Diseño y programación web

	

Parcial 2: Trabajo práctico domiciliario

Consignas para la entrega del parcial:

- El desarrollo es **individual**.
- Se desarrolla a partir de lo realizado en el parcial 1
- Si en TP1 trabajaron grupalmente deben **continuar individualmente** y realizando aspectos diferentes del mismo trabajo. O sea dos personas que integraron el mismo equipo antes no deben presentar lo mismo entre sí.
- Si dos entregas tienen el mismo código para el TP2 se invalida el que haya entregado último.

Forma de presentación:

- Únicamente por Classroom
    - No se aceptan entregas por otros medios (email, drive, wetransfer, etc.)
- Fecha de entrega: Ver en Classroom
    - No se aceptan trabajos fuera de la fecha límite, **sin importar el motivo**.
    - Archivos subidos pero **no entregados** no cuentan.
- Formato de entrega
    - En archivo .zip o .rar
    - Nombre de archivo **mobile-parcial-2-dwn3bp-apellido.nombre.zip**
    - Todo en minúsculas, separado por guión, sin espacios
- Incumplir lo anterior resta 2 puntos, pudiendo desaprobar.
- Archivos mal comprimidos o corruptos se consideran **ausente**.

- En caso de "no llegar a tiempo", enviar igualmente lo hecho hasta el momento y agregar un mensaje avisando que no realizará la entrega y detallando los motivos.

Consigna del trabajo

Durante las clases dadas entre el trabajo práctico 1 y la presentación del trabajo práctico 2, se vieron lo siguientes temas:

- Seeders
- Factories
- Relaciones:
    - Uno a uno (tiene uno)
    - Uno a muchos (tiene muchos)
    - Pertenece a
    - Muchos a muchos
- Controllers
- Route Model Binding
- Rutas y controladores resourceful
- Middlewares

Sobre estos temas **y <u>solo de estos temas</u>** consta el desarrollo del trabajo práctico 2.

Para obtener la nota mínima 4 (cuatro)

- Continuar con la temática del parcial 1.
- Continuar utilizando todo lo aprendido en la primera entrega.
- Incluir un seeder y una factory para cada modelo. En el caso de relaciones, con la generación adecuada a través de los métodos correspondientes. Ver documentación.
- Cada modelo debe tener al menos 1 relación de cada tipo:
    - Uno a uno / Pertenece a
    - Uno a muchos / Pertenece a
- Se debe aplicar un caso de relación de cada tipo.

- En el caso que ninguno de los modelos actuales en el proyecto permita establecer alguno de los tipos de relación, agregar nuevos modelos para que se pueda lograr.
- Cada controller debe ser resourceful.
- Separar los controllers en carpetas. Ej. los controllers de Admin y de Customer.
- En cada controller implementar únicamente los métodos correspondientes a cada rol o espacio. Se supone que dos usuarios con distinto rol o en distinto espacio no **deben** ver la misma información ni pueden realizar las mismas operaciones.
- No incluir la carpeta **vendor** en el entregable **(muy importante)**
- Cualquier falta de lo mencionado implica la desaprobación del parcial, haya hecho o no cosas no solicitadas del punto siguiente.

Agregados para obtener 10

- Se requiere primero aprobar los puntos básicos para obtener 4.
- Migrar todas las validaciones a FormRequest.
- Definir mensajes de validación para cada campo.
- Hacer una vista base (layout) para admin que se distinga del sitio público.
- Reutilizar componentes de vistas con @include
- Definir más de 1 relación (si tiene sentido) en los modelos.
- Migrar (correctamente) todos los nombres a inglés.
- Implementar paginado en las páginas de listado, sin Javascript, con el uso del Paginador de Laravel. Ver documentación. - Opcional
- Utilizar tablas pivot con datos o columnas extra.
- Muchos a muchos - opcional

Prácticas que restan puntos:

- Incumplir cualquiera de los puntos anteriores.
- Copiar y pegar fragmentos de código producidos mediante IA u obtenidos por otros medios, como posteos de Stack Overflow, proyectos de Github u otros recursos similares, sin dejar documentada la fuente de dicho recurso.
- Utilizar funciones, clases y/o métodos del framework de **temas que no vimos en clase y que no sepan explicar en una evaluación teórica**.
- Agregar en los controllers métodos que no permitan el funcionamiento resourceful (ej. indexAdmin, createForm)
- Instalar y usar plugins, librerías o packages para Laravel que automaticen o simplifique la realización de las tareas solicitadas, por ejemplo ejecutando un solo comando de Artisan.
- Errores en consola relativos a la aplicación antes o durante el uso de la app.
- Lógica de la aplicación en las vistas (Blade). Por ejemplo, calcular el descuento de un producto o calcular el total a pagar en el archivo de vista.
- Links que no redirigen correctamente.
- Crear modelos para tablas Pivot. Más información.
- El diseño es importante pero no tanto como el uso del framework. Aplicar un buen diseño pero sin descuidar lo "que no se ve" en el navegador.
- Copy / Paste de los ejemplos usados en clase por el profesor para explicar.

Consideraciones finales:

- La entrega del TP con el cumplimiento correcto de las consignas es suficiente para considerar como mínimo un 4 (cuatro).
- Si al intentar servir el proyecto desde la consola (php artisan serve) no se puede completar por errores o faltantes en el código de la aplicación, automáticamente se califica con 2 (dos).
- El profesor puede requerir una explicación teórica oral o escrita para completar

la nota final. La explicación puede ser solicitada aunque el TP esté o no aprobado. Y se tomará en horario de clase posterior a la fecha de entrega.

- Se recomienda **entregar el TP 1 día antes de la fecha límite** para no arriesgarse a perder la posibilidad de presentarlo por fallos de conexión, daño del equipo, pérdida de los archivos, u otro tipo de situaciones imprevistas.

Profesor: Alejandro Villafañe alejandro.villafane@davinci.edu.ar