En el presente informe se detallan las respuestas a los desafíos solicitados.

Desafío N° 1

Para obtener un ambiente productivo de laravel debemos tener en cuenta los principales requerimientos de un servidor para este framework. Según la documentación, es necesario tener instalado lo siguiente:

- PHP >= 7.2.5
- BCMath PHP Extension
- Ctype PHP Extension
- Fileinfo PHP extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

Para gestionar los paquetes que se utilizarán durante el desarrollo y luego para producción es necesario composer. Con el mismo se puede iniciar un nuevo proyecto de la siguiente manera:

```
composer create-project --prefer-dist laravel/laravel desafiotwg "6.*"
```

En el ejemplo se utiliza la versión 6 de laravel ya que para un ambiente de producción es importante seleccionar la versión LTS. Esta estará soportada por la comunidad por un largo tiempo.
Para el ambiente productivo se puede utilizar Nginx. El mismo será intermediario entre los clientes y el servidor php donde estará ejecutándose la aplicación de laravel.
Ahora, es importante generar un nuevo archivo para las variables de entorno de producción. Por lo que debemos copiar el archivo de ejemplo y modificarlo a nuestro gusto. Una forma de hacerlo en linux es de la siguiente manera:

```
cp .env.example .env
```

Luego generamos la clave que se utilizará para encriptar en laravel:

```
php artisan key:generate
```

Siguiendo con el archivo ".env", se solicita una base de datos MySQL o MariaDB. Para ello necesitamos usuario, contraseña, dirección y puerto del host que posee el servicio de la base de datos y nombre de la misma. En nuestro archivo .env del proyecto laravel configuramos los siguientes parámetros con estos valores:

- DB_CONNECTION=mysql
- DB_HOST=db
- DB_DATABASE=db
- DB_USERNAME=laravel
- DB_PASSWORD=laravel

Nota: esto es solo un ejemplo, se deben utilizar contraseñas seguras.

Para la conexión a la base de datos redis necesitamos la dirección IP del host que contiene el servicio y el puerto.

- REDIS_HOST=redis
- REDIS_PASSWORD=null
- REDIS_PORT=6379

Para el servidor SMTP podríamos utilizar un servidor que provea de este servicio. Por lo que también en el archivo .env indicamos:

- MAIL_DRIVER=smtp
- MAIL_HOST=mail.undominio.com
- MAIL_PORT=2525
- MAIL_USERNAME=miemail@undominio.com
- MAIL_PASSWORD=unaclave
- MAIL_ENCRYPTION=
- MAIL_FROM_ADDRESS=
- MAIL_FROM_NAME="${APP_NAME}"

Otro paso muy importante es cambiar la variable APP_DEBUG a false. Esto cambiará el modo en el que se muestran los mensajes de error, comúnmente llamado "modo verbose". Además, se recomienda agregar la siguiente configuración al archivo app.php en las configuraciones del proyecto para ocultar información sensible al momento de hacer debug.

```
'debug_blacklist' => [
        '_ENV' => [
                'APP_KEY',
                'DB_PASSWORD',
        ],
        '_SERVER' => [
                'APP_KEY',
                'DB_PASSWORD',
        ],
        '_POST' => [
                'password',
        ],
],
```

Ahora es necesario ejecutar las migraciones necesarias para crear las tablas en la base de datos. Mediante artisan podemos llevar a cabo esta tarea:

```
php artisan migrate
```

Desafío N° 2

Este desafío solicita definir las funciones de relación entre dos modelos:

- Publication (id, title, content, user_id)
- Comment (id, publication_id, content, status)

Donde una publicación puede tener 0 o más comentarios. Por ello, las siguientes funciones pueden ser una de las soluciones a este desafío.

En el modelo Publication:

```
public function comments(){
       return $this->hasMany(Comment::class,'publication_id','id');
   }
```

El modelo Comment:

```
public function publication(){
       return  $this->belongsTo(
             Publication::class,
             'publication_id','id'
        );
 }
```

Desafío N° 3

A continuación se muestra la consulta. Se puede ver desarrollada con algunas modificaciones en el repositorio entregado.

```
$db = DB::table('publications as p')
           ->join('comments as c','p.id','=','c.publication_id')
           ->where('c.status','=','APROBADO')
           ->whereRaw("c.content REGEXP '.*hola.*'")
           ->select('p.id','p.content','p.user_id','p.title')
           ->groupBy('p.id','p.title')
           ->get();
```

Desafío N° 4

Hemos nombrado las migraciones pero no para qué sirven en los ambientes productivos. Estas nos facilitan las operaciones sobre nuestra base de datos de una forma controlada y segura, permitiendo agregar y/o modificar campos, tablas, realizar operaciones, entre otros.

También, es posible generar un nuevo ambiente en otro lugar fácilmente. Además, es independiente del tipo de motor de bases de datos utilizado, lo que es una ventaja a la hora de cambiarlo.

Desafío N° 5

El enlace al repositorio con este desafío es el siguiente:
https://github.com/fkmurphy/desafio_twgroup
Nota: en el archivo README.md se encuentra instructivo para crear el ambiente en docker. Los requisitos son poseer docker y docker-compose instalado. Fue probado con las versiones 19.03.12 y 1.24.0 respectivamente sobre un sistema operativo GNU/Linux distribución Fedora 30.

Bibliografía

- Documentación de Laravel. URL:https://laravel.com/docs/6.x
- Documentación de Bootstrap 4. URL: https://getbootstrap.com/docs/4.4/getting-started/introduction/