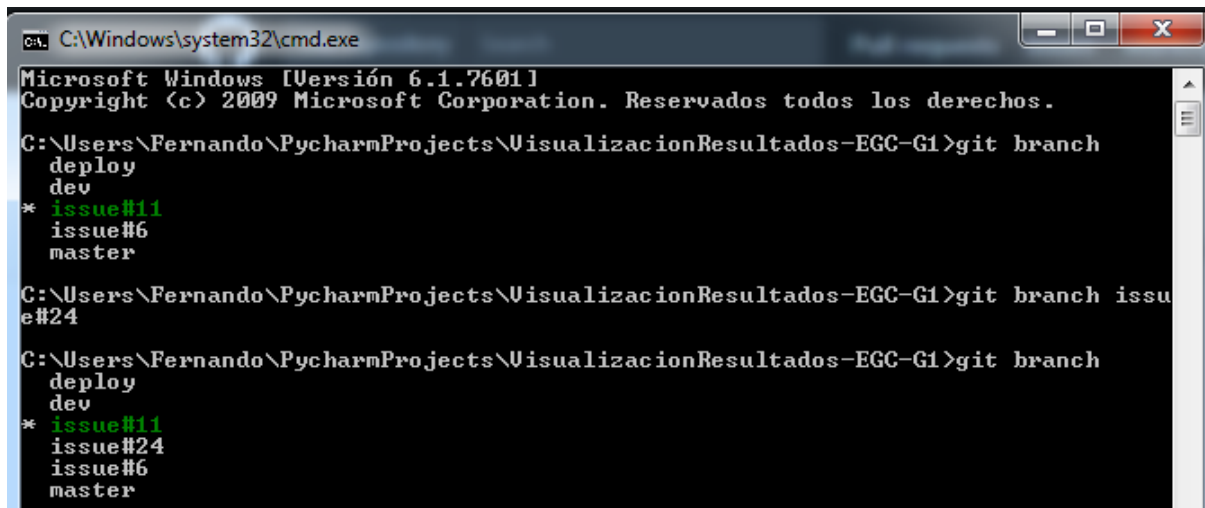
The screenshot shows the GitHub 'New issue' form. The title field contains 'Esta es una incidencia de prueba'. The body field contains the text 'Se desea cambiar la gráfica de estilo queso por una de barra en la manera que se representan las votaciones.' To the right of the form, there are configuration options: 'Assignees' with 'SuperFTB' selected, 'Labels' with 'New' and 'Medium' selected, 'Projects' with 'Visualización de Resul...' selected, and 'Milestone' with 'Milestone 4' selected. A green 'Submit new issue' button is at the bottom right of the form.

Paso 3: Podemos ver como una vez creada se sitúa en el tablero del proyecto en la columna correspondiente a TO DO.



The screenshot shows the GitHub Projects board. The top navigation bar includes 'Code', 'Issues 7', 'Pull requests 2', 'Projects 1', 'Wiki', 'Insights', and 'Settings'. Below the navigation bar, there is a 'Visualización de Resultados' section with a progress bar and a 'Filter cards' search bar. The main area displays three columns: 'TO DO 3', 'En progreso 4', and 'En espera/Con problemas 0'. The 'TO DO' column contains three issue cards: 1. 'Esta es una incidencia de prueba' (#24 opened by SuperFTB, Medium priority, New label). 2. 'Arreglado error del nombre de uno de los archivos' (#18 opened by kafok, Low priority). 3. 'Opción para guardar gráfica como imagen' (#13 opened by kafok, Low priority). The 'En progreso' column contains three issue cards: 1. 'Implementar la página de las preguntas de una encuesta' (#11 opened by kafok, Medium priority). 2. 'Implementar la página principal de las encuestas' (#12 opened by kafok, Medium priority). 3. 'Documentación' (#1 opened by SuperFTB, Medium priority, Started label). Each column has a 'Manage' button at the bottom.

Paso 4: Abrimos nuestra consola localizando la ruta de nuestro proyecto. En este caso se ha usado GitHubDesktop y se ha llamado a la consola. Vemos en que rama estamos colocados y creamos una nueva con las instrucciones que vemos a continuación.



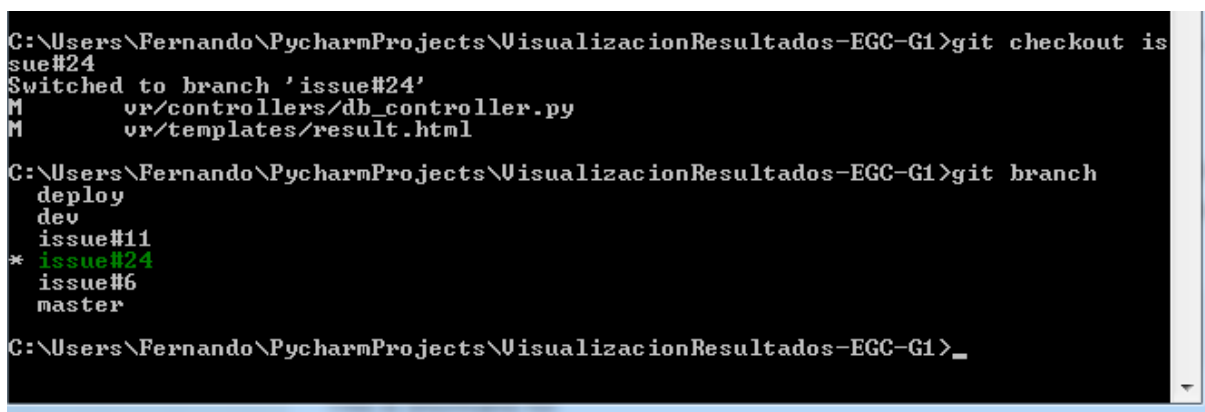
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git branch
  deploy
  dev
* issue#11
  issue#6
  master

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git branch issue#24

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git branch
  deploy
  dev
* issue#11
  issue#24
  issue#6
  master
```

Paso 5: Nos cambiamos a la rama en donde vamos a realizar dicha modificación con su correspondiente modificación.

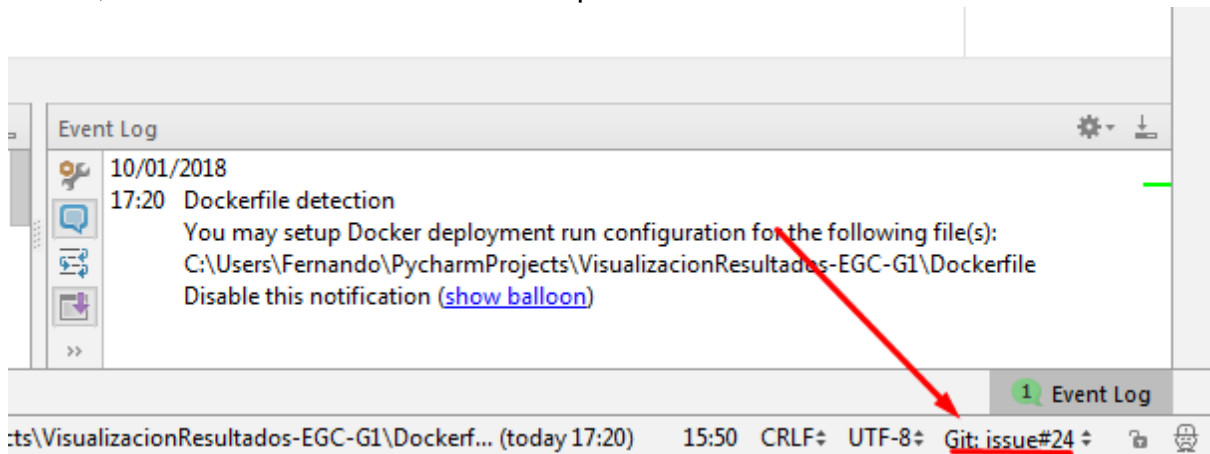


```
C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git checkout issue#24
Switched to branch 'issue#24'
M      vr/controllers/db_controller.py
M      vr/templates/result.html

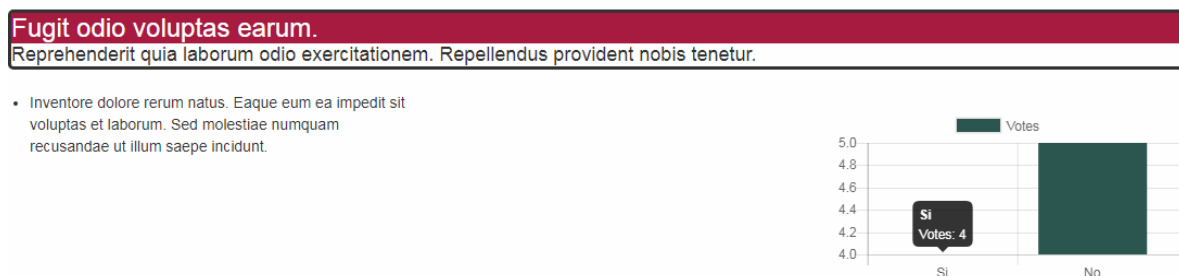
C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git branch
  deploy
  dev
  issue#11
* issue#24
  issue#6
  master

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>_
```


Toma 6: Comprobamos en nuestro entorno de desarrollo Pycharm que estamos en la rama correcta. Pycharm tiene adaptación con Git y aquí también pueden crearse ramas, cambiarse de rama entre otras opciones de Git.



Paso 7: Una vez realizado los cambios vemos efectivamente que se han producido para proceder a subir dicho cambio.



Paso 8: Procedemos a subir el cambio que hemos realizado a nuestro repositorio local. En este caso vr/templates/answer.html.

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git status
On branch issue#24
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   vr/controllers/db_controller.py
        modified:   vr/templates/answer.html
        modified:   vr/templates/result.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        vr/__pycache__/
        vr/controllers/__pycache__/
        vr/logic/__pycache__/

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>

```

Paso 9: Utilizamos el comando add para “marcar” aquellos archivos que queramos subir y status para comprobar que están correctos. Después proseguimos a hacer commit y por último push para subir lo realizado al repositorio remoto.

```
C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git add vr/templates/answer.html

C:\Users\Fernando\PycharmProjects\VisualizacionResultados-EGC-G1>git status
On branch issue#24
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

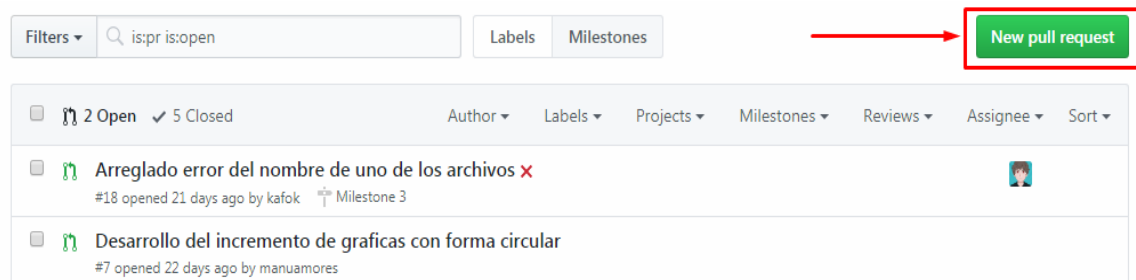
    modified:   vr/templates/answer.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   vr/controllers/db_controller.py
    modified:   vr/templates/result.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

Paso 10: Entramos en GitHub y pedimos hacer un Pull Request para que el encargado de mergear (unir) pueda hacerlo sin crear conflicto.

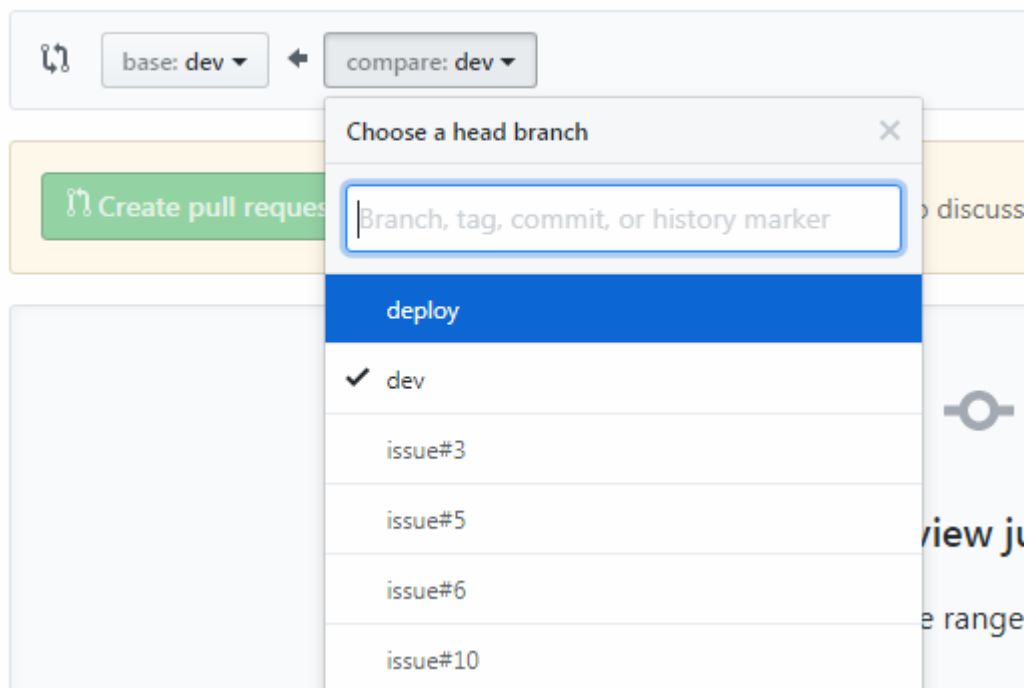


The screenshot shows the GitHub web interface. At the top, there's a search bar with 'is:pr is:open' and buttons for 'Filters', 'Labels', and 'Milestones'. A red arrow points to a green button labeled 'New pull request'. Below this, there's a table of issues. The first issue is 'Arreglado error del nombre de uno de los archivos' with a red 'x' icon, opened 21 days ago by 'kafok'. The second issue is 'Desarrollo del incremento de graficas con forma circular', opened 22 days ago by 'manuamores'.

Issue	Status	Author	Opened
Arreglado error del nombre de uno de los archivos	Open	kafok	21 days ago
Desarrollo del incremento de graficas con forma circular	Open	manuamores	22 days ago

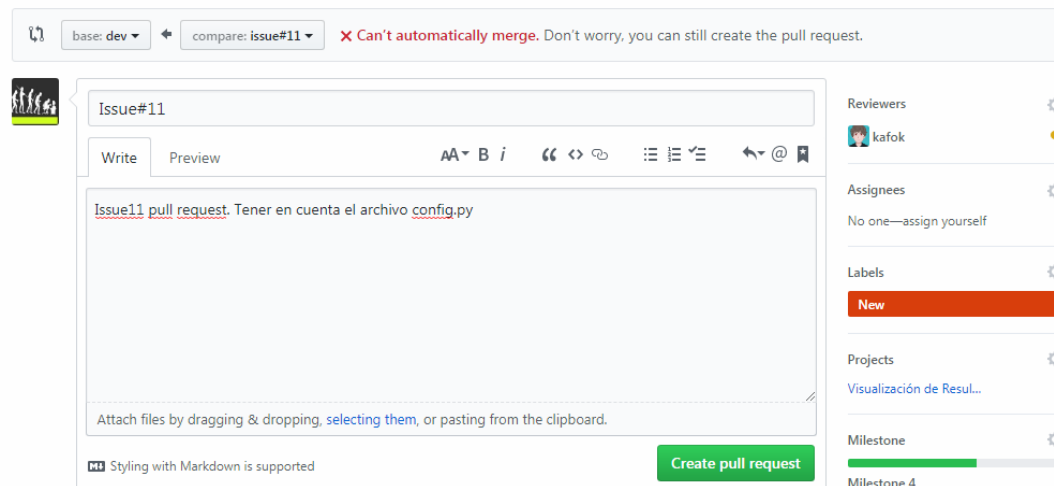
💡 ProTip! Notify someone on an issue with a mention, like: @SuperFTB.

Paso 11: Marcamos de que rama a que rama queremos hacer el pull.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



Paso 12: La issue a espera de que se realice el merge pasará a la tabla de progreso y una vez realizada se cerrará y cambiará a la columna de cerrado.

The screenshot shows a Jira board with two columns: 'TO DO' (2 items) and 'En progreso' (5 items). A red arrow points from the 'TO DO' column to the 'En progreso' column. The first issue in the 'En progreso' column is highlighted with a red box: 'Esta es una incidencia de prueba' (#24 opened by SuperFTB, Medium priority, New status). Other issues are visible in both columns.

Paso 13: Por último al estar haciendo integración continua con Travis en la rama development, se realizarán las pruebas pertinentes sobre esta.

Proyecto-EGC-G1 / VisualizacionResultados-EGC-G1 build passing

Current Branches Build History Pull Requests More options

✓ dev Merge pull request #29 from Proyecto-EGC-G1/issue#17 → #13 passed Restart build

Close #17

Commit 58b3c6a [View](#)

Compare 3a456ae..58b3c6a [View](#)

Branch dev [View](#)

Marcos Marín Cabrera authored GitHub committed

Ran for 1 min 49 sec
11 minutes ago

This job ran on our **Trusty** environment, which has been updated on Tuesday, December 12th, 2017. Read all about this [on our blog](#) and take note that you can add `group: deprecated-2017Q4` in your `.travis.yml` file to use the previous image version.

11 Conclusiones y trabajo futuro

11.1 Conclusiones

Para finalizar decir que una de las mayores dificultades de este trabajo ha consistido en la construcción de montar la base de datos para que junto con la tecnología utilizada para el proyecto funcionara correctamente. Por otro lado al ser un proyecto de gran envergadura la comunicación no siempre depende del propio equipo de desarrollo para avanzar y debe buscar la manera de no quedarse parado.

Si bien al final hemos conseguido sacar adelante el proyecto el desconocimiento por parte de la tecnología relacionada con MariaDB por parte del equipo ha hecho que se perdiera mucho tiempo. Sin embargo y como se menciona al comienzo el proyecto no debe quedarse detenido en ningún momento y en esos casos hemos sabido aprovechar para tomar decisiones sobre la gestión de incidencias, código, etc, además de ir pesando como llevar a cabo la implementación de la visualización de resultados. Como también se mencionaba en un principio y es algo que queremos recalcar para futuros grupos, hay módulos que dependen de otros para funcionar completamente y la comunicación es importante para la realización del trabajo. Y por último los integrantes del grupo deben de tener un conocimiento sobre las herramientas que van a utilizar puesto que lleva una cantidad de tiempo que se necesitará para el desarrollo del código, pruebas, integración, etc.

11.2 Mejoras

Visualización de resultados no es que tenga mucho margen más allá de mostrar los resultados pero si que hay ciertos añadidos estéticos que se pueden realizar y algunos de ellos requerirían de modificar la base de datos. Los cambios serían:

- Añadido de geolocalización: Representar en un mapa el reparto de votos, es decir, cuantos votos de qué opción ha habido en que zonas. Para ello habría que introducir en la base de datos la localización del votante.
- Escrutinio: Un porcentaje de cuántas personas del total censado ha votado hasta ahora.
- Comparación de votos: Si se trata de una encuesta realizada otros años se puede realizar una comparación con la encuesta realizada actualmente.
- Optimización: Mejora del código realizado.