Entregable # 1 – Arquitectura MVC

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 (no importa si el logo no tiene nada que ver con su proyecto). Lea el siguiente texto para que entienda que tipo de logo esperamos. >

COMMUNICATE TEAM PRESENCE

Great project teams have a distinct personality. People look forward to meetings with them, because they know that they'll see a well-prepared performance that makes everyone feel good. The documentation they produce is crisp, accurate, and consistent. The team speaks with one voice.

There is a simple marketing trick that helps teams communicate as one: generate a brand. When you start a project, come up with a name for it, ideally something off-the-wall. (In the past, we've named projects after things such as killer parrots that prey on sheep, optical illusions, gerbils, cartoon characters, and mythical cities.) Spend 30 minutes coming up with a zany logo, and use it. Use your team's name liberally when talking with people. It sounds silly, but it gives your team an identity to build on, and the world something memorable to associate with your work.

2. Modelo verbal definitivo

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

3. 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.>

4. 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.>

5. Implementación en Laravel

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

- Cree un repositorio público en GitHub.
- Clone el repositorio localmente, luego cree un proyecto en Laravel 10.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.
- El arquitecto analizará cada commit y push hecho en el proyecto y podrá revertir a versiones anteriores si no se siguen las reglas definidas por el arquitecto.
- Cree un wiki en su repositorio GitHub. Ese wiki tendrá una página principal, con una pequeña descripción de la aplicación, los nombres de los integrantes, y con enlace a otras 3 páginas:
    - **Entregable:** la página entregable contendrá los elementos 1 a 4 de este entregable.
    - **Guía de estilo de programación.** Defina o (i) una guía de estilo de codificación manual que usted se invente, o (ii) explique cómo se debe usar Laravel Pint en su proyecto.
    - **Reglas de programación.** Defina en esta página 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 app. (iv) Todas las vistas deben ser Blade. (v) nunca abra y cierre php dentro de las vistas. Etc. **Nota:** Si un compañero envía un push que no siga las reglas definidas en las 2 páginas anteriores, remítalo a esa documentación.
- Utilice el sistema de “projects” de GitHub para dividir las tareas, y que cada integrante señale en que está trabajando y que está pendiente.

Instrucciones globales:

- El sistema debe ser un proyecto web con Laravel 11 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.
- Vaya creando datos ficticios, y cuando tenga un buen grupo de datos ficticios exporte el SQL (correspondiente a las filas) y suba ese SQL al repositorio (normalmente jamás se subiría un SQL a un proyecto público, pero para facilidad de la entrega así lo haremos).
- 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.md en la raíz del repositorio donde explique cómo ejecutar el programa, cuál es la ruta principal que se debe invocar, entre otros.

- La aplicación debe tener 2 secciones principales. Sección del usuario final, y sección del administrador. Por lo tanto, se debe utilizar un sistema de Login. Utilice el **Tutorial LOGIN LARAVEL** para agregar un sistema de login totalmente funcional a su proyecto.
    - **Sección administrador,** por ejemplo "/admin/*". Es una sección donde solo puede ingresar un administrador, y donde se ejecutarán CRUDs de las clases (implemente por lo menos 2 CRUDs completos de 2 clases de su proyecto en el panel de administración). **Ejemplo CRUD para clase Producto:** una sección donde puedo listar productos, crear productos, modificar productos, eliminar o desactivar productos. El taller 1 del curso ayudará a completar bastante esta sección. **Nota importante:** esos 2 CRUDs de administrador, son independientes de la sección de usuario final, son vistas independientes con controladores independientes.
    - **Sección usuario final,** por ejemplo "/*". Es la aplicación que ve el usuario final, ve los productos, los compra, etc. Pero no puede borrarlos, ni modificarlos, ni crearlos.
    - **Nota:** ojo que las vistas de la sección de administrador no deberían ser las mismas (o compartirse) con las vistas del usuario final. Los paneles de admin se ven muy diferentes a las vistas de usuario final, ejemplo de panel de admin: https://nova.laravel.com/img/laravel-nova-mockup-light.png

- La aplicación deberá contener por lo menos "4 funcionalidades interesantes" diferentes a las tradicionales: crear, editar, borrar y leer. Ejemplos: (i) búsqueda de productos por nombre, (ii) ver top 3 productos más vendidos, (iii) generar en pdf la factura de venta, (iv) ver top 4 productos más comentados.

- Añada 2 páginas al wiki y enlácelas en la página principal.
    - **Funcionalidades interesantes:** en esta página describa cuales son las 4 funcionalidades interesantes, y en qué archivos están implementadas (especifique la línea exacta desde donde arranca la implementación).
    - **Pantallazos:** Tome un pantallazo de las 3 secciones más importantes de la aplicación y colóquelas en esta página.

- 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 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.
- 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 GCP.
- Comparta el link del repositorio y de la ruta de la instancia GCP con el docente (al correo del docente) antes de la fecha de finalización de entrega. Si lo desea, enlace la instancia GCP con un nombre de dominio gratuito (por ejemplo .tk).

Enlace GitHub: _____
Enlace GCP: _____