Da Vinci - Portales y Comercio Electronico

Final - 2023 2Q

Profesor: Alejandro Villafañe

Sobre el trabajo 👷

Trabajo individual, **no grupal**

No incluir en la entrega carpetas `vendor` o `node_modules`

En la carpeta raiz un archivo "datos.txt" con la siguiente información: Carrera, materia, cuatrimestre, año, turno, comisión, apellido y nombre, docente, carácter de entrega (final).

Condiciones para la entrega 📁

Unicamente por DV Panel - **no classroom**

Fecha definida en DV Panel

Formato de archivo zip o rar

Nombre del archivo "**apellido-nombre_final_.zip**" - Ejemplo "alejandro-villafane_final_.zip"

Advertencias ⚠️

El incumplimiento de la modalidad de entrega en cualquiera de sus puntos estipulados puede incurrir en una penalización en la nota de al menos un punto.

De detectarse un trabajo copiado, tanto copiador como copiado recibirán automáticamente una calificación de 1 (uno).

Los trabajos generados parcial o totalmente por inteligencia artificial también recibirán automáticamente una calificación de 1 (uno).

Requerimiento 💻

Sitio web de temática libre que ofrezca una aplicación que sea un servicio.

Se divide en al menos 2 partes: pública y privada.

La parte pública es el sitio comercial, navegable y usable para cualquier visitante. Debe tener al menos 3 páginas que sean: Detalle de producto o servicio, Registro y Login.

La parte privada es el sitio exclusivo para usuarios registrados con algún rol o cualidad que les permita hacer operaciones.

Los usuarios autenticados pueden ver sus datos personales y modificarlos (email, password, avatar, nombre, apellido, etc.)

Tiene que existir algún ABM o más de uno, que implementen **todas** las funcionalidades de lectura, creación o alta, modificación, borrado.

Algun usuario con rol administrador debe poder ver todos los usuarios, con los servicios contratados o los productos comprados por cada uno.

También alguna página que muestre estadísticas relevantes. Por ejemplo: cantidad de usuarios registrados por dia, plan con mas suscriptores, promedio de ventas semanales o mensuales, etc.

La estructura HTML debe ser válida, correcta, y semánticamente óptima.

Usar algun framework para estilizar el sitio, puede ser Bootstrap, Tailwind, Bulma. No usar templates prefabricados.

La base de datos debe contener al menos 5 tablas **ademas de las que usa laravel** para su funcionamiento.

Entre las 5 tablas deben estar incluídas todas las formas de relaciones: 1 a 1, 1 a muchos, muchos a muchos.

Deben existir al menos 2 roles de usuario.

Seguimiento ✍️

Deberan realizar un Changelog de los avances diariamente.

Qué es un changelog ? Un registro de cambios.

Por ejemplo, si visito el repositorio de Laravel en Github voy a ver un changelog del framework. Lo encuentran aqui **https://github.com/laravel/laravel/blob/10.x/CHANGELOG.md**

Si no les sale el formato identico, no importa, lo importante es que registren por cada día los avances que realizaron.

En un changelog no se ponen preguntas, ni anotaciones personales, ni lo que voy a hacer, ni lo que me quedo pendiente. Es un registro de **LO QUE HICE** en cada avance.

```
[2023-12-06]
- Redacté la consigna de evaluación final
- Modifiqué la lista de requerimientos
- Publiqué el documento en classroom

[2023-12-07]
- ...
```

Con el proyecto podria ser algo como...

```
[2023-12-06]
- Cree el proyecto con Composer
- Cree un modelo Example
    - Le coloque las propiedades $fillable
    - Especifique el nombre de la tabla
    - Agregue el SoftDeletes
- Cree una migracion para la tabla `examples`
    - Defini 5 columnas para: id, title, description, created_at, updated_at

[2023-12-07]
- ...
```

Existe un estandar al respecto y es una practica presente en toda la industria del software. En el sitio oficial pueden ver mas ejemplos **https://keepachangelog.com/en/1.1.0/**

Cosas que NO deben hacer 🚫

Inventar relaciones o casos irreales para forzar casos dificiles de encontrar.

Ej. "*Como no encontre 1 a 1 hice que una foto solo pueda tener un comentario*". 🤦

Usar los seeders para cargar datos "reales".

Modificar migraciones que ya ejecutaron, los cambios son siempre hacia adelante.

Usar librerías que generen soluciones inmediatas (ej. Auth)

Cosas que SI deben hacer ✅

- ✔Convenciones de nombres adecuadas a Laravel (ver link abajo)
- ✔Blade para generar las vistas **https://laravel.com/docs/10.x/blade**
- ✔Migraciones **https://laravel.com/docs/10.x/migrations**
- ✔Factories para todos los modelos **https://laravel.com/docs/10.x/eloquent-factories**
- ✔Seeders para todas las factories **https://laravel.com/docs/10.x/seeding**
- ✔Modelos con relaciones **https://laravel.com/docs/10.x/eloquent#introduction**
- ✔Controladores resourceful **https://laravel.com/docs/10.x/controllers#resource-controllers**
- ✔Subida de archivos **https://laravel.com/docs/10.x/filesystem#file-uploads**
- ✔Form Request con autorizacion y reglas. **https://laravel.com/docs/10.x/validation#form-request-validation**
- ✔Route Model Binding **https://laravel.com/docs/10.x/routing#route-model-binding**
- ✔Rutas restful **https://restfulapi.net/resource-naming/**

Buenas practicas

En el siguiente link tienen un ejempo de las mejores practicas que pueden implementar con Laravel, algunas de las cuales vimos durante las clases **https://github.com/alexeymezenin/laravel-best-practices**

Ahi podrán observar muchas de las formas que ustedes usaron en los TP1 y TP2 que están desaconsejadas y además **que no estaban acordes a lo visto en clase**. Eso ocurre cuando en lugar de seguir las clases siguen videos tutoriales de youtube o usan fragmentos de código de ChatGPT o de otras fuentes. Eviten tutoriales de youtube, por mas que sean muy populares o parezcan muy buenos o porque muestran como resolver el problema que necesitan. A veces no enseñan las mejores practicas o ni siquiera tocan temas que vimos en clases.

Recursos

Para lo que necesiten ayuda les recuerdo nuevamente que pueden ver el tutorial oficial sobre Laravel producido por los creadores del framework **https://laracasts.com/series/laravel-8-from-scratch** - Ademas de explicar el uso del framework, aplica muy buenas prácticas y es muy similar a la modalidad que vimos en la cursada.

Eviten descargar archivos o codigo de internet, usen siempre lo que proveen los sitios oficiales de las herramientas que usan.

Otros aspectos que consideraremos

- Realismo, originalidad y creatividad.
- Complejidad de la tarea realizada.
- Coherencia en los nombres de variables, clases, métodos, etc.
- Uso correcto de las etiquetas semánticas de HTML.
- Estética del sitio.
- Prolijidad en el código y carpeta del proyecto.
- Accesibilidad. Son muchos principios que los vemos aplicados a diario en apps y sitios, pero que no sabemos su origen. Pueden conocerlos leyendo aqui **https://www.id4you.com/blog/tendencias/los-4-principios-basicos-de-una-web-accesible/**
- Navegabilidad. Nada de enlaces rotos o enlaces que estan solo por que hay que rellenar espacio. Tampoco páginas a las que tenga que acceder colocando manualmente la url en el navegador. Todo debe ser guiado a través de la vista y de un proceso que me lleve a la página de destino.
- Usabilidad. El diseño tiene que servir para ayudar al usuario a usar la app, no para dificultarle la visita. Por ejemplo, poner un mensaje de error arriba cuando el input esta por la mitad de la página, no es bueno. O poner un boton rojo para acciones positivas y uno azul para negativas, es totalmente contra intuitivo. Mas sobre el tema pueden leer aqui **https://es.semrush.com/blog/usabilidad-web-principios-jakob-nielsen/**

Dudas o consultas durante el desarrollo de la consigna pueden enviarlas a **alejandro.villafane@davinci.edu.ar**