PRUEBA TÉCNICA BACK-END

Este documento contiene un informe referente al desarrollo de la prueba técnica para desarrollador back-end.

Prueba técnica back-end

PUNTO NUMERO 1(CODING CHALLENGE)

Para la realización de este punto se usó un framework llamado Cakephp el cual presenta un patrón conocido como MVC (modelo vista controlador) en el cual se estructuro la aplicación de la siguiente manera:

La estructura de carpetas que se maneja es la siguiente.

App|
- |Controller (almacena los controladores de la aplicacion)
- |Modelo
  - |Clases (almacena las clases necesarias)
- |templates
  - |users (almacena vistas necesarias)

Modelo

El modelo es el encargado del acceso a la información en este caso proveniente un objeto creado por la clase matriz.php que se mantiene vivo durante la ejecución.

Matriz.php

Esta clase está construida para manejar las diferentes operaciones que se deben realizar dentro del ejercicio y adicionalmente mantener durante la ejecución de la operación las variables necesarias para la ejecución del programa.

Controlador

En la parte de controlador se encarga de manejar la lógica del negocio y dentro de la realización del ejercicio se utilizaron las clases UserCotroller.php.

Usercontroller.php

Dentro de la lógica del negocio esta clase es la encargada de recibir la información que viene desde la vista de usuario validarla y usar las funciones declaradas dentro de matriz.php para la realización del proceso solicitado y envía nuevamente la información de la respuesta hacia la vista del usuario.

Vistas

Las vistas son las encargadas de mostrar al usuario tanto el formulario para la carga de datos, como los resultado obtenidos del procesamiento de la información. El archivo que define esta vista se llama operaciones.ctp.

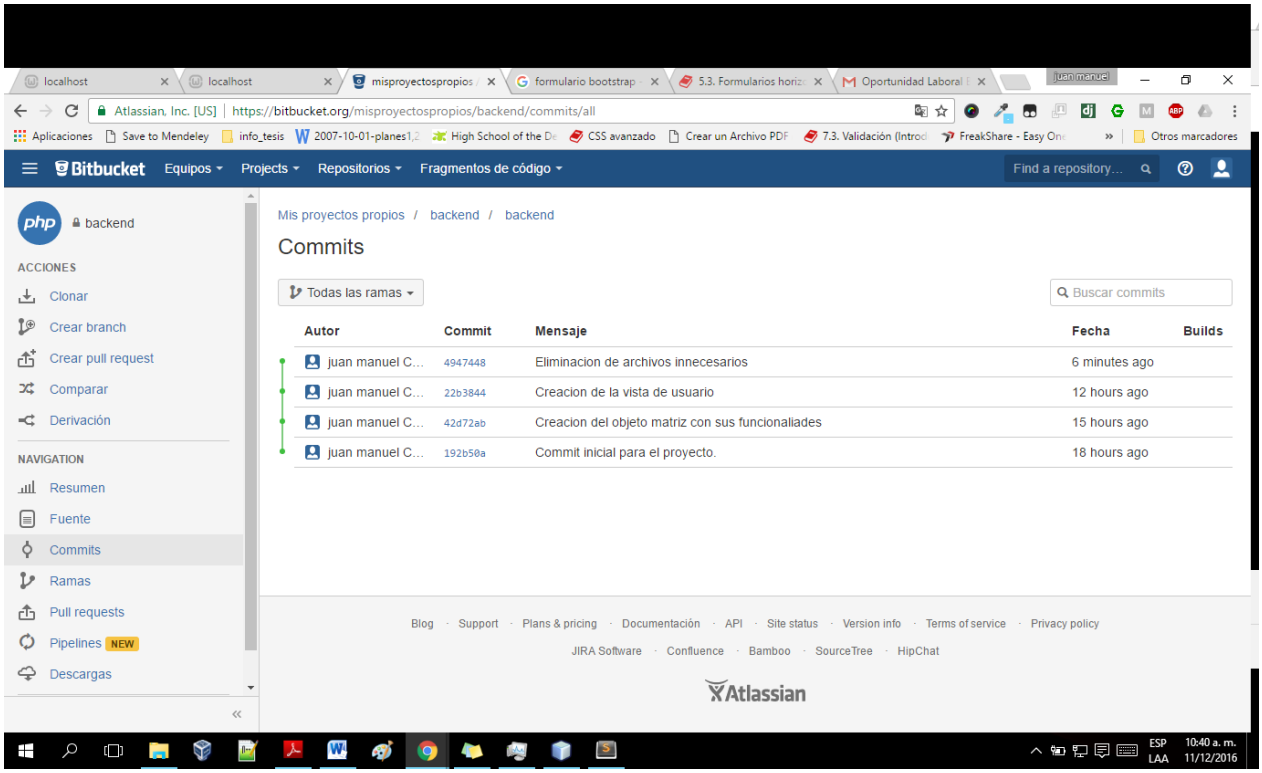
operaciones.ctp

Cuenta con el código php y html necesario para pintar la información que se le va mostrar al usuario.

Repositorio

Por un error en la creación del repositorio quedo creado como privado por lo cual no es accesible para quien no este agregado al equipo de trabajo por lo cual adjunto imagen de los commit hechos.

Si es necesario puedo agregar a un usuario que me indiquen para que se peuda acceder al repositorio.



Punto numero 2(CODE REFACTORING)

1. La funcionalidad no presenta ningún tipo de documentación que indique operación desempeña ni tampoco indica los parámetros de salida y salida respectivamente.
2. Existe gran cantidad de código comentado que genera ruido al momento de interpretar el código funcional.
3. Las validaciones de los comandos if se deben manejar con las funciones propias para php como por ejemplo para saber si un parámetro es null debemos usar la función propia de php llamada is_null().
4. No se verifica si las entidades se actualizaron con éxito.
5. No se está verificando que los mensajes push fueron enviado de forma correcta.
6. Los códigos de error retornado deberían ser numéricos y no strings para facilitar su tratamiento.

El código refactorizado queda anexo al presente documento en el archivo cod_refactorizado.php.

Punto numero 3(Preguntas)

¿En qué consiste el principio de responsabilidad única? ¿Cuál es su propósito?

El principio de responsabilidad única es utilizado en su mayoría en la programación orientada a objetos y establece que una clase debe tener una sola responsabilidad y un solo propósito aumentando de esta manera que el nivel de cohesión sea lo más alto posible y el nivel de acoplamiento sea lo más bajo posible; esto traduce en que la clase es fácilmente reutilizable y que un cambio en el código no afectar el funcionamiento.

¿Qué características tiene según tu opinión "buen" código o código limpio?

- No contenga código comentado que genere ruido al momento de leer.
- Que se encuentre correctamente indentado y ordenado.
- Los nombres de las variables y nombres de las funciones deben tener nombres acordes con su entorno y función.
- Debe expresar con facilidad para que esta hecho.
- Hay que evitar la redundancia de código.
- Mantener las capas de implementación separadas vistas, lógica del negocio, acceso a datos.