Bienvenido al test de evaluación de chicfy para el puesto de Full Stack Developer.

Por favor, intenta responder a todas las preguntas sin extenderte demasiado. Algunas preguntas pueden ser respondidas googleando, pero intenta responder personalmente.

Fecha:

Nombre y apellido:

**1. ¿Qué frameworks PHP has usado? ¿Qué destacarías de cada uno de ellos? Explica brevemente qué son los namespaces y cual es su funcionalidad.**

**2. ¿Qué patrón de diseño intenta im****plementar el siguiente código? Pros y contras de usar este patrón.**

<?php

class Foo {

private static $instance;

private function \_\_construct() {

}

public static function getInstance() {

if (null === static::$instance) {

static::$instance = new static;

}  
 return static::$instance;

**3. Nombre algún otro patrón de diseño usado en aplicaciones web.**

**4. ¿Qué es la inyección de dependencias? ¿Cuales son los principales beneficios de usarla?**

**5. ¿Has usado herramientas de integración continua? Cita alguna explicando para qué las has usado.**

**6. ¿Has usado tests unitarios o programado con metodología TDD? ¿Conoces algún mock framework? ¿Cuales son los principales beneficios de usar tests unitarios y mocks?**

**7. Escribe en pseudocódigo cómo harías un pequeño algoritmo que, dada una cadena de texto, la devuelva al revés. Ejemplo: chicfy -> yfcihc.**

**No uses código en ningún lenguaje, usa pseudocódigo para crear un algoritmo que realice la tarea y sin utilizar funciones predefinidas, simplemente los tipos de datos y estructuras de flujo básicas.**

**8. Ejercicio MySql**

Un departamento de una universidad decidió hace tiempo crear una pequeña biblioteca de libros relacionados con las materias que imparte y para poder gestionarla desarrolló una pequeña aplicación, basada en plataforma LAMP, que les permitía llevar el control de sus libros y de los préstamos de los mismos a alumnos y profesores. Dado que esta biblioteca fue un éxito, otros departamentos copiaron la idea y pasado un tiempo se decidió unificar las bibliotecas de los distintos departamentos en una única biblioteca general. La estructura de datos sobre la que se basa la gestión de la biblioteca es la siguiente:

CREATE TABLE `libros` (  
`id` int(10) unsigned NOT NULL,  
`isbn` varchar(13) DEFAULT NULL,  
`referencia` varchar(50) DEFAULT NULL COMMENT 'Código del libro en la biblioteca',  
`titulo` varchar(100) NOT NULL,  
`autor` varchar(150) NOT NULL,  
`departamento` varchar(50) NOT NULL COMMENT 'Dpto. dueño del libro',  
`id\_alumno` int(10) unsigned DEFAULT '0',  
`id\_profesor` int(10) unsigned DEFAULT '0',  
`fecha\_prestamo` date DEFAULT NULL COMMENT 'Fecha en que fue sacado de la biblioteca', `fecha\_limite` date DEFAULT NULL COMMENT 'Fecha limite para la devolución',  
PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE TABLE `alumnos` (  
`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,  
`nombre` varchar(100) NOT NULL,  
`apellidos` varchar(150) NOT NULL,  
`curso` int(10) unsigned NOT NULL DEFAULT '1' COMMENT 'curso en que está el alumno (1 ­ 3)', `libros` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '# libros actualmente en su posesión', PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE TABLE `profesores` (  
`id` int(10) unsigned NOT NULL,  
`nombre` varchar(100) NOT NULL,  
`apellidos` varchar(150) NOT NULL,  
`departamento` varchar(100) NOT NULL COMMENT 'Departamento al que pertenece',  
`libros` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '# libros actualmente en su posesión', PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Pese a que la solución actual es bastante funcional, al querer integrar nuevas funcionalidades ha surgido la duda de si sería interesante realizar una reprogramación del sistema que permita un mantenimiento y modificación más sencillos y facilite la implementación de las nuevas funcionalidades solicitadas.

Los nuevos requerimientos son los siguientes:

●  No todos los libros podrán ser retirados por todos los usuarios, algunos libros estarán disponibles únicamente para profesores y para alumnos adjuntos a proyectos de investigación.

●  Los alumnos sólo podrán retirar un máximo de 3 libros simultáneamente, los alumnos adjuntos hasta 5 libros y los profesores hasta 8 libros. Además el tiempo máximo del préstamo será de 2 semanas para los alumnos, 3 semanas para los adjuntos y 5 semanas para los profesores.

●  Debido a la alta demanda de algunos libros, se comprarán ejemplares adicionales de algunos de ellos.

●  Se quiere penalizar a los usuarios que no han devuelto sus libros en las fechas acordadas y por tanto un alumno que no devuelva un libro no podrá retirar más libros hasta devolver el/los libros atrasados, un profesor podrá acumular hasta 2 libros retrasados antes de que se le bloquee el acceso a más libros.

●   Se desea además mantener información histórica del uso de libros por parte de los usuarios y poder consultar el histórico de libros concretos.

**Plantea los cambios de estructura de datos para permitir que el sistema soporte todas las nuevas funcionalidades solicitadas. Debes entregar el script SQL de creación de tablas y la documentación de las decisiones de diseño de estructura de datos tomadas.**

**9. Ejercicio PHP**

<?php

class ListNode

{

public $data;  
 public $next;

function \_\_construct($data)

{  
 $this­->data = $data;

$this­->next = NULL;

}

}

class LinkedList

{

private $firstNode;

function \_\_construct()

{

$this­->firstNode = NULL;

}

public function first()

{

return $this­->firstNode;

}

public function insertFirst($data)

{

$link = new ListNode($data);

$link­->next = $this­->firstNode;

$this­->firstNode = &$link;

}

}

?>

**Añadir al código anterior dos nuevos métodos a la clase LinkedList. addLast($data) que añadirá un nodo al final de la lista y reverse() que modificará la lista a una de orden inverso.**

**10. ¿Qué consideras que es lo más importante a la hora de resolver un problema?**