

Recuento de votos

Grupo 2

ID de Opera: 97

Barrera Roldán, Antonio Jesús: 5

Cansino Suárez, Juan Carlos: 5

Martín Galván, María Inmaculada: 5

Rodríguez López, Daniel: 5

Sánchez Rodríguez, Mario: 5

Soldado Caro, Alfonso: 5

Enlaces de interés:

Repositorio de GitHub: <https://github.com/EGC-G2-Trabajo-1718/recuento-de-votos>

Wiki del grupo: [https://1984.lsi.us.es/wiki-egc/index.php/Recuento de votos -
17 18 - G2](https://1984.lsi.us.es/wiki-egc/index.php/Recuento_de_votos_-_17_18_-_G2)

Índice

Índice.....	2
1. Historial de versiones	3
2. Resumen.....	3
3. Introducción y contexto	3
4. Descripción del sistema.....	4
5. Planificación del proyecto	10
6. Entorno de desarrollo	10
7. Gestión del cambio, incidencias y depuración.....	11
a. Gestión de incidencias internas	11
b. Gestión de incidencias externas.....	13
8. Gestión del código fuente	13
9. Gestión de la construcción e integración continua.....	15
10. Gestión de liberaciones, despliegue y entregas.....	16
11. Mapa de herramientas.....	17
12. Ejercicio de propuesta de cambio	17
13. Conclusiones y trabajo futuro	18
14. Lecciones aprendidas	18

1. Historial de versiones

Versión	Fecha	Realizado por
1.0	14/01/2018	Juan Carlos Cansino Suárez

2. Resumen

El problema que se ha tratado en este trabajo es el recuento de votos del sistema votaciones de Agora Voting, el recuento de los votos es parte fundamental del sistema de votaciones ya que es la forma principal de obtener los resultados de la votación.

La solución que hemos planteado es realizar una API la cual reciba del grupo de Almacenamiento de datos todos los datos de la votación (mediante un JSON) y realizamos un recuento de los votos en función de las diferentes opciones que posee la pregunta, así como otros métodos en los que podemos obtener la pregunta, las opciones de las preguntas entre otros.

Es una solución óptima ya que en función de los datos que recibimos del JSON es el recuento general que podemos realizar para obtener el resultado real de la votación.

3. Introducción y contexto

El principal objetivo de este trabajo es realizar el recuento de votos del sistema de votaciones de Agora Voting (proceso fundamental en un sistema de votación).

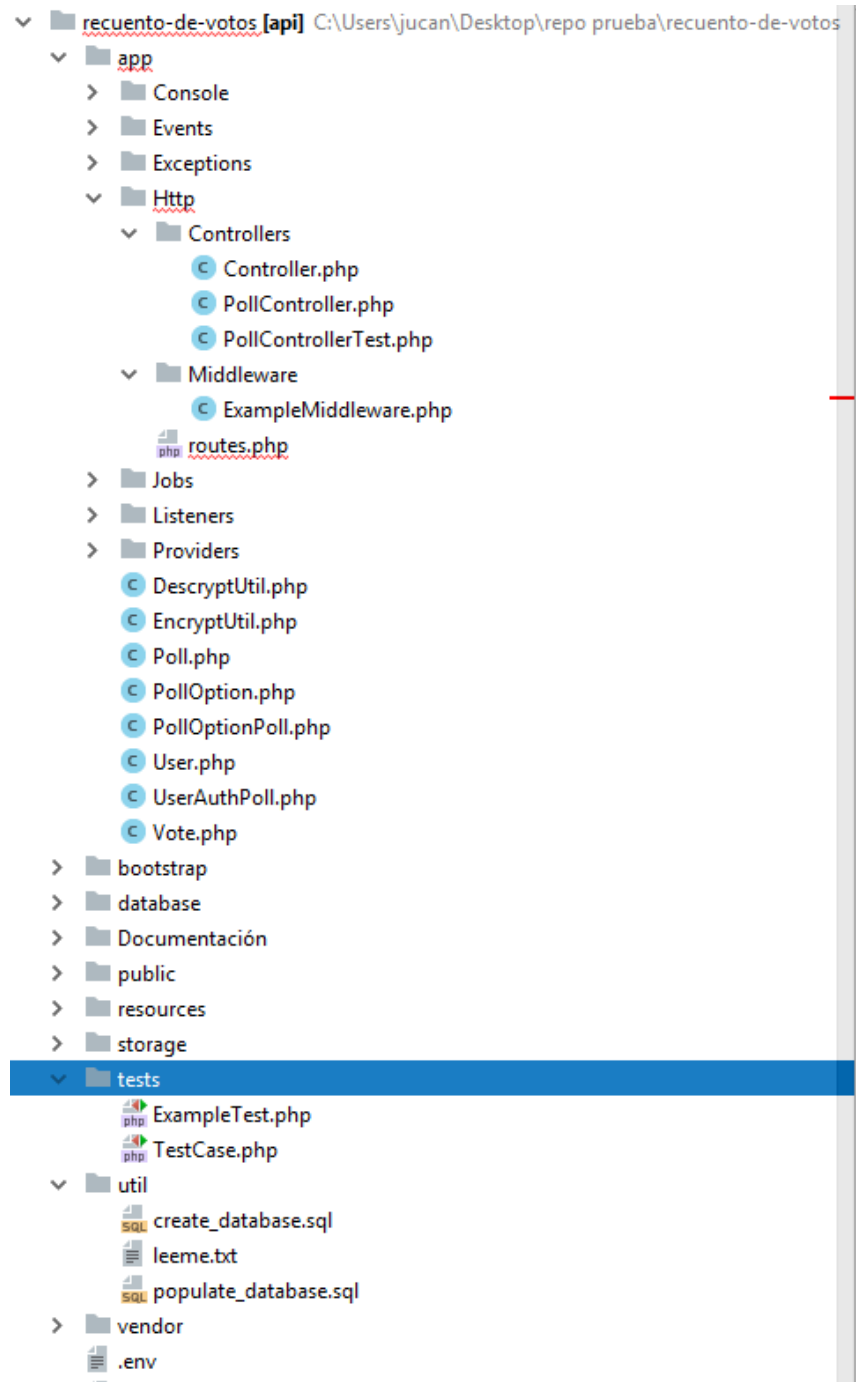
Para el desarrollo del proyecto software vamos a utilizar las herramientas posteriormente descritas usando el lenguaje PHP con PhpStorm y como herramientas para la gestión de código utilizaremos el repositorio de GitHub con la estructura por ramas más adelante detallada, así como para la gestión de las diferentes incidencias que nos puedan ir surgiendo utilizaremos los issues del propio GitHub.

En la planificación del proyecto se podría decir que todos los miembros son desarrolladores, aunque algunos han realizado más tareas de implementación que otros todos conocemos la implementación y funcionamiento del código.

4. Descripción del sistema

El subsistema de recuento de votos del sistema Agora Voting lo hemos realizado en lenguaje PHP y también posee una base de datos en MySQL:

- El proyecto en PHP está estructurado de la siguiente forma:



- Un método para encriptar (EncryptUtil.php) Se encuentra dentro de la carpeta app:

```
<?php

namespace App;

class EncryptUtil
{
    function __construct() {}

    function encrypt($textToEncrypt) {
        try{
            $method = 'aes-256-cbc';
            $clave = env( key: 'CLAVE');
            $iv = chr( ascii: 0x0 ) . chr( ascii: 0x0 ) . chr( ascii: 0x0 ) . chr( ascii: 0x0 ) . chr( ascii: 0x0 ) . chr( ascii: 0x0 );
            $encrypted = base64_encode(openssl_encrypt($textToEncrypt, $method, $clave, options: OPENSSESL_RAW_DATA,
        }catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en EncryptUtil", $e);
        }
        return $encrypted;
    }
}
```

- Un método para desencriptar los datos que nos llegan del JSON de Almacenamiento (DecryptUtil.php) Se encuentran dentro de la carpeta app:

```
<?php

namespace App;

class DecryptUtil
{
    function __construct() {}

    function decrypt($encrypted) {
        try{
            $method = 'aes-128-cbc';
            $clave = env( key: 'CLAVE');
            $texto_sin_base64 = base64_decode($encrypted);
            $iv = substr($texto_sin_base64, start: 0, length: 16);
            $decrypted = openssl_decrypt(substr($texto_sin_base64, start: 16, strlen($texto_sin_base64)), $method, $clave, options: OPENSSESL_RAW_DATA, $iv);
        }catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en DecryptUtil", $e);
        }
        return $decrypted;
    }
}
```

- Dentro del directorio app/Http encontramos el archivo routes.php que contiene las diferentes formas de llamar a nuestra API:

```
<?php

// ...

$app->get( uri: '/', function() use ($app) {
    return $app->welcome();
});

$app->get( uri: 'api/vote/{token_bd}/{id}/{auth}', action: 'PollController@getVotesByPoll');
$app->get( uri: 'api/test/vote/{token_bd}/{id}/{auth}', action: 'PollControllerTest@getVotesByPoll');

$app->get( uri: 'api/optionspoll/{token_bd}/{id}/{auth}', action: 'PollController@getOptionsByPoll');
$app->get( uri: 'api/test/optionspoll/{token_bd}/{id}/{auth}', action: 'PollControllerTest@getOptionsByPoll');
```

- Los dos métodos que implementa nuestra api se encuentran en app/http/controllers y son:
 - o getVotesByPoll(\$token_bd,\$token_votacion,\$token_pregunta) que devuelve todos los votos de una encuesta.

```
public function getVotesByPoll($token_bd,$token_votacion,$token_pregunta)
{
    if($token_votacion == null || $token_votacion== "" || $token_pregunta == null || $token_pregunta == ""){
        abort( code: 400, message: 'Bad request');
        error_log( message: "Error 400, alguna variable vacia o nula");
    }else{
        try{
            $url_api = env( key: 'URL_ALMACENAMIENTO');
            $url_api = $url_api . "/" . $token_bd . "/" . $token_votacion . "/" . $token_pregunta;

            $json = file_get_contents($url_api);
            $obj = json_decode($json);
            $decryptUtil = new DecryptUtil();
            $res = array();

            foreach($obj as $value){
                array_push($res,$decryptUtil->decrypt($value->token_respuesta));
            }
            $resFinal = array_count_values($res);
            $result = array("total_votes" => sizeof($res));

            $encryptUtil = new EncryptUtil();
            foreach($resFinal as $key=>$value){
                $result[$encryptUtil->encrypt($key)] = $value ;
            }

            return response()->json($result, status: 200);
        }catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en PollController",$e);
        }
    }
}
```

- `getOptionsByPoll($token_bd, $token_votacion, $token_pregunta)` que devuelve las opciones de una encuesta.

```
public function getOptionsByPoll($token_bd,$token_votacion,$token_pregunta)
{
    if($token_votacion == null || $token_votacion== "" || $token_pregunta == null || $token_pregunta == ""){
        abort( code: 400, message: 'Bad request');
        error_log( message: "Error 400, alguna variable vacia o nula");
    }else{
        try{
            $url_api = env( key: 'URL_ALMACENAMIENTO');
            $url_api = $url_api . "/" . $token_bd . "/" . $token_votacion . "/" . $token_pregunta;

            $json = file_get_contents($url_api);

            $obj = json_decode($json);
            $descriptUtil = new DescriptUtil();
            $res = array();

            foreach($obj as $value){
                array_push($res,$descriptUtil->descript($value->token_respuesta));
            }
            $resFinal = array_count_values($res);

            $encryptUtil = new EncryptUtil();
            $result = array('options');
            foreach($resFinal as $key=>$value){
                array_push($result,$encryptUtil->encrypt($key));
            }

            return response()->json($result, status: 200);
        }catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en PollController",$e);
        }
    }
}
```

- Los test implementados para comprobar dichos métodos se encuentran en app/http/controllers y son:

- o getVotesByPoll(\$token_bd, \$token_votacion, \$token_pregunta)

```
public function getVotesByPoll($token_bd,$token_votacion,$token_pregunta)
{
    if($token_votacion == null || $token_votacion== "" || $token_pregunta == null || $token_pregunta == "" || $token_bd == "" || $token_bd == null)
        abort( code: 400, message: 'Bad request');

    }else{

        $url_api = env( key: 'URL_ALMACENAMIENTO');
        $url_api = $url_api . "/" . $token_bd . "/" . $token_votacion . "/" . $token_pregunta;

        // $json = file_get_contents($url_api);
        $json = <<<EOF

            {
                "1": {
                    "id": "1",
                    "token_usuario": "1",
                    "token_pregunta": "1",
                    "token_respuesta": "NLpGXg7r60dD9jyx1+o6O5YAk6eAfo9MhJ+JJY1ScgU="
                },
                "2": {
                    "id": "1",
                    "token_usuario": "1",
                    "token_pregunta": "1",
                    "token_respuesta": "4xXyB9FapWewjQjNI4SZGwqyTEPKH7b/hSxIsCQhqrw="
                },
                "3": {
                    "id": "1",
                    "token_usuario": "1",
                    "token_pregunta": "1",
                    "token_respuesta": "NLpGXg7r60dD9jyx1+o6O5YAk6eAfo9MhJ+JJY1ScgU="
                }
            }
        ;

        $obj = json_decode($json);
        $descriptUtil = new DescriptUtil();
        $res = array();

        foreach($obj as $value){
            array_push($res,$descriptUtil->descript($value->token_respuesta));
        }

        $resFinal = array_count_values($res);
        $result = array("total_votes" => sizeof($res));

        $encryptUtil = new EncryptUtil();
        foreach($resFinal as $key=>$value){
            $result[$encryptUtil->encrypt($key)] = $value ;
        }
    }
}
```


- `getOptionsByPoll($token_bd, $token_votacion, $token_pregunta)`

```
public function getOptionsByPoll($token_bd,$token_votacion,$token_pregunta)
{
    if($token_votacion == null || $token_votacion== "" || $token_pregunta == null || $token_pregunta == "" || $token_bd == "" || $token_bd
        abort( code: 400, message: 'Bad request');

    }else{

        $url_api = env( key: 'URL_ALMACENAMIENTO');
        $url_api = $url_api . "/" . $token_bd . "/" . $token_votacion . "/" . $token_pregunta;

        //$json = file_get_contents($url_api);
        $json = <<<EOF

        {
            "1": {
                "id": "1",
                "token_usuario": "1",
                "token_pregunta": "1",
                "token_respuesta": "NLpGXg7r60dD9jyx1+o6O5YAk6eAfo9MhJ+JJYlScgU="
            },
            "2": {
                "id": "1",
                "token_usuario": "1",
                "token_pregunta": "1",
                "token_respuesta": "4xXyB9FAPwewjQjNI4SZGwqyTEPKH7b/hSxIsCQhgrw="
            },
            "3": {
                "id": "1",
                "token_usuario": "1",
                "token_pregunta": "1",
                "token_respuesta": "NLpGXg7r60dD9jyx1+o6O5YAk6eAfo9MhJ+JJYlScgU="
            }
        }

    };

    $obj = json_decode($json);
    $descriptUtil = new DescriptUtil();
    $res = array();

    foreach($obj as $value){
        array_push($res,$descriptUtil->descript($value->token_respuesta));
    }
    $resFinal = array_count_values($res);

    $encryptUtil = new EncryptUtil();
    $result = array('options');
```

5. Planificación del proyecto

Como grupo de trabajo queremos tener una planificación del proyecto equitativa con sus diferentes roles bien definidos, aunque, sin embargo, queremos que todos los miembros del equipo sean desarrolladores para que todos conozcamos la implementación del código.

La siguiente tabla nos muestra las diferentes tareas y a que persona del grupo con su rol está asignada:

Apellidos, Nombre	Rol	Implementación	Documentación
Barrera Roldán, Antonio Jesús.	Desarrollador.	Configuración del proyecto	
Cansino Suárez, Juan Carlos.	Desarrollador. Coordinador.	Encriptación y desencriptación	Documentación del proyecto.
Martín Galván, María Inmaculada.	Desarrollador.	Funciones API	Realizar wiki.
Rodríguez López, Daniel.	Desarrollador. Gestor de incidencias.	Integración del subsistema	Creación y gestión de issues de GitHub
Sánchez Rodríguez, Mario.	Desarrollador.	Funciones API	Diario del grupo.
Soldado Caro, Alfonso.	Desarrollador.	Base de datos	Actualización de la wiki del grupo.

Esta tabla es un resumen de lo que podemos ver en las issues creadas en nuestro repositorio de GitHub en la cual están todas las tareas creadas y asignadas a sus respectivos miembros del grupo como muestra la tabla.

6. Entorno de desarrollo

El lenguaje que hemos usado es PHP y el entorno de desarrollo es PhpStorm versión: 2017.2.4. Para la instalación de PhpStorm accedemos a la siguiente dirección: <https://www.jetbrains.com/phpstorm/> y hacemos click en download now y una vez descargado el archivo procedemos a instalarlo con las recomendaciones establecidas por el fabricante.

El sistema de gestión de bibliotecas que utilizamos es composer y concretamente usamos Eloquent que es un ORM de laravel para la gestión de la base de datos.

7. Gestión del cambio, incidencias y depuración

Cuando se requiera realizar una nueva funcionalidad la persona encargada de la gestión de dichas tareas ha de utilizar un método para ponerse en contacto con el miembro del grupo al que se le haya asignado esa tarea e informarle de que ha de realizarla describiéndole el resultado que queremos obtener y que una vez esté completada la funcionalidad se cierre la incidencia y esa parte del trabajo quede como realizada para que los demás grupos del trabajo tengan constancia de ello.

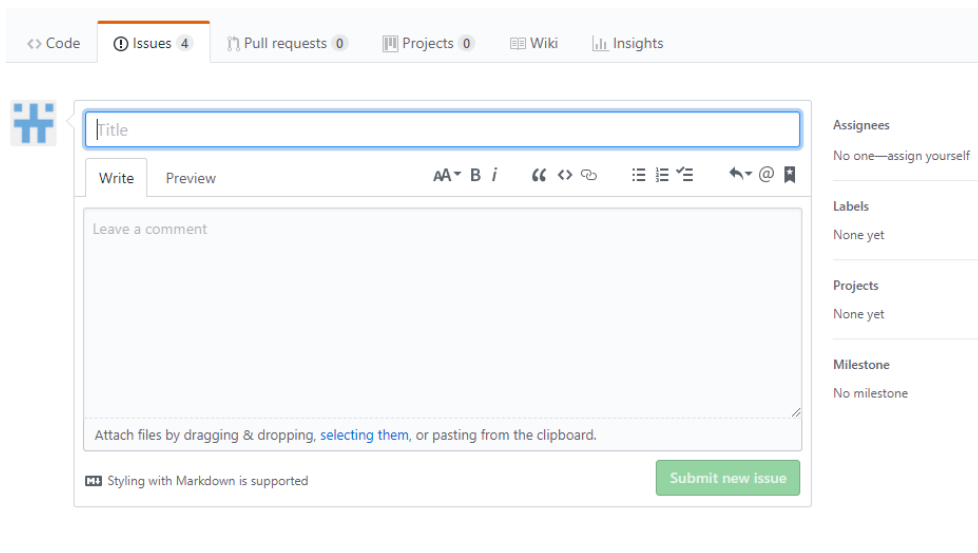
Como equipo necesitamos un sistema de gestión de incidencias el cual nos permita abrir, modificar y cerrar una incidencia que tenga un título y un cuerpo para realizar la descripción de la misma. También necesitaremos que dicha incidencia se le pueda asignar a los diferentes miembros del grupo, así como crearle y asignarle diferentes etiquetas para su mejor comprensión.

Además, queremos que el cierre de las incidencias pueda automatizarse mediante la realización de los commits para ahorrar tiempo y tener una mejor usabilidad.

Para la gestión de incidencias hemos utilizado las issues de GitHub.

a. Gestión de incidencias internas

Para crear una issue dentro de nuestro proyecto nos dirigimos a nuestro repositorio de GitHub y hacemos click en el apartado llamado issues, dentro de este hacemos click en el apartado de New issue lo cual nos redirige a una página en la que podemos crear y editar todos los aspectos de una issue.



- Assignees: le asignamos al encargado de realizar el issue y la gente relacionada con él.
- Labels: contiene unas etiquetas que le podemos asignar a los diferentes issues creadas por nosotros a nuestro gusto. Un ejemplo son las etiquetas Importante, muy importante, por comenzar.... (pueden editarse al gusto personal).
- Projects: no hemos creado ningún proyecto.
- Milestone: dependiendo de para que milestone tenga que estar terminado el issue se le asigna. Previamente hemos de haber creado el milestone.

- Título del issue y una descripción en la que seguimos la plantilla proporcionada por integración.

Una de nuestras issues perfectamente creada quedaría:

Programar recibir API #6

Open jucansu opened this issue on 11 Dec 2017 · 0 comments

Member + 👤 ✎

jucansu commented on 11 Dec 2017

Prioridad: Alta

Se ha de implementar los métodos que se encarguen de procesar los JSON que recibiremos de las demás API.
Principalmente necesitamos recibir una lista de todos los votos almacenados por el subsistema de Almacenamiento de votos.

jucansu added **enhancement** **requirement** labels on 11 Dec 2017

jucansu added this to the **Milestone 3** milestone on 11 Dec 2017

jucansu assigned **marmargal** and **MarioShez** on 11 Dec 2017

jucansu added the **por comenzar** label 13 days ago

Assignees

- marmargal
- MarioShez

Labels

- enhancement
- por comenzar
- requirement

Projects

None yet

Milestone

Milestone 4

Notifications

Cerramos issues automáticamente al realizar un commit colocando al final de este la palabra Closes #número de issue. En el cuerpo del commit hay que explicar que se ha realizado y porque se cierra la issue. Queda de tal forma:

Programar funciones API #5

Closed jucansu opened this issue on 11 Dec 2017 · 0 comments

Member + 👤 ✎

jucansu commented on 11 Dec 2017

Prioridad: Alta

Hemos de implementar las funciones que se encarguen de crear los JSON que le proporcionará nuestra API a los subsistemas que necesiten de ella.
Necesitaremos la API de Almacenamiento de datos.

jucansu added **enhancement** **requirement** labels on 11 Dec 2017

jucansu added this to the **Milestone 3** milestone on 11 Dec 2017

jucansu assigned **marmargal** and **MarioShez** on 11 Dec 2017

MarioShez closed this in [599910d](#) 23 days ago

Assignees

- marmargal
- MarioShez

Labels

- enhancement
- requirement

Projects

None yet

Milestone

Milestone 3

Notifications

b. Gestión de incidencias externas

Para abrirle una incidencia a otro grupo nos dirigimos a su repositorio de GitHub y mediante el procedimiento anterior abrimos una issue en la cual solo escribimos el título y una descripción detallada del problema encontrado, dejando así que el encargado de las issues del grupo en cuestión asigne etiquetas y empleados a los que considere necesarios.

Ejemplo de una issue abierta a Almacenamiento.

The screenshot shows a GitHub issue page for 'Encriptación de datos. #4'. The issue was opened by 'jucansu' 28 days ago and has 2 comments. The issue is marked as 'Open' with a green icon. The first comment, from 'jucansu' (Member), is dated 28 days ago and contains the text: 'Prioridad: Alta' and 'Se necesita saber que método de cifrado de datos va a realizar el equipo de Almacenamiento, para saber como se podrá descifrar los datos necesarios para realizar el recuento de las votaciones.' The second comment, from 'pedrosr14' (Member), is dated 23 days ago and contains the text: 'Hola, ante todo perdón por la tardanza.' and 'Aún estamos en pruebas viendo qué método vamos a usar porque estábamos usando implementando un método de cifrado que no permitía el descifrado después fácilmente. Ahora mismo estamos tratando de implementar el cifrado AES, cuando lo tengamos se os proporcionará el método y la clave pública que se necesita para descifrar.' The right sidebar shows 'Assignees' (No one assigned), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Notifications' (Unsubscribe).

8. Gestión del código fuente

Como grupo necesitamos un sistema de gestión de código para poder llevar todo nuestro trabajo en paralelo y trabajar remotamente. Un repositorio en el que cada uno podamos tener nuestro propio espacio en el que realizar las modificaciones necesarias y que una vez que una funcionalidad esté implementada lo pasemos a un repositorio común en el que estén las funcionalidades terminadas y testeadas implementadas por todos los miembros del equipo.

También necesitaremos un nuevo espacio en el que subamos todo el proyecto implementado por completo y con todo en funcionamiento que usaremos únicamente para el despliegue del subsistema.

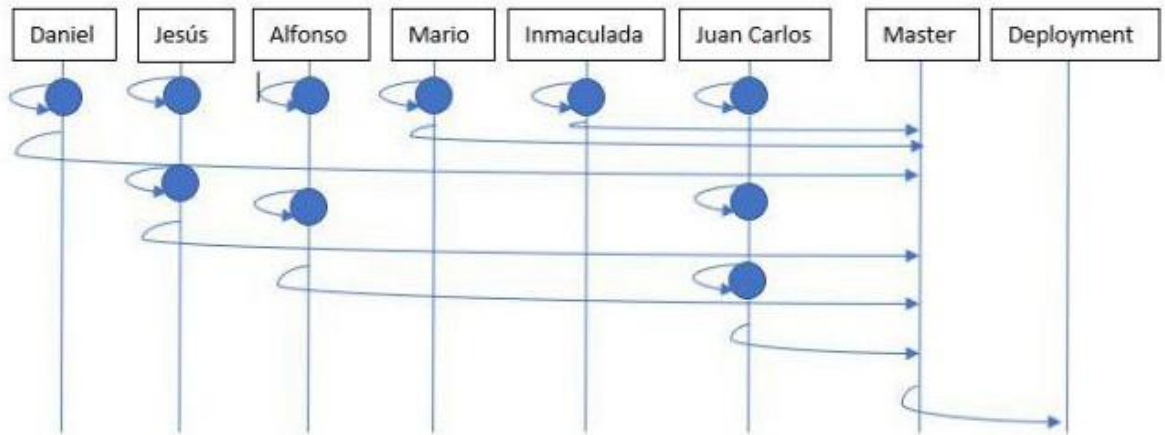
Para la gestión de código utilizamos GitHub como repositorio. Para saber cuándo y cómo hacer commits seguimos las pautas proporcionadas en la wiki por el equipo de integración:

https://1984.lsi.us.es/wiki-egc/index.php/Equipo_de_integraci%C3%B3n_general_-_17_18_-_G2

La estructura por ramas que utilizamos está formada por una rama personal para cada uno de los miembros del grupo con el nombre de dicha persona, en la cual cada uno realizará sus commits personales.

Una rama master a la que se podrá hacer commit desde una rama personal únicamente cuando una funcionalidad esté totalmente acabada y comprobada.

Una rama Deployment a la cual se podrá hacer commit desde master cuando ya este toda la funcionalidad por completo funcionando, se podría decir que es una rama para el despliegue de la aplicación.



9. Gestión de la construcción e integración continua

Para la construcción e integración continua de nuestro proyecto hemos usado Travis según nos ha indicado el grupo de integración, además de usar la máquina virtual para trabajar con la API en local.

Usamos Travis siguiendo el procedimiento indicado por integración, así que hemos añadido a nuestro proyecto en GitHub un archivo .travis.yml siguiendo la plantilla proporcionada. El fichero quedaría de la siguiente forma:

```
1  language: php
2
3  php:
4    - 7.1
5
6  sudo: required
7
8  services:
9    - mysql
10   - docker
11
12  env:
13    - Laravel 5.5
14
15  before_install:
16    # INSERTAR AQUÍ LÍNEA PARA DESENCRIPTAR CLAVE DE ACCESO AL SERVIDOR DE DESPLIEGUE,
17    #   PROPORCIONADA POR INTEGRACIÓN
18
19    # INICIO DE DESPLIEGUE EN UN CONTENEDOR DE LA BASE DE DATOS DEL SISTEMA DE
20    #   VOTACIONES
21    - openssl aes-256-cbc -K $encrypted_80d99b7f2682_key -iv $encrypted_80d99b7f2682_iv -in deploy.enc -out deploy -d
22    - mkdir mysql_build
23    - cd mysql_build
24    - curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/docker/dockerfiles/mysql/init.sql
25    - curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/docker/dockerfiles/mysql/Dockerfile
26    - docker build -t egc/mysql .
27    - docker network create --subnet=172.18.0.0/16 dev
28    - docker run -d --network=dev --ip=172.18.2.2 -e MYSQL_ROOT_PASSWORD=nothing egc/mysql
29    - cd ..
30    # FIN DEL DESPLIEGUE, YA SE PUEDEN REALIZAR TESTS CON LA BD
31    # mysql -e 'CREATE DATABASE api;'
32    # mysql -uroot api < create_database.sql
33
34  before_deploy:
35    - chmod 600 deploy && mv deploy ~/.ssh/id_rsa
36    - curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/tools/deploy.sh
37    # ADMINISTRACIÓN DE VOTACIONES: DESCOMENTAR SIGUIENTE LÍNEA
38    # - scp -o StrictHostKeyChecking=no target/*.war deploy@egc.duckdns.org:g2/tmp/ROOT.war
39
40  deploy:
41    skip_cleanup: true
42    provider: script
43    script: ssh -o StrictHostKeyChecking=no deploy@egc.duckdns.org 'bash -s' < deploy.sh recuento # INSERTAR NOMBRE DEL SUBSISTEMA
44    on:
45      branch: master
46
47  install:
48    - composer install
49    - php artisan serve
50
```

Tendríamos el subsistema integrado:

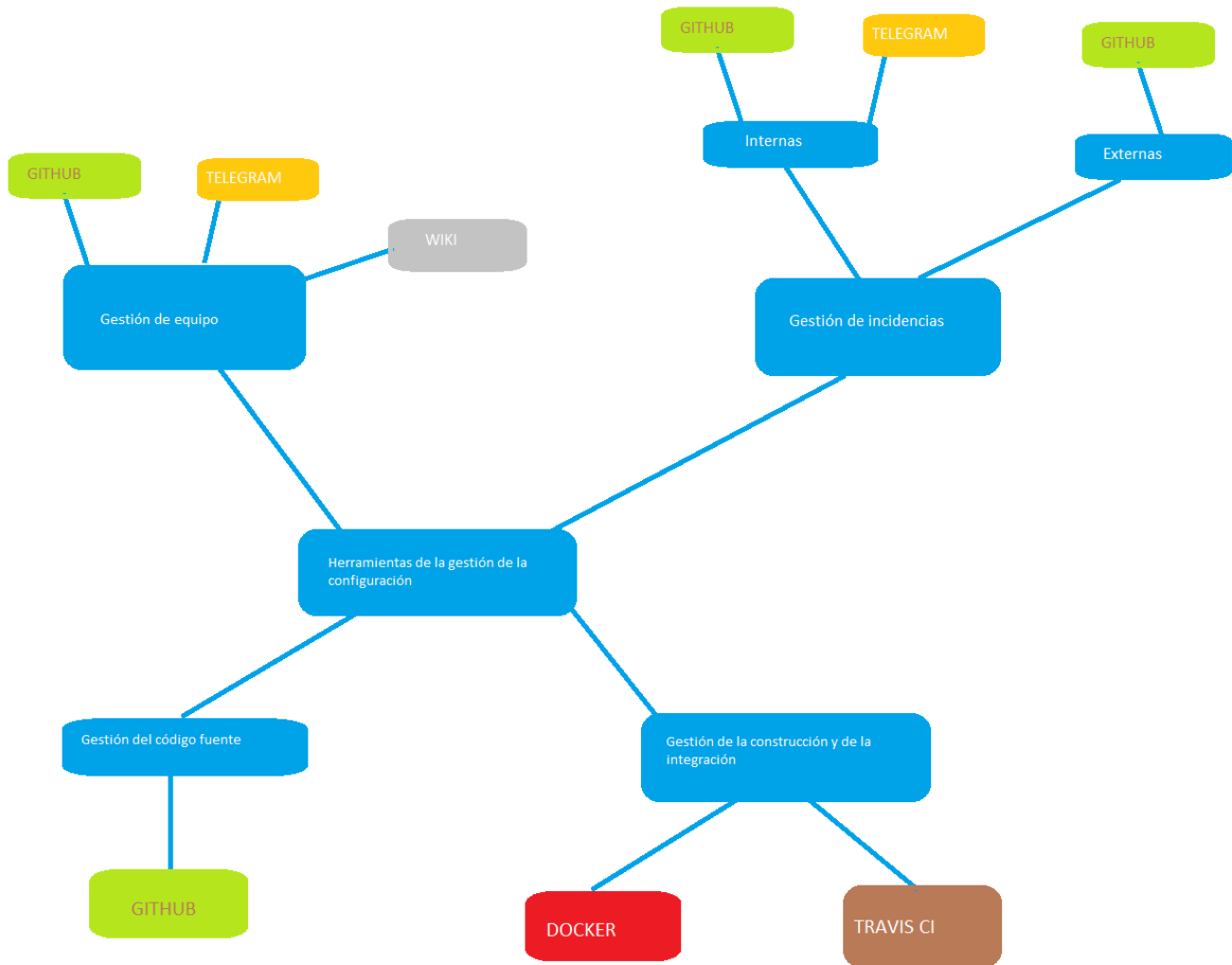


10. Gestión de liberaciones, despliegue y entregas

El proceso que hemos utilizado para la entrega es el siguiente:

- Tenemos todo el proyecto junto con toda la documentación en nuestro repositorio oficial de GitHub:
- En la sección de descripción del portal de opera del grupo añadimos un enlace a dicho repositorio el cual contiene todo el entregable.
- La entrega hay que realizarla antes de las 23:59 del 14/01/2018.

11. Mapa de herramientas



12. Ejercicio de propuesta de cambio

Planteamos como ejercicio de propuesta de cambio para nuestro proyecto que se implemente una nueva llamada para la API “getWinnerPoll” en la cual se realice un recuento de la votación y devuelva la opción con más votos obtenidos.

13. Conclusiones y trabajo futuro

Finalmente hemos conseguido implementar la funcionalidad de recomtar los votos, pero tan solo hemos realizado un recuento, el recuento general de una votación en el que se devuelven las diferentes opciones con el número de votos obtenidos por cada una. No se ha podido realizar otro tipo de recuento como por ejemplo por ciudades por dos motivos: porque almacenamiento solo nos proporcionaba el id de una pregunta y el id de las opciones y por falta de tiempo.

No se ha realizado una buena planificación inicial del proyecto lo que nos ha llevado a ir atrasados en las entregas y no poder cumplir con todos los objetivos estimados.

14. Lecciones aprendidas

Como lecciones aprendidas de este proyecto hemos sacado que PHP no es el lenguaje idóneo para realizar este trabajo ya que trae tras de sí mucha librería y problemas en la implementación, hubiese sido más fácil la implementación en Python.

Una buena planificación inicial del proyecto y dividir bien los roles del equipo es esencial para ahorrarnos tiempo a lo largo del trabajo, así como conocer el entorno de trabajo con el que vamos a trabajar.