

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
4.1 Automatización de las pruebas	10
6. Planificación del proyecto	13
7. Entorno de desarrollo	14
8. Gestión del cambio, incidencias y depuración	14
1. Gestión de incidencias internas	15
2. Gestión de incidencias externas.....	17
9. Gestión del código fuente	17
10. Gestión de la construcción e integración continua.....	19
11. Gestión de liberaciones, despliegue y entregas.....	21
12. Mapa de herramientas.....	22
13. Ejercicio de propuesta de cambio	22
14. Conclusiones y trabajo futuro	23
15. Lecciones aprendidas	23

1. Historial de versiones

Versión	Fecha	Realizado por
1.0	14/01/2018	Juan Carlos Cansino Suárez
2.0	06/02/2018	Mario Sánchez Rodríguez

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).

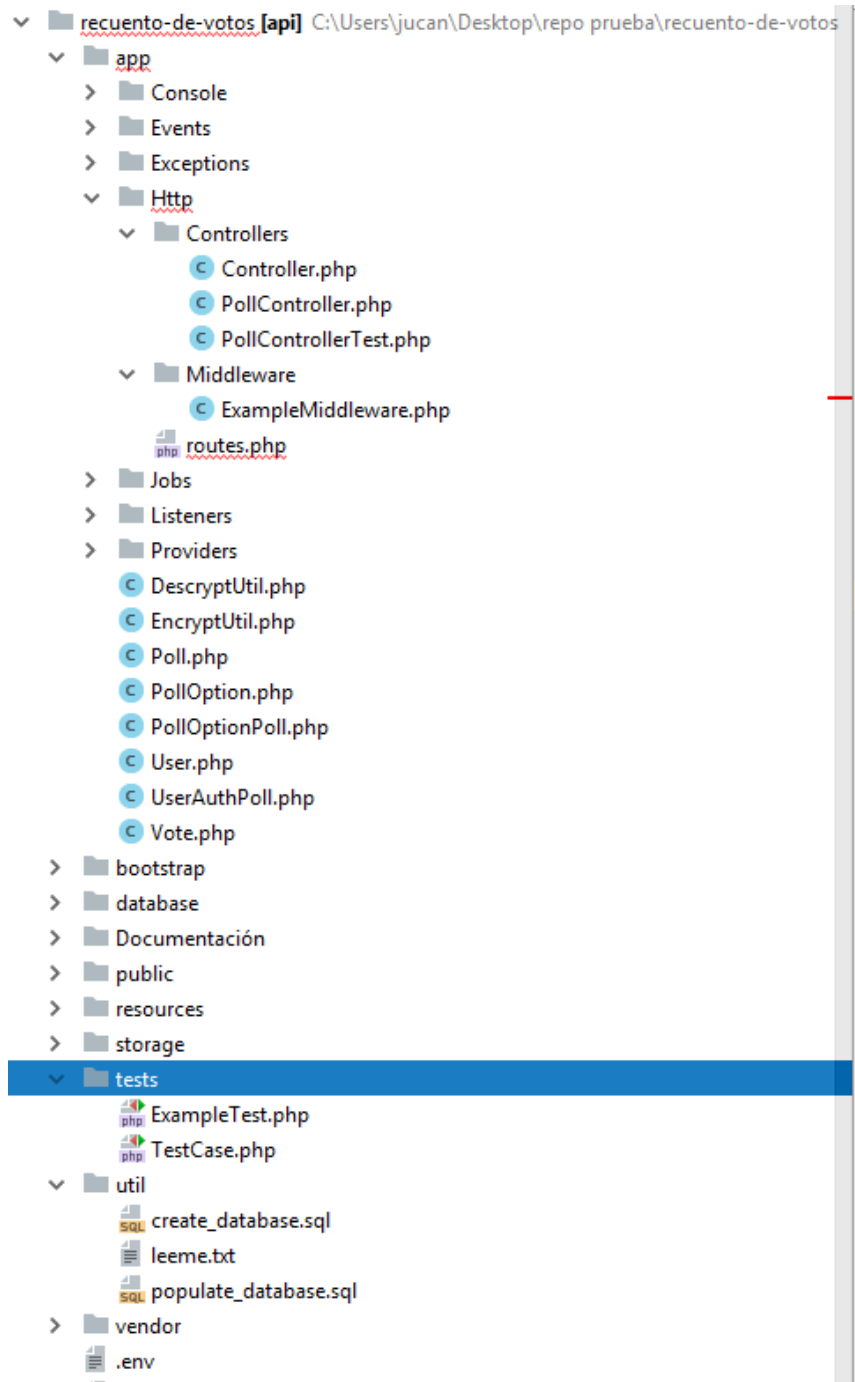
Para el desarrollo del proyecto software vamos a utilizar las herramientas posteriormente descritas usando el lenguaje PHP con PhpStorm y como herramientas para la gestión de código utilizaremos el repositorio de GitHub con la estructura por ramas más adelante detallada, así como para la gestión de las 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:



1. 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: OPENSSSL_RAW_DATA,
        )catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en EncryptUtil", $e);
        }
        return $encrypted;
    }
}
```

2. 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: OPENSSSL_RAW_DATA, $iv);
        }catch(Exception $e){
            error_log( message: "Se ha producido un error no controlado en DecryptUtil", $e);
        }
        return $decrypted;
    }
}
```

3. 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');
```

4. Los dos métodos que implementa nuestra api se encuentran en app/http/controllers y son:

- `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);
            $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 ;
            }

            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);
            $decryptUtil = new DecryptUtil();
            $res = array();

            foreach($obj as $value){
                array_push($res,$decryptUtil->decrypt($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);
        }
    }
}
```

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

1. 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": "4xXyB9FpWewjQjNI4SZGWqyTEPKH7b/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 ;
        }
    }
}
```


2. 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');
```

4.1 Automatización de las pruebas

La automatización de los test se han implementado en app/test, en las clases:

- DecryptTest.php

```

<?php
use App\DecryptUtil;

class DecryptTest extends TestCase
{
    public function testEmptyTextDecryptedNegativa()
    {
        print "Test Negativo";
        print "Test : Texto ha desencryptar vacio";
        $decryptUtil = new DecryptUtil();
        $this->assertEquals(null,$decryptUtil->decrypt( encrypted: "", clave: "Almacen de votos"),"El texto ha desencryptar no puede ser vacio");
    }

    public function testEmptyPassowrdDecryptedNegativa()
    {
        print "Test Negativo";
        print "Test : Password ha desencryptar vacio";
        $decryptUtil = new DecryptUtil();
        $this->assertEquals(null,$decryptUtil->decrypt( encrypted: "NLpGXg7r60dD9jyx1+o605YAk6eAfo9MhJ+JJYlScgU=", clave: ""),"La password para desencryptar no puede estar vacia");
    }

    public function testEmptyTextDecryptedPositiva()
    {
        print "Test Positivo";
        print "Test : Texto ha desencryptar vacio";
        $decryptUtil = new DecryptUtil();
        $res = $decryptUtil->decrypt( encrypted: "NLpGXg7r60dD9jyx1+o605YAk6eAfo9MhJ+JJYlScgU=", clave: "Almacen de votos");
        $this->assertTrue($res != null,"Test correcto");
    }

    public function testEncryptPositiva()
    {
        print "Test Positivo";
        print "Test : Texto ha desencryptar vacio";
        $decryptUtil = new DecryptUtil();
        $res = $decryptUtil->decrypt( encrypted: "NLpGXg7r60dD9jyx1+o605YAk6eAfo9MhJ+JJYlScgU=", clave: "Almacen de votos");
        $this->assertTrue($res == 3,"Test correcto");
    }
}

```

- EncryptTest.php:

```

<?php
use App\EncryptUtil;

class EncryptTest extends TestCase
{
    public function testEmptyTextDecryptedNegativa()
    {
        print "Test Negativo";
        print "Test : Texto ha encryptar vacio";
        $encryptUtil = new EncryptUtil();
        $this->assertEquals(null,$encryptUtil->encrypt( textToEncrypt: "", clave: "Almacen de votos"),"El texto ha encryptar no puede ser vacio");
    }

    public function testEmptyPassowrdDecryptedNegativa()
    {
        print "Test Negativo";
        print "Test : Password para encryptar vacio";
        $encryptUtil = new EncryptUtil();
        $this->assertEquals(null,$encryptUtil->encrypt( textToEncrypt: "Texto de prueba", clave: ""),"La password para encryptar no puede estar vacia");
    }

    public function testNotEmptyTextAndNotEmptyDecryptedPositiva()
    {
        print "Test Positivo";
        print "Test : Texto ha encryptar relleno y password rellena";
        $encryptUtil = new EncryptUtil();
        $res = $encryptUtil->encrypt( textToEncrypt: "NLpGXg7r60dD9jyx1+o605YAk6eAfo9MhJ+JJYlScgU=", clave: "Almacen de votos");
        $this->assertTrue($res != null,"Test correcto");
    }
}

```

Ambas clases utilizan métodos similares ya que en ambos métodos se utiliza el método de descifrado y cifrado AES-256.

- testEmptyTextDescryptedNegativa() comprueba que el parámetro a desencriptar no esté vacío.
 - testEmptyPasswordDescryptedNegativa() comprueba que la contraseña no esté vacía.
 - testEmptyTextDescryptedPositiva() comprueba que el resultado de la desencriptación no sea null.
 - testEncryptPositiva() comprueba que el resultado de la desencriptación sea el esperado.
- UserAuthPoll.php: Los métodos implementados en esta clase testearán las dos llamadas de nuestra API (getOptionByPoll y getVotesByPoll).
1. testgetVotesByPollPositivoX() comprueba que introduciendo todos los valores correctamente no devuelva ningún fallo inesperado.

```
public function testgetVotesByPollPositivoX()
{
    print "Test Negativo";
    print "Test : Check parametros getVotesByPoll";
    $pollController = new PollControllerTest();

    $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": "4xKyB9FpWewjQjNI4SZGwqyTEPKH7b/hSxIsCQhqrw="
            },
            "3": {
                "id": "1",
                "token_usuario": "1",
                "token_pregunta": "1",
                "token_respuesta": "NLpGXg7r60dD9jyx1+o6O5YAk6eAfo9MhJ+JJYlScgU="
            }
        }
    EOF;

    $res = $pollController->getVotesByPoll( token_bd: 1, token_votacion: 2, token_pregunta: 1,$json);
    $this->assertNotEquals(-1,$res,"El parametro token_votacion esta vacio o no se ha encontrado");
}
```

2. testgetVotesByPollNegativoX() comprueba que si hay algún parámetro vacío salte un error. Este método se implementa tres veces más. Cada método testeará que un cierto atributo no esté vacío.

```
public function testgetVotesByPollNegativo5()
{
    print "Test Negativo";
    print "Test : Check parametros getOptionsByPoll";
    $pollController = new PollControllerTest();

    $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="
        }
    }
EOF;
```

- testgetVotesByPollNegativoX() comprueba que el formato del JSON sea el correcto.

```
public function testgetVotesByPollNegativo8()
{
    print "Test Negativo";
    print "Test : Check json";

    $pollController = new PollControllerTest();

    $json = <<<EOF
    {
        "1": {
            "id": "1",
            "token_usuario": "1",
            "token_pregunta": "1",
            "token_respuesta": "NLpGXg7r60dD9jyx1+o6Q5YAk6eAfo9MhJ+JJY1ScgU="
        },
        "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+o6Q5YAk6eAfo9MhJ+JJY1ScgU="
        }
    }
    EOF;

    $res = $pollController->getVotesByPoll( token_bd: 1, token_votacion: 2, token_pregunta: 1,$json);
    $this->assertEquals(-1,$res,"El json se encuentra mal formado");

}
```

6. 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 desenscriptación	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.

7. 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.

8. 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.

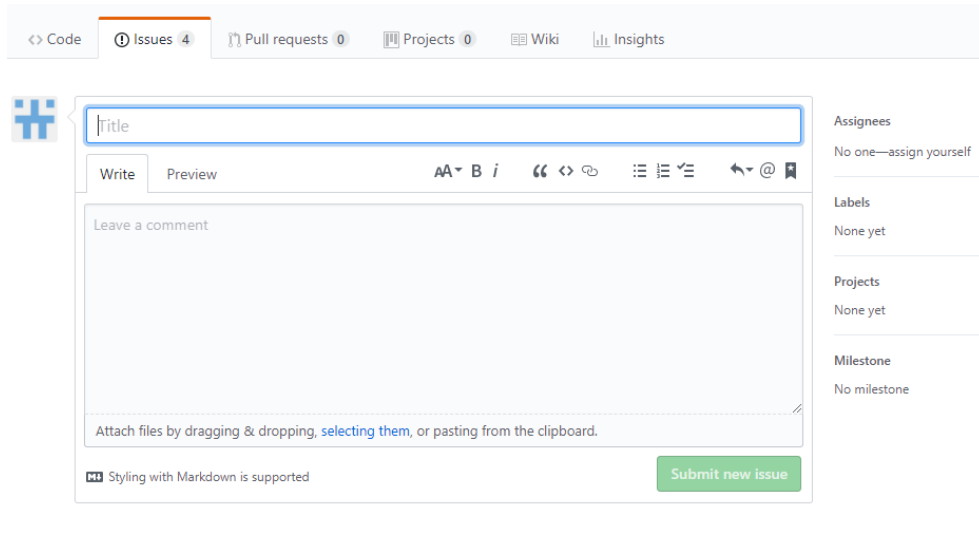
Como equipo necesitamos un sistema de gestión de incidencias el cual nos permita abrir, modificar y cerrar una incidencia que tenga un título y un cuerpo para realizar la descripción de la misma. También necesitaremos que dicha incidencia se le pueda asignar a los diferentes miembros del grupo, así como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

1. 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

[Edit](#)

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



jucansu commented on 11 Dec 2017

Member + 🗨️ ✎️

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

[Edit](#)

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



jucansu commented on 11 Dec 2017

Member + 🗨️ ✎️

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

2. 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.

Encriptación de datos. #4

Open jucansu opened this issue 28 days ago · 2 comments

jucansu commented 28 days ago

Member

+ 🗨️ ✎️

Prioridad: Alta

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.

pedrosr14 commented 23 days ago

Member

+ 🗨️ ✎️

Hola, ante todo perdón por la tardanza.

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.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

🔊 Unsubsc

9. 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:

- El formato general de los commits parte de la siguiente:
 - **feat** (new feature for the user, not a new feature for build script)
 - **fix** (bug fix for the user, not a fix to a build script)
 - **docs** (changes to the documentation)
 - **style** (formatting, missing semi colons, etc; no production code change)
 - **refactor** (refactoring production code, eg. renaming a variable)
 - **test** (adding missing tests, refactoring tests; no production code change)
 - **chore** (updating grunt tasks etc; no production code change)
- A continuación se detalla un ejemplo:

Título del commit: fix: redirección errónea tras emitir voto

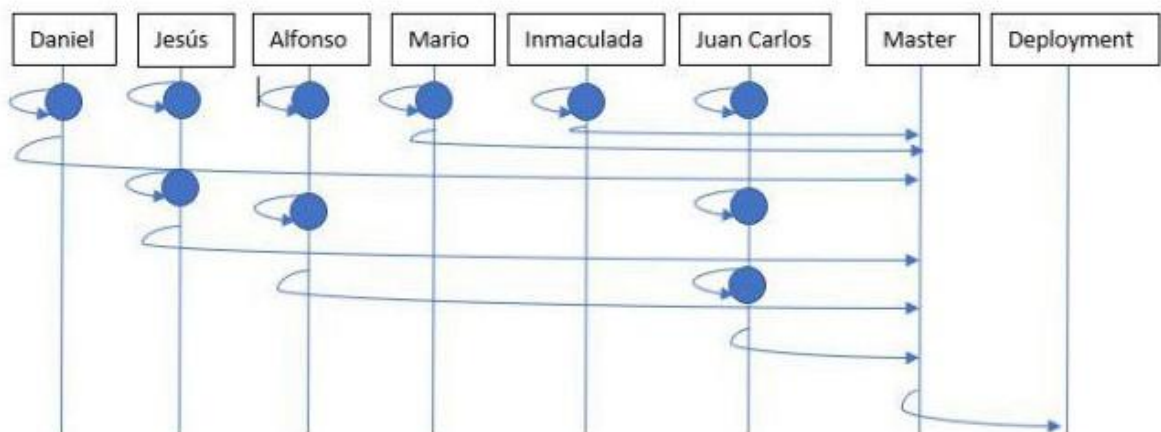
Cuerpo del commit: Después de que el usuario emitiese su voto este era redireccionado a una URL no existente. Ahora el usuario es redireccionado al panel principal.

Pie del commit: Closes #<número de la incidencia en GitHub>

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.



10. 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. A continuación se describirán las direcciones IPs y puertos disponibles en la máquina virtual:

Subsistema	Dirección IP	Puerto
BD Wordpress	172.18.2.1	No publicado al exterior
BD Sistema de votación	172.18.2.2	3306
Wordpress (Portal Jornadas)	172.18.2.5	50000
Portal votaciones	172.18.2.10	50010
Autenticación	172.18.2.20	50020
Administración de censos	172.18.2.30	50030
Administración de votaciones	172.18.2.40	50040
Almacenamiento de votos	172.18.2.50	50050
Cabina de votaciones	172.18.2.60	50060
Recuento	172.18.2.70	50070
Cabina Telegram	No	No

- Para acceder a algún servicio la URL consiste en `http://localhost:<puerto>`.
Ej: `http://localhost:50000` para acceder al portal wordpress.
- Para comunicarse con la API de algún subsistema es necesario especificar la dirección IP de dicho subsistema.
- Durante el desarrollo en local se pueden hacer peticiones a localhost gracias al mapeo de puertos entre el sistema anfitrión y la máquina virtual.
- No hay nombres de dominio para los subsistemas, puesto que se encuentran en una red privada y por motivos de seguridad sus servicios no son expuestos al exterior. Cabe recordar el principal consumidor de las APIs de los subsistemas será el portal web del sistema de votaciones.
- Aunque la BD del sistema de votaciones tenga un puerto publicado en Docker, se necesita hacer un mapeo en la máquina virtual. En [esta guía](#) hay información de cómo hacerlo. Basta con establecer el puerto anfitrión y el puerto invitado a 3306.

Para conseguir un entorno de pruebas lo más parecido al de producción (pre-producción), se han creado varios scripts que automatizan todos los procesos de despliegue, tal y como expresa el equipo de integración. Para acceder a ellos basta con ejecutar lo siguiente:

```
cd ~/scripts/integracion/tools
ls # El comando "ls" lista los ficheros existentes en el
    directorio actual.
```

Entre los ficheros de la carpeta tools se encuentra el script de inicio general "start.sh". Este script descarga los repositorios de los distintos subsistemas en la rama especificada por el

usuario e inicia un contenedor por cada subsistema con el código descargado. Para iniciar el script escribiremos el siguiente comando:

```
bash ~/scripts/integracion/tools/start.sh
bash start.sh # En caso de estar situados dentro de la carpeta
"tools".
```

Este script realiza las mismas tareas que el script de inicio general, pero con la diferencia de que permite iniciar un contenedor dentro de la máquina virtual con el código modificado en tu máquina local para realizar diversas pruebas. Los pasos a realizar para ejecutar este script son los siguientes:

1. Comprobar que en el código se hacen peticiones a la dirección IP de cada subsistema.
2. Realizar una copia del directorio raíz del repositorio. Con esto nos referimos a la carpeta resultante de hacer un "clone" del repositorio, no puede ser una carpeta de dentro del repositorio como "src" o derivados. Esta copia debe guardarse en el directorio compartido entre el sistema anfitrión y la máquina virtual.
3. Ejecutar el script en la máquina virtual mediante el siguiente comando:

```
bash ~/scripts/integracion/tools/deploy-shared.sh
bash deploy-shared.sh # Suponiendo que estamos situados dentro del directorio "tools"
```

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

```
language: php

php:
  - 7.1

sudo: required

services:
  - mysql
  - docker

env:
  - Laravel 5.5

before_install:
  # INSERTAR AQUÍ LÍNEA PARA DESENCRIPTAR CLAVE DE ACCESO AL SERVIDOR DE DESPLIEGUE,
  # PROPORCIONADA POR INTEGRACIÓN

  # INICIO DE DESPLIEGUE EN UN CONTENEDOR DE LA BASE DE DATOS DEL SISTEMA DE
  # VOTACIONES
  - openssl aes-256-cbc -K $encrypted_80d99b7f2682_key -iv $encrypted_80d99b7f2682_iv -in deploy.enc -out deploy -d
  - mkdir mysql_build
  - cd mysql_build
  - curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/docker/dockerfiles/mysql/init.sql
  - curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/docker/dockerfiles/mysql/Dockerfile
  - docker build -t egc/mysql .
  - docker network create --subnet=172.18.0.0/16 dev
  - docker run -d --network=dev --ip=172.18.2.2 -e MYSQL_ROOT_PASSWORD=nothing egc/mysql
  - cd ..
  # FIN DEL DESPLIEGUE, YA SE PUEDEN REALIZAR TESTS CON LA BD
  # mysql -e 'CREATE DATABASE api;'
```

```
# mysql -uroot api < create_database.sql

before_deploy:
- chmod 600 deploy && mv deploy ~/.ssh/id_rsa
- curl -O https://raw.githubusercontent.com/EGC-G2-Trabajo-1718/integracion/master/tools/deploy.sh
# ADMINISTRACIÓN DE VOTACIONES: DESCOMENTAR SIGUIENTE LÍNEA
# - scp -o StrictHostKeyChecking=no target/*.war deploy@egc.duckdns.org:g2/tmp/ROOT.war

deploy:
  skip_cleanup: true
  provider: script
  script: ssh -o StrictHostKeyChecking=no deploy@egc.duckdns.org 'bash -s' < deploy.sh recuento # INSERTAR NOMBRE DEL SUBSISTEMA ESPECIFICADO
  on:
    branch: master

before_script:
- composer self-update

script:
- php vendor/bin/phpunit

install:
- composer install --no-interaction

after_script:
- php artisan serve --env=testing-ci --port=8000 --host=localhost &
```

La automatización de las pruebas se han implementado en Travis, de tal forma que se ejecuten todas las pruebas de forma automática.

Tendríamos el subsistema integrado:

✓ master fix: PollController.php #107 passed

Se añaden nuevas modificaciones en la clase PollController.php para adecuarlo a las pruebas anteriormente realizadas.

🕒 Ran for 1 min 25 sec
🕒 10 minutes ago

🔗 Commit 15a1996 🔗
🔗 Compare ef68658..15a1996 🔗
🔗 Branch master 🔗

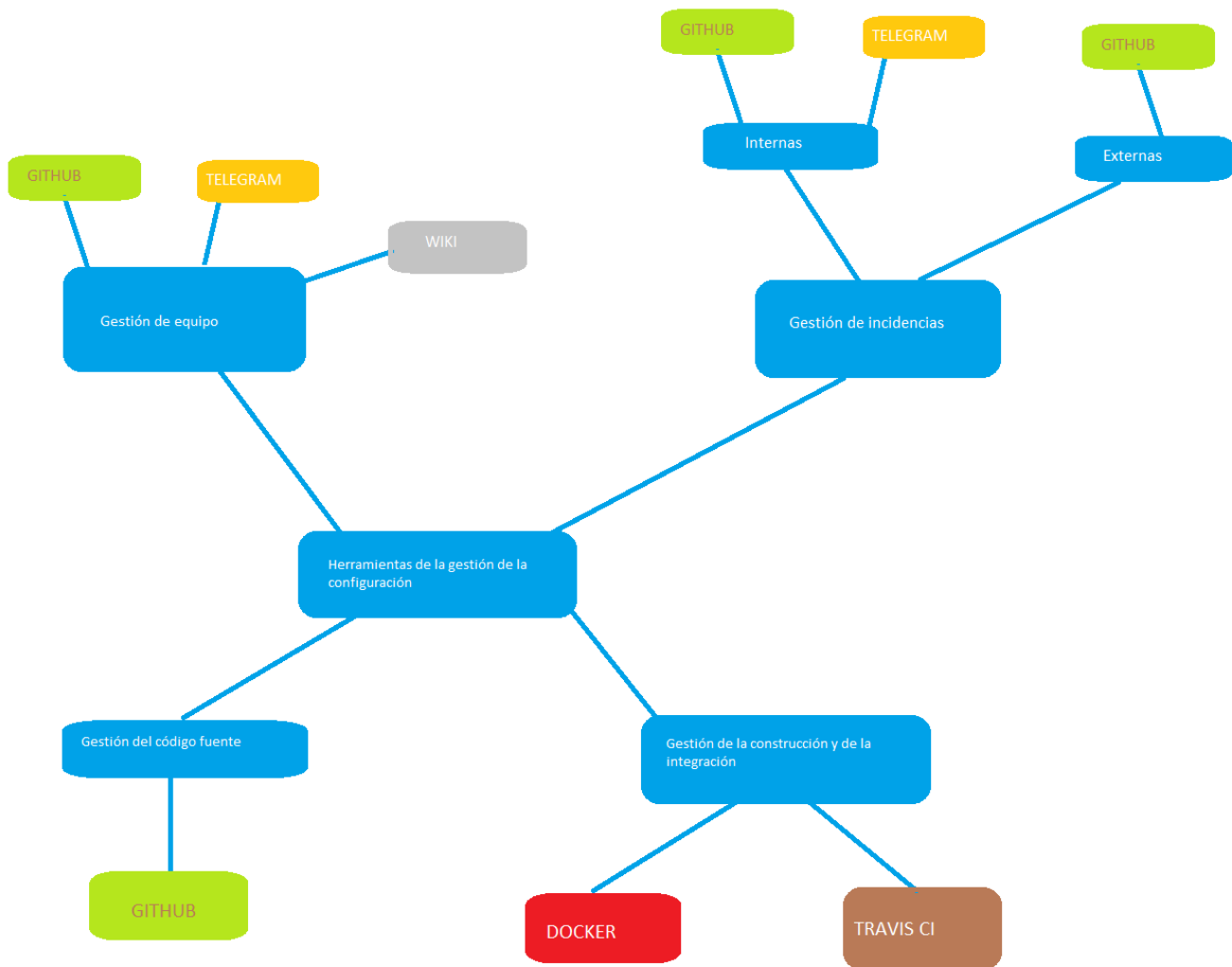
👤 Juan Carlos Cansino authored and committed

11. 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.

12. Mapa de herramientas



13. Ejercicio de propuesta de cambio

Planteamos como ejercicio de propuesta de cambio para nuestro proyecto que se implemente una nueva llamada para la API “getRatioPoll” en la cual se realice un recuento de la votación y devuelva el porcentaje de votos de cada una de las opciones con respecto al total de votos.

14. 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.

15. 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.