Entregable # 1 – Arquitectura MVC

|  |  |
| --- | --- |
| **Equipo de Trabajo** | **Nombre** |
|  |
|  |
|  |

**Nombre proyecto:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Logo equipo

<Dele una identidad a su equipo, diseñe un logo que represente a su equipo. Coloque en este espacio el logo de su equipo.>

1. Modelo verbal definitivo

<Explique en este espacio en que consiste el proyecto que pretende realizar, defina el alcance del proyecto, los actores involucrados y el beneficio de la propuesta que se presentará.>

1. Diagrama de clases

<Desarrolle el diagrama de clases del proyecto – este diagrama de clases se enfocará en el dominio de la aplicación (elementos del negocio). Enfóquese en los modelos, olvídese de los controladores, u otras clases adicionales programadas.>

1. Diagrama de arquitectura

<Desarrolle un diagrama de la arquitectura MVC del proyecto, enfóquese en la relación entre los clientes y el servidor, y especifique claramente las capas MVC en el diagrama.>

1. Implementación en Laravel

Instrucciones para el arquitecto (un integrante del equipo será el arquitecto):

* Cree un repositorio en GitHub (rama master).
* Clone el repositorio localmente, luego cree un proyecto en Laravel 7.x y súbalo al repositorio GitHub.
* Comparta el repositorio con sus compañeros.
* Repártales a los compañeros las clases que deberán programar. Por ejemplo, si su proyecto consta de 6 clases, repártales de a 2 o 3 clases a cada compañero. Usted como arquitecto tiene total libertad de la repartición de las clases. Recuerde que usted deberá sacar tiempo para la elaboración de unos documentos y para revisar cada código nuevo que intenten aplicar sus compañeros.
* Sus compañeros nunca podrán hacer “push” directo a la rama master.
* Cuando un compañero decida trabajar en el proyecto, deberá: (i) crear una nueva rama, o (ii) realizar un fork del proyecto. Y cuando un compañero desee aplicar los cambios a la rama principal (master o development) deberá realizar un pull request desde GitHub.
* El arquitecto analizará el pull request y solo aceptará los cambios si los considera pertinentes.
* Defina en GitHub un directorio de documentos de codificación. Ese directorio incluirá 2 archivos:
  + **Guía de estilo de programación.** Defina un documento corto (de no más de 2 páginas) con todas las indicaciones de estilo de codificación. Por ejemplo, defina estilo para los ifs, para los while, como nombrar las clases, los métodos, las variables, etc. Ejemplos: <https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md> - <https://guidelines.spatie.be/code-style/laravel-php>
  + **Reglas de programación.** Defina otro documento corto (de no más de 2 páginas) con las reglas que usted considera esenciales de programación en su proyecto Laravel. Divida las reglas por categorías (reglas para controladores, para modelos, para vistas, para rutas, etc). Ejemplo de reglas esenciales: (i) nunca haga un echo en un controlador. (ii) Toda ruta debe estar asociada a un controlador. (iii) Toda vista debe extender del layout master. (iv) Todas las vistas deben ser Blade. (v) nunca abra y cierre php dentro de las vistas. Etc.
  + Si un compañero envía un pull request que no siga las reglas definidas en los dos documentos anteriores, remítalo a esa documentación.
* Cree un tablero en asana, trello, o cualquier otro sistema donde pueda controlar las actividades y tareas pendientes del equipo.

Instrucciones globales:

* El sistema debe ser un proyecto web con Laravel y base de datos MySQL.
* Deberán implementar todas las clases de su proyecto.
* Deberán utilizar el sistema “migrations” de Laravel para llevar el registro de cambios del SQL.
* Utilice fakers para facilitar la ejecución inicial del programa, o como plan b, provea un SQL con inserción de datos ficticios.
* Recuerde que cada línea (asociación) en el diagrama de clases se convierte en dos funciones (dinámicamente propiedades) en las dos clases que se relacionan.
* Cree un archivo README.TXT donde explique cómo ejecutar el programa, cual es la ruta principal que se debe invocar, entre otros.
* Utilice el sistema de Login que ofrece Laravel (todo un sistema de Login se puede crear en menos de 15 minutos). Ese sistema está asociado a la clase User que ya viene con los proyectos de Laravel.
* La aplicación deberá contener por lo menos “7 funcionalidades interesantes” diferentes a las tradicionales: crear, editar, borrar y leer. Ejemplo: (i) búsqueda de productos, (ii) ver top 3 productos más vendidos, (iii) generar en pdf la factura de venta, (iv) descargar mis notas en excel, etc.
* **Detalle las funcionalidades principales de su aplicación:** haga una tabla en excel, con 3 columnas. Columna 1 nombre de la funcionalidad, Columna 2 nombre del archivo donde está implementada la funcionalidad, Columna 3 línea desde la cual inicia la implementación de la funcionalidad. Agregue el archivo a GitHub. Liste las funcionalidades interesantes al final.
* Tome un pantallazo de las 3 secciones más importantes de la aplicación guárdelas en GitHub.
* Todos los textos del proyecto deben ir resources/lang/\* - si le da tiempo, implemente el proyecto en 2 idiomas.
* Aplique principios DRY y ETC. Sugerencia: ver libro “Pragmatic programmer” topic 8 y 9 (son solo 4 páginas).
* Aplique los principios, tips, y sugerencias que se han brindado durante todo el curso. Muchos de ellos no aparecen en las presentaciones ya que se discutieron en clase.
* Ojo al reutilizar el código de los talleres, aunque es una muy buena base para iniciar, hay varios cambios y mejoras que se deben realizar que se discutieron durante las clases.
* Implemente 2 pruebas unitarias.
* Coloque la información del autor del código de cada archivo en la parte de arriba de cada archivo (como un comentario).

Instrucciones de entrega para el arquitecto:

* Suba el proyecto a la cuenta que se le brindó en AWS.
* Cree un .zip de todo el proyecto y súbalo por Teams (en caso de que no se pueda por cuestiones de tamaño, envíelo por wetransfer o mega al correo del docente).
* Suba este documento al repositorio de GitHub. Y finalmente comparta el link del repositorio con el docente (al correo del docente) antes de la fecha de finalización de entrega.