Bienvenido al test de evaluaciónón 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: 12 de nov. de 15

Nombre y apellido: Jaume Sala Tantinya

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

He utilizado symphony2 y Zend.   
Symphony para mi fue el primer contacto con un framework, me gustó mucho la estructura de carpetas propuesta.  
Zend lo aprendí por requerimiento de un cliente. Destacto de el la estructura MVC.

**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;

En este patrón de diseño se está definiendo la clase Foo.  
El patrón creacional esta vacio (construct) que se llama en la creación de los objetos Foo.  
Al definir la función getInstance como pública se está definiendo un patrón de comportamiento al poder ser instanciado des de fuera de la lacas foo.  
Al final se está definiendo un método publico para llegar a $instance encapsualando $instance dentro de la clase FOO  
El PRO es que se está protegiendo la propiedad privada $instance y se está forzando a usar el método getInstance es una buena praxis.

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

Uno de los patrones más usados es el de MVC.  
Se usa para separa model (datos) vista (pintado) controlador (código asociado a la vista)  
Al separa modelo vista y controlador permite intercambiar algún modulo de tu proyecto sin afectar a los demás. Por ejemplo si cambiar de base de datos al solo tienes que cambiar el código asociado al modelo.

**4. ¿Qué es la inyección de dependencias? ¿Cuales son los principales beneficios de usarla?**La inyección de dependencias se basa en que tienes una clase o objecto que depende de un método o objecto de otra.   
El principal beneficio es poder reutlizar código.  
Por ejemplo si tienes la clase http que controla las llamadas a una API.  
En el controlador vista-main inyectas http y puedes llamar a la API rest.

**5. ¿Has usado herramientas de integración continua? Cita alguna explicando para qué las has usado.**Si. He usado Jenkins y bamboo. Jenkins por que es opensource bamboo en Inditex es de la suite de atlasian (que es la usada en ITX).Por ejemplo puedes configurar Jenkins para que después de cada comit pase todos los testos unitarios y funcionales.  
Utlizaabamos bamboo para desplegar en los diferentes entornos DEV, PRE, PRE

Luego yo personalmente también añado al paquete de integración continua, herramientas como gulp y grunt, hint pero se que la mayoría no lo hace.

**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?**Si he usado testos unitarios. Si he trabajado usando TDD (test driven development).  
Sinceramente no conozco ningún mock framework.  
Pero si he usado la idea de mockear.  
Por ejemplo puedes mockar una base de datos o servicio para asegurar un estado con el cual partirás para hacer los testos.  
  
El principal beneficio de usar testos unitarios es que tienes la seguridad que si rompes algo del código y los testos están bien hechos el bug no llegará a producción tu test-runner te lo habrá chivado.  
El principal beneficio de mockear algo es que puedes hacer los testos mucho más atomicos. Es decir si tu aplicación depende de un producto de terçeros, por ejemplo una API REST si no mockeas esta a API en tus testos no puedes asegurar los resultados que vas a obtener de ella y tus testos serán “falsos”. Por que para asegurar el resultado del test tienes que asegurar un estado inicial.

**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.**

Inicio función giraTexto (textoAGirar): texto

var: textoGirado : cadena\_de\_texto   
 var: longTextoAGirar: numérico

longTextoAGirar = textoAGirar. longitud

desde i = 1 hasta longTextoAGirar hacer

textoGirado[i] = textoAGirar [longTextoAGirar-(i-1)]

fin desde

Devuelve textoGirado

Fin funcion

giraTexto(“chicfy”)

**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.**

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;

}

public function addLast($data){

$this->fistNode.push ($dat)

}

puclic fucntion reverse(){

//no me dio tiempo ☹

}

}

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

**Creo que lo más importante para resolver un problema primero es definirlo exactamente. Por que si lo definies y analizas correctamente verás que el 90% de los casos no es un problema, más bien es un reto.**