

MEMORIA DE ESTADÍA

PARA OBTENER EL TÍTULO DE

TÉCNICO SUPERIOR UNIVERSITARIO

EN

TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE DESARROLLO DE SOFTWARE

MULTIPLATAFORMA

“SISTEMA WEB PARA CURSOS INTERNOS AUTOMOTRIZ BONN”

“AUTOMOTRIZ BONN S.A DE C.V.”

PRESENTA
ROMERO ROMERO MARCO ANTONIO

SAN PABLO HUIXTEPEC OAXACA A 14 DE AGOSTO DE 2023

“CONOCIMIENTO PRÁCTICO QUE TRANSFORMA”

MEMORIA DE ESTADÍA

PARA OBTENER EL TÍTULO DE

TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE LA INFORMACIÓN ÁREA DE DESARROLLO DE SOFTWARE MULTIPLATAFORMA

“AUTOMOTRIZ BONN S.A DE C.V.”

PRESENTA
ROMERO ROMERO MARCO ANTONIO

ASESOR ACADÉMICO

M.E. Moisés Adrián Hernández Luis

ASESOR EMPRESARIAL

Lic. Olivia Pacheco Díaz

“CONOCIMIENTO PRÁCTICO QUE TRANSFORMA”

SAN PABLO HUIXTEPEC OAXACA A 14 DE AGOSTO DE 2023



Villa de San Pablo Huixtepec, Oaxaca, a 14 de agosto de 2023.

**DIVISIÓN DE CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN.
OFICIO: UTVCO.TICS.0175.2023
ASUNTO: DICTAMEN**

**ROMERO ROMERO MARCO ANTONIO
ALUMNO (A) DE LA CARRERA DE
TSU. EN TECNOLOGÍAS DE LA INFORMACIÓN.
P R E S E N T E**

Con la finalidad de iniciar su trámite de titulación para obtener el grado de TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE LA INFORMACIÓN ÁREA “DESARROLLO DE SOFTWARE MULTIPLATAFORMA” y después de haber concluido satisfactoriamente su periodo de estadía en la empresa “AUTOMOTRIZ BONN S.A. DE C.V.” donde desarrolló el proyecto.

“SISTEMA WEB PARA CURSOS INTERNOS AUTOMOTRIZ BONN”

Esta dirección autoriza la impresión de la memoria respectiva.



“Conocimiento práctico que transforma”
SUPERIOR
UNIVERSITARIO EN
TECNOLOGÍAS DE
LA INFORMACIÓN
UNIVERSIDAD TECNOLÓGICA DE LOS VALLES CENTRALES

MTRA. YESENIA DÍAZ PEREZ
ENCARGADA DE CARRERA

UTVCO

2022-2028

M.E. MOISÉS A. HERNÁNDEZ LUIS
ASESOR ACADÉMICO

Vb.Bb

C.c.p. Expediente
YDP/erm.

AGRADECIMIENTOS

Querida familia, amigos y a mis queridos perros,

Hoy quiero tomar un momento para expresar mi más profundo agradecimiento a cada uno de ustedes. Durante el tiempo de elaboración de mi proyecto, su apoyo incondicional ha sido un pilar fundamental en mi camino hacia el logro de mis metas.

Familia, ustedes han sido mi fuente constante de aliento y apoyo emocional. Sus palabras de aliento y sus abrazos cálidos han sido mi refugio en los momentos de duda y cansancio. Gracias por creer en mí y por estar siempre a mi lado, motivándome a seguir adelante.

Amigos, ustedes han sido mi fuerza impulsora. Cada palabra de aliento, cada sonrisa compartida y cada momento de diversión han sido un recordatorio de que puedo enfrentar cualquier desafío con valentía y determinación. Gracias por estar presentes, por escucharme y por ser mis aliados incondicionales.

A ti querida Andrea quiero agradecerte por brindarme tu apoyo y cariño siempre, por darme tus consejos por creer en mí y ayudarme en instantes en las que ya no quería continuar.

Y a ustedes, mis queridos perritos, gracias por ser mis compañeros fieles en todo momento. Su presencia amorosa y sus gestos llenos de cariño han sido una luz en los días más oscuros. Su alegría y lealtad han sido una inspiración constante para nunca rendirme.

Juntos, han sido mi red de apoyo, sosteniéndome en cada paso del camino. Han compartido mis triunfos y han sido mi consuelo en los momentos de dificultad. Sin su amor y confianza en mí, este logro no hubiera sido posible.

Hoy celebro este éxito con el corazón lleno de gratitud hacia cada uno de ustedes. Agradezco su paciencia, comprensión y ánimo, que han sido el motor que me ha impulsado hacia el logro de mis sueños.

Con el corazón rebosante de agradecimiento, les prometo que seguiré esforzándome para honrar su apoyo y demostrarles que cada paso en este camino ha valido la pena.

Gracias por ser mi familia, mis amigos y mi fiel compañía. Sin ustedes, este logro no habría tenido el mismo significado ni propósito. Les llevo siempre en mi corazón y sé que juntos enfrentaremos cualquier desafío que la vida nos depare.

Con amor y gratitud.

RECONOCIMIENTOS

Estimados maestros e institución,

Quiero expresar mi más sincero reconocimiento y agradecimiento por el invaluable apoyo y orientación que me han brindado a lo largo de mi carrera. Su dedicación y compromiso han sido fundamentales para mi crecimiento académico y profesional.

Gracias a su experiencia y conocimientos, he podido enfrentar con confianza los retos académicos y empresariales que se me presentaron. Cada consejo y sugerencia que me han brindado ha sido de gran valor, guiándome hacia el éxito y el logro de mis metas.

Sus palabras de aliento y motivación han sido un impulso para mantenerme enfocado y perseverar en momentos de dificultad. Siempre estuvieron dispuesto a escucharme, comprender mis inquietudes y ofrecerme soluciones que me ayudaron a superar obstáculos.

Agradezco su paciencia y dedicación en cada una de nuestras reuniones y asesorías. Su compromiso con mi desarrollo profesional ha sido evidente en cada paso que hemos dado juntos. Gracias por creer en mí y por alentarme a alcanzar mi máximo potencial. Es un honor haber contado con su guía a lo largo de mi carrera.

De igual manera quiero reconocer a mi gran amiga Andrea por su apoyo, sus consejos en el tiempo de mi estadía profesional y por siempre apoyarme y creer en mí en los momentos más difíciles.

Con gratitud y admiración.

RESUMEN

El proyecto "Sistema Web de Cursos Internos" tiene como objetivo principal mejorar la capacitación y el desarrollo de los empleados de la Automotriz Bonn S.A DE C.V, a través de una plataforma en línea accesible y efectiva. Mediante este sistema, se busca brindar a las y los colaboradores una manera más conveniente y personalizada de adquirir conocimientos y habilidades relevantes para su desempeño laboral.

El sistema web ofrece una amplia variedad de cursos internos, diseñados para satisfacer las necesidades específicas de cada empleado. Gracias a su naturaleza en línea, los cursos pueden ser accedidos desde cualquier lugar y en cualquier momento, lo que brinda a los empleados la flexibilidad para aprender a su propio ritmo y ajustarse a sus horarios.

El enfoque práctico de los cursos ha demostrado ser altamente efectivo para mejorar la retención de conocimientos y la aplicación directa de lo aprendido en el trabajo diario. Además, el sistema proporciona herramientas para realizar un seguimiento del progreso de cada empleado, permitiendo a los responsables de capacitación y a los gerentes evaluar el rendimiento individual y tomar decisiones informadas.

Este proyecto se espera que genere resultados positivos, incluyendo un aumento en la participación de los empleados en la capacitación, una mayor eficiencia en las tareas laborales y una reducción de costos asociados con la capacitación tradicional. Asimismo, sea fomentando un ambiente de aprendizaje continuo y fortalecer la motivación y compromiso del equipo hacia la empresa.

ÍNDICE

INTRODUCCION.....	1
I.- DESCRIPCION DE LA EMPRESA.....	2
1.1.-Datos Generales.....	2
1.2.- Misión.....	2
1.3.- Visión.....	2
1.4.- Micro-Localización	2
1.5 Antecedentes.....	3
II.- DESCRIPCIÓN DEL PROYECTO	4
2.1.- Planteamiento del Problema	4
2.2.- Justificación	4
2.3.- Objetivos.....	6
2.4.- Alcances y Limitaciones.....	7
2.4.1.- Alcances	7
2.4.2.- Limitaciones	7
2.4.3.- Riesgos del Proyecto.....	8
III.- HIPÓTESIS	9
3.1.- Justificación:.....	9
IV.- DESARROLLO DEL PROYECTO	10
4.1.- Marco Teórico.....	10
4.1.2.- Sistema Web	10
4.1.3.- Sistemas Interactivos.....	11
4.1.4.- Internet	11
4.1.5.- Carga de Trabajos al Sistema Web	13
4.1.6.- Pruebas de Usuario.....	14
4.1.7.- Framework Laravel	15
4.1.8.- PHP	16
4.1.9 María DB	17
4.1.10.- Servidor Linux	18
4.1.11.- Bases de datos Relacionales	19
4.1.12.- Tablas	20
4.1.13.- Consultas.....	21
4.1.14.- Backend.....	22

4.1.15.-	Frontend	23
4.1.16.-	Microservicios	24
4.2.1	Metodología Scrum.....	25
4.2.2	Cronograma de Actividades	26
4.2.3	Product backlog del proyecto	28
	Sprint 1: Diseño y creación de la base de datos.....	28
	sprint 2: Diseño y programación de las primeras interfaces administrador.....	29
	Sprint 3: Diseño y programación de las interfaces administrador.....	34
	sprint 4: Diseño y programación de las interfaces administrador.....	40
	sprint 5: Diseño y programación de las interfaces administrador.....	49
	sprint 6: Diseño y programación de las interfaces administrador.....	63
	Sprint 7: Diseño y programación de las interfaces Inicio cursos internos (ver avances por usuario en PDF), Reporte general.	67
	Sprint 8: Diseño y programación de las interfaces empleado	76
RESULTADOS		88
CONCLUSIONES.....		90
LITERATURA CITADA		91

INDICE DE FIGURAS

Ilustración 1 Macro-Localización Automotriz BONN.....	2
Ilustración 2 Diagrama EDR. Fuente: Elaboración Propia.....	8
Ilustración 3 Sistemas Web. Fuente: https://sommarsolucoes.com.br/	10
Ilustración 4 Sistema LMS. Fuente: https://www.ispring.es/blog/what-is-lms	10
Ilustración 5 Sistemas Interactivos. Fuente: https://exactas.unsj.edu.ar/2016/04/07/sistemasinteractivos/	11
Ilustración 6 Sistemas Interactivos IPO. Fuente: https://sites.google.com/a/uabc.edu.mx/sistemas-interactivos-de-aprendizaje/	11
Ilustración 7 Internet. Fuente: https://s2.pplstatics.com/rc/www/multimedia/	12
Ilustración 8 Internet Para Acceder. Fuente: https://concepto.de/internet/	12
Ilustración 9 Carga de Trabajo. Fuente: https://www.redeszone.net/tutoriales/servidores/balanceador-carga-load-balancer-que-es-funcionamiento/	13
Ilustración 10 Pruebas de Usuarios. Fuente: https://acortar.link/LBv5ps	14
Ilustración 11 framework Laravel. Fuente: https://laravel.com/	15
Ilustración 12 Lenguaje PHP. Fuente: https://www.qualitydevs.com/2021/05/31/que-es-para-que-sirve-php/	16
Ilustración 13 María DB. Fuente: https://mariadb.com/	17
Ilustración 14 Servidor Linux. Fuente: https://e-tinet.com/top-10-distribuicoes-linux-servidores/	18
Ilustración 15 BD Arquitectura. Fuente: http://data-base-management.blogspot.com/	19
Ilustración 16 Tablas BD. Fuente: https://fmhelp.filemaker.com/help/16/fmp/es/	20
Ilustración 17 Back-End. Fuente: https://tuyendung.kfcvietnam.com.vn/blog/lap-trinh-backend	22
Ilustración 18 Frontend. Fuente: https://www.servnet.mx/blog/backend-y-frontend/	23
Ilustración 19 Arquitectura de Microservicios. Fuente: https://www.chakray.com/es/experiencia/microservicios/	24
Ilustración 20 cronograma de actividades	26
Ilustración 21 cronograma de actividades	27
Ilustración 22 modelo de la Base de datos.....	28
Ilustración 23 interfaz login.	29
Ilustración 24 función showLoginForm(LoginController).....	30
Ilustración 25 función logout(LoginController).	30
Ilustración 26 función login(LoginController	31
Ilustración 27 interfaz cursos internos.....	32
Ilustración 28 función index (CursosInternos/CursoController).....	32
Ilustración 29 interfaz cursos internos por curso.	33
Ilustración 30 función show (CursosInternos/CursoController).....	34
Ilustración 31 interfaz creación de cursos internos pagina 1	34
Ilustración 32 interfaz creación de cursos internos pagina 2.	35
Ilustración 33 variables públicas(livewire/wizard).	35
Ilustración 34 función render (livewire/wizard).....	36
Ilustración 35 función firstStepSubmit (livewire/wizard).....	36

Ilustración 36 función secondStepSubmit (livewire/wizard).....	36
Ilustración 37 función threeStepSubmit (livewire/wizard)	37
Ilustración 38 función messages (livewire/wizard).....	37
Ilustración 39 función submitform (livewire/wizard)	38
Ilustración 40 función back y clearForm (livewire/wizard).....	39
Ilustración 41 interfaz creación de lecciones.....	39
Ilustración 42 función show (CursosInternos/LeccionController)	40
Ilustración 43 función store (CursosInternos/LeccionController).	40
Ilustración 44 interfaz creación de contenido	41
Ilustración 45 función show (CursosInternos/ContenidoController)	41
Ilustración 46 función store (CursosInternos/ContenidoController).....	42
Ilustración 47 interfaz edición de contenido	43
Ilustración 48 función update (CursosInternos/ContenidoController).....	44
Ilustración 49 interfaz edición de lección.....	45
Ilustración 50 función update() (cursosinternos/LeccionesController)	46
Ilustración 51 interfaz edición de curso	47
Ilustración 52 función update() (cursosinternos/CursoController)	48
Ilustración 53 interfaz creación del examen	49
Ilustración 54 función store() (cursosinternos/ExamenController)	50
Ilustración 55 función ExamenFinal() (cursosinternos/ExamenController).....	51
Ilustración 56 interfaz edición del examen.	53
Ilustración 57 función update().....	54
Ilustración 58 función actualizar().....	55
Ilustración 59 interfaz simulación del contenido administrador.....	57
Ilustración 60 interfaz simulación del contenido administrador.....	57
Ilustración 61 función ver().....	58
Ilustración 62 interfaz simulación del examen administrador.....	59
Ilustración 63 interfaz simulación del examen administrador.....	59
Ilustración 64 función verEx().....	60
Ilustración 65 función validarExam() y función isAnswerCorrect().....	60
Ilustración 66 interfaz simulación de la calificación administrador.....	62
Ilustración 67 interfaz de vista usuarios.	63
Ilustración 68 interfaz de vista agregar usuarios.	64
Ilustración 69 función store().....	64
Ilustración 70 interfaz de eliminar usuarios.	65
Ilustración 71 función destroyUser().....	66
Ilustración 72 interfaz de inicio cursos internos.....	67
Ilustración 73 función index()	68
Ilustración 74 función getExamenUsuario(\$examen, \$id_usuario)	69
Ilustración 75 función getExamenfinal(\$examen, \$id_usuario)	69
Ilustración 76 función getCursosUsuarios(\$usuarios)	70
Ilustración 77 función para mandar los datos al PDF.....	75
Ilustración 78 interfaz de reporte general.....	76
Ilustración 79 interfaz de inicio empleado.	77

Ilustración 80 función index(\$request).	77
Ilustración 81 función index(\$request).	78
Ilustración 82 interfaz de curso interno empleado.	80
Ilustración 83 función show(\$id).	80
Ilustración 84 interfaz de contenido empleado.	81
Ilustración 85 función ver(\$id).	82
Ilustración 86 interfaz de examen empleado.	83
Ilustración 87 interfaz de examen empleado.	83
Ilustración 88 función verExam(\$id).	84
Ilustración 89 interfaz de preguntas examen empleado.	85
Ilustración 90 función validarExam(\$id).	85
Ilustración 91 función insanswerCorrect (\$par).	86

INTRODUCCION

Actualmente la empresa Automotriz BONN S.A. DE C.V. Cuenta con un Área de Capacitación Administrativa el cual se encarga de Generar Cursos, Capacitaciones para las y los nuevos trabajadores que se incorporan dentro de la empresa y a todo personal Administrativo para mantenerse actualizados en cuanto a lo más reciente en cada respectiva área.

Por lo cual el siguiente documento se plantea el desarrollo del sistema de Cursos internos de la Automotriz BONN S.A DE C.V. donde se explica detalladamente el paso a paso del proceso de desarrollo del sistema y la investigación llevado a cabo para encontrar el mejor entorno de desarrollo del sistema y facilitar al usuario a su fácil comprensión y uso.

Además de la hipótesis planteada y los resultados finales obtenido para comprobar si la hipótesis planteada fue correcta o incorrecta.

I.- DESCRIPCION DE LA EMPRESA

En el siguiente apartado se describen los datos generales tanto de la Automotriz BONN S.A. de C.V, así como del Área de Capacitación Administrativa.

1.1.-Datos Generales.

La Automotriz BONN S.A. de C.V. Es una empresa del Sector Automotriz del Grupo Volkswagen que tiene como objetivo en materia de calidad optimizar sus procesos en la producción de sus productos y servicios y la prevención de la contaminación ambiental, seguridad y salud laboral. Al Igual de Fomentar una actitud de excelencia a todos sus colaboradores y socios comerciales.

1.2.- Misión.

Entusiasmar a nuestros clientes en todo el mundo con automóviles innovadores, confiables y amigables con el medio ambiente, así como con servicios de excelencia, para obtener resultados sobresalientes.

1.3.- Visión.

Somos una empresa exitosa que genera utilidades de manera sustentable. Somos líderes en el mercado mexicano, logrando satisfacer y retener al cliente ofreciendo un servicio excelente. Somos competitivos y confiables en el desarrollo y la producción de vehículos y componentes.

1.4.- Micro-Localización

La Automotriz BONN S.A De C.V, se encuentra ubicada en Prolongación de, Avenida Universidad 801, Ex hacienda Candiani, C.P 68130 Oaxaca de Juárez, Oaxaca



Ilustración 1 Macro-Localización Automotriz BONN.

1.5 Antecedentes

La fábrica de automóviles Volkswagen está vinculada en su origen con el ingeniero austriaco Ferdinand Porsche, quien en 1930 fundó en Stuttgart un negocio con el objeto de fabricar un coche pequeño y barato para Alemania. Porsche era un hombre de origen humilde que quería construir un automóvil que fuera económico, accesible al pueblo, en un momento de crisis en que los fabricantes, en el continente, producían sólo para la clase alta. No existía aún en Europa lo que ya era una realidad en Norteamérica: "un coche para el pueblo", en las palabras dichas por Henry Ford en 1906, al concebir el Modelo T.

En 1932, Porsche diseñó un auto diminuto y compacto, con un motor refrigerado por aire, parecido al que equiparía años después al Volkswagen. Su nombre era Tipus 32. En él ya es posible vislumbrar al VW Sedán. Meses después, pese a las reticencias, la Asociación Alemana de Fabricantes de Automóviles contrató a Porsche para desarrollar un nuevo vehículo financiado por el Estado alemán, un modelo familiar para cuatro personas, con un motor refrigerado por aire, de 7 litros por cada 100 kilómetros.

Así nació, al final de la década de los 30, el Tipus 60, a partir del cual la Daimler-Benz construyó el modelo VW3, que ya tiene los rasgos del Sedán (Beetle en inglés -es decir, Escarabajo). Para impulsar la construcción en masa de ese modelo, se desarrolla la fábrica VW, en el condado de Wolfsburgo. La segunda guerra paró la producción de automóviles Sedán, que sería retomada por los británicos ahí mismo, en Wolfsburgo.

En septiembre de 1948, la Volkswagenwerk GmbH pasó de nuevo a manos de Alemania, bajo la dirección de Heinrich Nordhoff, quien durante 20 años habría de dirigir la empresa con un objetivo claro: "fabricar un solo modelo de automóvil, hacer los menos cambios posibles en su diseño (solo para mejorar la calidad) para no perder su espíritu" En 1950, año en que murió Ferdinand Porsche, Nordhoff llegó a 100 mil vehículos producidos en Wolfsburgo.

Empezó también a fabricar otros modelos (como el que llamamos Kombi). En 1953 inauguró la planta de Sao Paulo, en Brasil; en 1955 llegó a un millón de autos producidos con base en el modelo VW Sedán. En 1968, año de la muerte de Nordhoff, fue construida la fábrica de Puebla, en México, y en 1972, el VW Sedán superó con 15 millones de unidades fabricadas el récord de producción que conservaba el Ford Modelo T, convirtiéndose en el auto más fabricado de la historia.

II.- DESCRIPCIÓN DEL PROYECTO

2.1.- Planteamiento del Problema

El proceso de capacitación para los diferentes administrativos y personal de nuevo ingreso de la Automotriz BONN S.A. de S.V, se realiza de manera física y presencial con la ayuda de los empleados de dicha institución que cuentan con una experiencia mayor, por lo cual todo su proceso de capacitación no es la correcta.

En cada capacitación, el área de Capacitaciones de la Automotriz BONN S.A de S.V, asigna a un cierto grupo de personas capaces de impartir dichas capacitaciones a sus nuevos empleados,

Por lo cual la institución decidió hacer un sistema de cursos internos especialmente para las y los trabajadores de nuevo ingreso y así poder capacitarlos de una mejor manera y cumplir con el estándar de atención que deben de brindar en sus servicios.

2.2.- Justificación

En estos tiempos donde la tecnología se vuelve una parte fundamental para el ser humano, cualquier empresa requiere de tener un control en sus capacitaciones que se realizan, existen muchos sistemas y normas de calidad que ya cuentan con las características que requiere la empresa, sin embargo, son difíciles de operar y su adaptación a la institución requiere de muchos requisitos, por lo cual la institución requiere algo mucho más fácil de operar y que al mismo tiempo que sea lo mejor para así poder implementarlo en todas sus sucursales.

Las empresas en México y en muchos países de Latino América cuentan con departamentos para cada área en específico de la empresa. Cuentan con 3 niveles de supervisión (Dirección, Gerentes, Operadores). Lo que hace cada nivel es lo siguiente:

La Dirección es quien dirige una cierta área de una empresa y define los que la requiere la empresa, lo cual implica hacer lo siguiente:

1. Convocar a Reuniones a los Supervisores o Gerentes.
2. Definir el orden del día.
3. Tener acuerdos en la reunión, la regla es un acuerdo con un supervisor responsable, asignar a ese supervisor un tiempo de entrega y un presupuesto.
4. Hacer la minuta.

5. Dar seguimiento a los acuerdos y los no cubiertos, volver a convocarlos para la siguiente reunión.

El Gerente es la persona responsable de planear y dirigir el trabajo de un grupo de individuos, de monitorear su desempeño y tomar acción correctiva cuando es necesario, lo cual implica hacer lo siguiente:

1. Planificar.
2. Organizar y coordinar a un grupo y evaluar su desempeño.
3. tomar acción correctiva cuando es necesario.
4. Controlar acciones dentro de su grupo de trabajo.
5. Analizar el avance de sus actividades.

Un Operador recibe las tareas que debe de realizar, una vez definida deberá de hacer lo siguiente:

1. Recibirá en su celular a través de un correo electrónico un aviso con las tareas que deberá de realizar.
2. Deberá checar tarjeta de la tarea a realizar, en ese momento empieza a contabilizarse el tiempo.
3. Una vez terminada la tarea deberá de crear el documento que demuestre que ya la terminó, puede ser una foto o un documento y deberá de subir la foto a una carpeta que llamaremos repositorio, en ese momento se pone en pausa el tiempo y se manda un aviso al supervisor (gerente).
4. Si la tarea estuvo terminada. Se checa tarjeta de salida a la tarea asignada y cambia el control a la siguiente tarea.
5. Si la tarea estuvo deficiente, deberá de leer los comentarios y volver a checar tarjeta para corregir lo que estuviera mal, de esta forma cuando lo vuelva a terminar, volverá a subir el entregable que lo demuestre y espera que el supervisor lo califique.

2.3.- Objetivos

2.3.1.- Objetivo general.

Desarrollar un sistema web para la capacitación para los empleados que laboran dentro de la empresa y obtener la mejor capacitación de todo su personal.

2.3.2.- Objetivo específico.

- Programar la página de inicio de sesión para los usuarios (administradores y empleados).
- Programar la página principal del administrador con las opciones de:
 - Puestos
 - Cursos planta
 - Empleados
 - Sucursales
 - Generar reportes
 - Historial
 - Manual de usuario

2.4.- Alcances y Limitaciones.

2.4.1.- Alcances

Desarrollar una aplicación web en donde los empleados y personal administrativo se podrán inscribir en los diferentes cursos internos que se lleven a cabo para una mejor y correcta capacitación en aspectos de mejora para la institución.

Implementar el sistema web que pueda optimizar el proceso de capacitación del área de capacitación de la automotriz BONN S.A De C.V.

La plataforma tendrá las siguientes vistas para los usuarios los cuales son:

1. **Vista administrador:** Podrá visualizar los diferentes cursos internos que se lleven a cabo, agregar, modificar los cursos, dar de alta y baja a los empleados, además podrá generar informes de cada curso donde podrá visualizar la cantidad de usuarios inscritos y sus progresos.
2. **Vista Empleado:** En él se podrá visualizar los diferentes cursos internos, en los cuales está inscrito, podrá ingresar al sistema mediante su usuario (Otorgado por el área) y Contraseña (Generada por el área). Este usuario tendrá restringida la vista a las pantallas del administrador ya que estas almacenaran información confidencial.

A continuación, se muestra las funcionalidades

1. Historial del empleado respecto a los cursos internos tomados, empezar/continuar con el seguimiento de las actividades y desempeño del usuario durante el tiempo que duren los cursos a los que está inscrito.
2. El usuario podrá tomar los cursos asignados por el administrador.
3. Generación de certificado de culminación del curso, si este cuenta con el porcentaje mínimo de acuerdo con el reglamento del área de capacitación.

2.4.2.- Limitaciones

A continuación, se enlistan los factores externos al proyecto los cuales pueden limitar en parte el diseño y desarrollo, cabe mencionar que estas no pueden llegar a estar bajo control del equipo de desarrollo.

1. **Tiempo de Desarrollo:** El tiempo de desarrollo planteado para este proyecto es de un total de 3 meses, este tiempo se considera como un factor limitante, debido a que los integrantes tienen otras tareas y responsabilidades fuera del proyecto planteado con anterioridad.

- 2. Cliente:** La comunicación del equipo de desarrollo con el cliente durante todo el periodo de desarrollo de proyecto es sumamente esencial, importante y este mismo indicará el nivel de éxito de este.

2.4.3.- Riesgos del Proyecto.

Los riesgos se califican de acuerdo con lo siguiente:

1. Posibles cambios en el criterio de evaluación del proyecto.
2. Modificación de lenguaje de programación, plataforma de desarrollo.
3. Requerimientos, limitaciones y alcances no establecidos correctamente.
4. Informes pocos claros sobre el avance del proyecto.
5. Problemas médicos.
6. Falta de copias de seguridad de todo el proyecto.
7. Rechazo del cliente final del avance del proyecto final.
8. Pérdidas de partes del proyecto por daño de terceros en caso de robo o extravió.
9. Retrasos en las actividades orientadas al desarrollo del proyecto.
10. Insuficiente administración de los riesgos.
11. No realizar un buen seguimiento en el protocolo de normas.

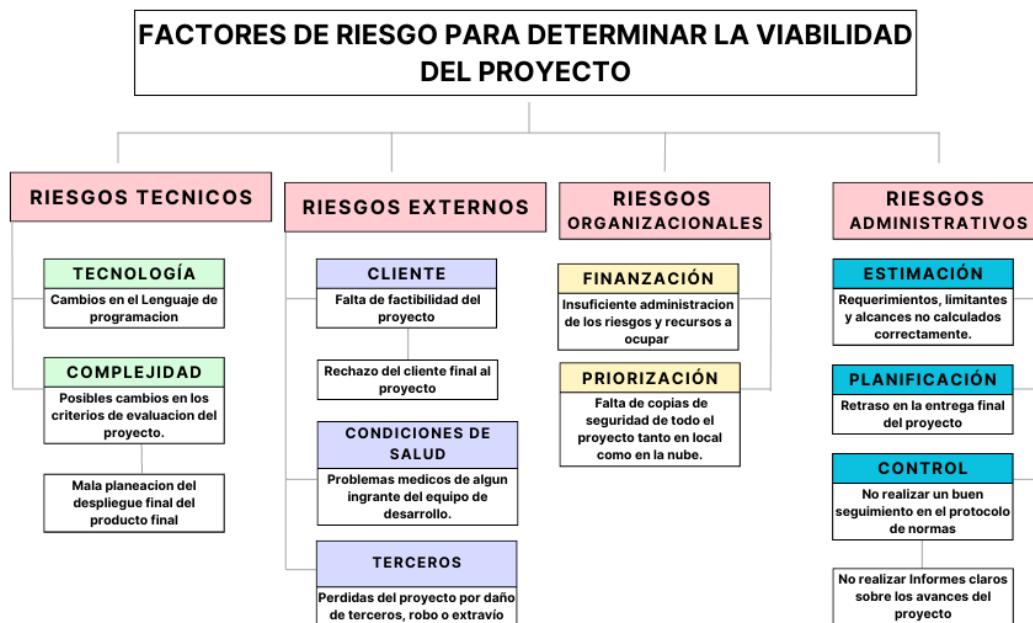


Ilustración 2 Diagrama EDR. Fuente: Elaboración Propia

III.- HIPÓTESIS

Implementar un sistema de cursos internos para la capacitación de los nuevos empleados de la Automotriz BONN S.A. de S.V, en lugar del proceso físico y presencial actual, mejorará la eficacia y calidad de la capacitación, permitiendo a los trabajadores adquirir los conocimientos y habilidades necesarios de manera más efectiva y cumplir con los estándares de atención requeridos en los servicios de la institución.

3.1.- Justificación:

La justificación planteada que el proceso de capacitación actual de la Automotriz BONN S.A. de S.V, el cual se realiza de manera física y presencial con la ayuda de empleados experimentados, no es el método adecuado. Se argumenta que la capacitación impartida por el personal de la empresa puede carecer de estructura y consistencia, lo que podría resultar en una formación deficiente para los nuevos empleados.

La hipótesis propone que implementar un sistema de cursos internos para la capacitación resolvería esta deficiencia. Este sistema de cursos internos podría ser diseñado y estructurado de manera que se cumplan los objetivos de capacitación de manera más efectiva y se brinden los conocimientos y habilidades necesarios para desempeñar las funciones del puesto de manera adecuada.

Además, se sugiere que, al establecer cursos internos, se puede garantizar una mayor uniformidad en la capacitación, ya que el contenido y la metodología de enseñanza podrían ser estandarizados y adaptados específicamente a los requisitos de la Automotriz BONN S.A. de S.V. Esto permitiría una mayor coherencia en los conocimientos y habilidades adquiridos por los nuevos empleados, asegurando que todos estén capacitados de manera uniforme y cumplan con los estándares de atención establecidos por la empresa.

En resumen, la hipótesis plantea que reemplazar el proceso de capacitación físico y presencial actual con un sistema de cursos internos mejoraría la calidad y eficacia de la capacitación de los nuevos empleados de la Automotriz BONN S.A. de S.V, permitiéndoles adquirir los conocimientos y habilidades necesarios de manera más efectiva y cumplir con los estándares de atención requeridos en los servicios de la institución.

IV.- DESARROLLO DEL PROYECTO

4.1.- Marco Teórico

4.1.2.- Sistema Web

Se denomina sistema web a aquellas aplicaciones de software que pueden utilizarse accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Las aplicaciones web son muy usadas hoy en día, debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo. Estos sistemas se basan en una arquitectura cliente-servidor, donde el cliente es el navegador web y el servidor es la parte centralizada que maneja las solicitudes y procesa los datos. (Craig A. & D, 2014).

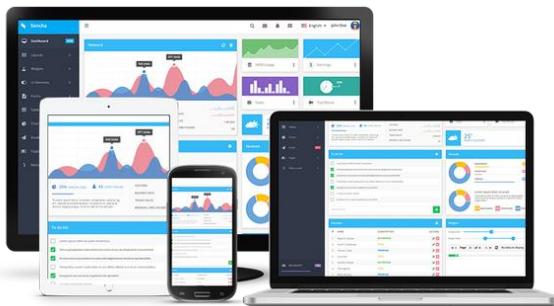


Ilustración 3 Sistemas Web. Fuente: <https://sommarsolucoes.com.br/>

El Sistema de Gestión de Aprendizaje se ha convertido en una herramienta increíblemente poderosa para las organizaciones que buscan mejorar el rendimiento y la retención de su fuerza de trabajo. La mayoría de los sistemas de gestión de aprendizaje son soluciones de software basados en la nube que las empresas usan como su herramienta fundamental para la gestión de sus programas de capacitación. El software LMS es usualmente la tecnología base usada por los departamentos de capacitación y desarrollo de las empresas. (SistemLms, 2022).



Ilustración 4 Sistema LMS. Fuente: <https://www.ispring.es/blog/what-is-lms>

4.1.3.- Sistemas Interactivos

Los sistemas interactivos son aquellos que permiten la comunicación bidireccional entre el usuario y el sistema. Estos sistemas proporcionan una interfaz que permite a los usuarios interactuar con el sistema, enviar comandos, recibir respuestas y realizar acciones. La interactividad es esencial para proporcionar una experiencia de usuario efectiva y satisfactoria, permitiendo una mayor participación y una retroalimentación inmediata (Dix et al., 2004, p. 101).



Ilustración 5 Sistemas Interactivos. Fuente: <https://exactas.unsj.edu.ar/2016/04/07/sistemasinteractivos/>

Esta obra se enmarca en el contexto de la disciplina conocida con el nombre de Interacción persona-ordenador (IPO), la cual está relacionada con el diseño, la evaluación y la implementación de sistemas interactivos para que sean utilizados por personas, y con el estudio de los fenómenos más importantes con los que está relacionada. (TecnoAccesible, 2005, p.137).



Ilustración 6 Sistemas Interactivos IPO. Fuente: <https://sites.google.com/a/uabc.edu.mx/sistemas-interactivos-de-aprendizaje/>

4.1.4.- Internet

Internet es una red mundial de computadoras interconectadas que permite la comunicación y el intercambio de información entre usuarios. Se basa en un conjunto de protocolos de comunicación, como el Protocolo de Internet (IP), que facilita la transmisión de datos de manera eficiente y confiable. Internet ha transformado radicalmente la forma en que las personas se comunican accede a la información y realizan actividades en línea. (Castells, 2001, p. 28).



Ilustración 7 Internet. Fuente: <https://s2.pplstatics.com/rc/www/multimedia/>

Para acceder a los billones de sitios web disponibles en la gran red de redes, que conocemos como la Internet, **se utilizan los** navegadores web (software), siendo algunos de los más utilizados Google Chrome, Internet Explorer, Mozilla Firefox, y Safari, todos desarrollados por distintas compañías tecnológicas. (Anderson B & H, 2018).

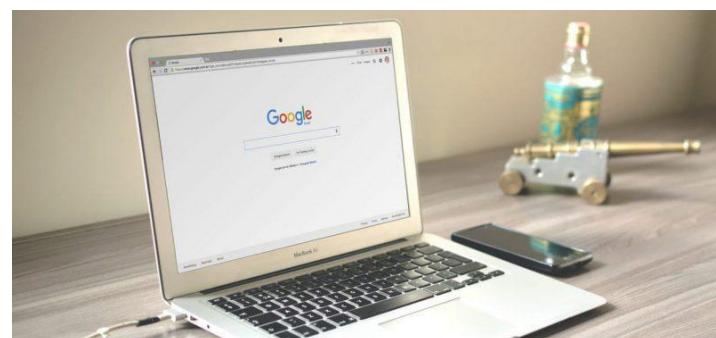


Ilustración 8 Internet Para Acceder. Fuente: <https://concepto.de/internet/>

4.1.5.- Carga de Trabajos al Sistema Web

La carga de trabajos al sistema web se refiere al proceso mediante el cual los usuarios envían y suben archivos o tareas al sistema a través de una interfaz en línea. Este proceso permite la entrega y revisión de trabajos, como documentos, imágenes o cualquier otro tipo de archivo, dentro del entorno del sistema web. La carga de trabajos es fundamental en diversos contextos, como plataformas educativas o sistemas de gestión de proyectos, ya que permite a los usuarios colaborar y compartir contenido. (Laudon & Laudon, 2020, p. 240).



Ilustración 9 Carga de Trabajo. Fuente: <https://www.redeszone.net/tutoriales/servidores/balanceador-carga-load-balancer-que-es-funcionamiento/>

Es importante incluir el trabajo que un sistema está haciendo en segundo plano. Por ejemplo, si un sistema contiene sistemas de archivos que están montados en NFS y a los que otros sistemas acceden con frecuencia, el manejo de esos accesos es probablemente una fracción significativa de la carga de trabajo global, aunque el sistema no sea oficialmente un servidor (IBM, v.7.3, 2023).

Una carga de trabajo que se ha estandarizado para permitir comparaciones entre sistemas distintos se denomina *benchmarking* o referencia. Sin embargo, pocas cargas de trabajo reales duplican los algoritmos exactos y el entorno de una referencia. “Incluso las referencias estándares de la industria que originalmente se derivaron de aplicaciones reales se han simplificado y homogeneizado para que se puedan portar a una amplia variedad de plataformas de hardware” (IBM, v.7.3, 2023).

4.1.6.- Pruebas de Usuario

"Las pruebas de usuario, también conocidas como pruebas de usabilidad, son una técnica utilizada para evaluar la facilidad de uso y la eficiencia de un sistema o aplicación desde la perspectiva del usuario final. Estas pruebas implican observar a los usuarios mientras interactúan con el sistema y recopilar datos sobre su desempeño, reacciones y niveles de satisfacción. El objetivo es identificar problemas de diseño, dificultades de uso y áreas de mejora, para así mejorar la experiencia del usuario y la calidad del sistema" (Nielsen & Molich, 1990, p. 6).

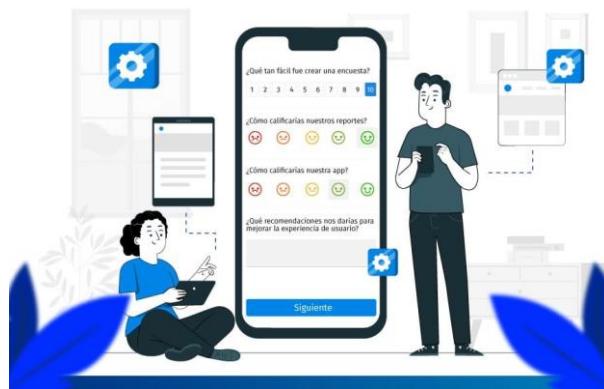


Ilustración 10 Pruebas de Usuarios. Fuente: <https://acortar.link/LBv5ps>

El objetivo de este proceso es evaluar la usabilidad del sitio web o aplicación, revelar áreas de confusión, descubrir oportunidades de mejora y, en última instancia, decidir si el producto está listo para ser lanzado para usuarios reales o no (Alejandro G. 2020, p.334)

Este tipo de prueba con usuarios es excepcionalmente importante en el caso del desarrollo de nuevos productos o nuevas actualizaciones de diseño. Sin ella, puedes quedarte atascado con un proceso de diseño de UX que los miembros de tu equipo entienden, pero tu público objetivo no. (Alejandro G. 2020, p.345)

4.1.7.- Framework Laravel

"Laravel es un framework de desarrollo de aplicaciones web de código abierto basado en PHP. Proporciona una estructura sólida y modular que agiliza el proceso de desarrollo y facilita la creación de aplicaciones web robustas y escalables. Laravel se destaca por su elegante sintaxis, su amplia gama de funcionalidades y su enfoque en la simplicidad y la legibilidad del código. Con su arquitectura MVC (Modelo-Vista-Controlador) y su amplia comunidad de desarrolladores, Laravel se ha convertido en una opción popular para el desarrollo de aplicaciones web modernas" (Otwell, 2011, p. 10).

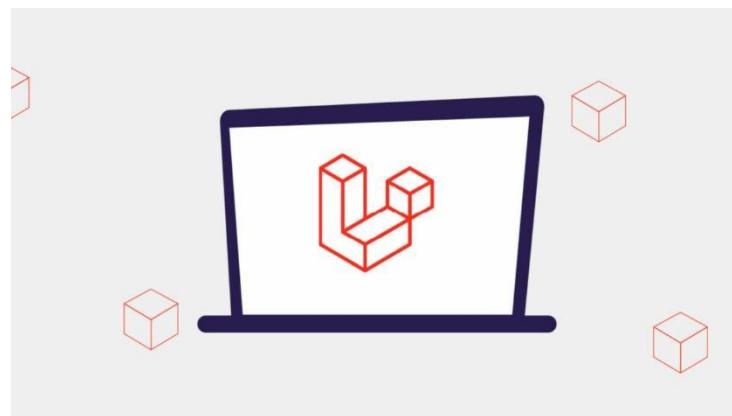


Ilustración 11 framework Laravel. Fuente: <https://laravel.com/>

Laravel es increíblemente escalable. Gracias a la naturaleza amigable con el escalado de PHP y al soporte incorporado de Laravel para sistemas de caché distribuidos rápidos como Redis, el escalado horizontal con Laravel es muy fácil. De hecho, las aplicaciones de Laravel se han escalado fácilmente para manejar cientos de millones de solicitudes por mes.

4.1.8.- PHP

PHP es un lenguaje de programación ampliamente utilizado en el desarrollo de aplicaciones web. Es un lenguaje interpretado, de código abierto y con una sintaxis similar a la de C y Perl. PHP se ejecuta en el lado del servidor y puede generar contenido dinámico en HTML. Es especialmente adecuado para el desarrollo rápido de aplicaciones web y permite la interacción con bases de datos y la manipulación de datos en línea (The PHP Group, 2021, p. 13).



Ilustración 12 Lenguaje PHP. Fuente: <https://www.qualitydevs.com/2021/05/31/que-es-para-que-sirve-php/>

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. (The PHP Group, 2021, p. 23)

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP. En unas pocas horas podrá empezar a escribir sus primeros scripts. (The PHP Group, 2021, p.25)

4.1.9 María DB

María DB es un sistema de gestión de bases de datos relacional, derivado de MySQL. Es una alternativa de código abierto y compatible con MySQL que ofrece un alto rendimiento, estabilidad y seguridad. María DB es ampliamente utilizado en aplicaciones web y proporciona una solución confiable para el almacenamiento y la manipulación de datos en línea (MariaDB Foundation, 2021, p. 8).



Ilustración 13 María DB. Fuente: <https://mariadb.com/>

MariaDB funciona bien con las aplicaciones web y las plataformas de comercio electrónico, y sus mecanismos de subprocessos múltiples le permiten gestionar cargas más altas que otros sistemas de bases de datos. Debido a su modelo de subprocessos múltiples y su alto rendimiento, MariaDB puede adaptarse para permitir que su aplicación o sitio gestione los picos de tráfico o el rápido crecimiento del negocio. (MariaDB Foundation, 2021, p. 12).

MariaDB viene con un cliente de línea de comandos MySQL nativo que admite el uso interactivo y no interactivo. Cuando se usa de forma interactiva, los resultados de la consulta se muestran en un formato de tabla ASCII, y cuando se usa de forma no interactiva (es decir, como filtro), los resultados se presentan en un formato separado por tabuladores. El formato de salida se puede cambiar mediante las opciones de comando. (MariaDB Foundation, 2021, p. 15).

4.1.10.- Servidor Linux

Linux es un sistema operativo de código abierto basado en Unix que es ampliamente utilizado como plataforma de servidor. Proporciona una base sólida y estable para la ejecución de aplicaciones web y ofrece características avanzadas de seguridad y rendimiento. Los servidores Linux son populares en entornos de alojamiento web y son compatibles con una amplia variedad de aplicaciones y tecnologías (Matthew & Stones, 2020, p. 45).



Ilustración 14 Servidor Linux. Fuente: <https://e-tinet.com/top-10-distribuicoes-linux-servidores/>

Los servidores Linux ofrecen una base sólida para los centros de datos y los entornos de cargas de trabajo empresariales complejos, desde los servidores dedicados (bare metal) hasta las máquinas virtuales, los contenedores y las nubes públicas y privadas. También pueden guiar su proceso de transformación digital y el desarrollo de las aplicaciones en la nube, ya que le ofrecen la capacidad para aumentar la productividad, prestar servicios más rápido e incorporar las innovaciones de software, como la automatización de la configuración, las nubes y los contenedores. (Matthew & Stones, 2020, p. 47).

4.1.11.- Bases de datos Relacionales

Las bases de datos relacionales son un tipo de sistema de gestión de bases de datos que organizan y estructuran los datos en tablas, estableciendo relaciones entre ellas. Cada tabla representa una entidad o relación, y las filas y columnas contienen los registros y los atributos respectivamente. Las bases de datos relacionales utilizan el lenguaje SQL para manipular y consultar los datos. Este enfoque ofrece flexibilidad, integridad y eficiencia en la gestión de grandes volúmenes de información (Elmasri & Navathe, 2020, p. 81).

“Esto hace refería a una cantidad de dato que se encuentran relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Las bases de datos son el producto de la necesidad humana de almacenar la información, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente”

Una base de datos es una herramienta para recopilar y organizar información. Las bases de datos pueden almacenar información sobre personas, productos, pedidos u otras cosas. Muchas bases de datos comienzan como una lista en una hoja de cálculo o en un programa de procesamiento de texto.

A medida que la lista aumenta su tamaño, empiezan a aparecer redundancias e inconsistencias en los datos. Cada vez es más difícil comprender los datos en forma de lista y los métodos de búsqueda o extracción de subconjuntos de datos para revisión son limitados. Una vez que estos problemas comienzan a aparecer, una buena idea es transferir los datos a una base de datos creada con un sistema de administración de bases de datos. (Database systems: A practical approach to design, implementation, and management (6th ed.), 2014)

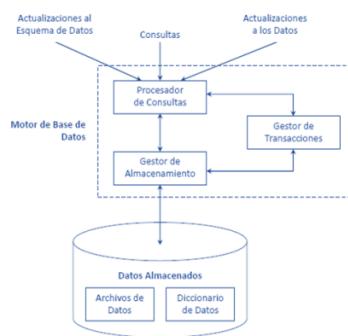


Ilustración 15 BD Arquitectura. Fuente: <http://data-base-management.blogspot.com/>

El término "sprint" se utiliza comúnmente en el contexto de la metodología ágil de desarrollo de software, y su definición varía ligeramente según el autor.

Según Jeff Sutherland, uno de los creadores de Scrum (un marco de trabajo ágil), un sprint es un período de tiempo fijo, generalmente de dos a cuatro semanas, durante el cual un equipo de desarrollo trabaja para entregar un conjunto de funcionalidades valiosas y completas.

Por otro lado, según Mike Cohn, autor y experto en metodologías ágiles, un sprint es un ciclo de trabajo de tiempo fijo, generalmente de dos a cuatro semanas, durante el cual el equipo de desarrollo construye y entrega una funcionalidad potencialmente entregable.

4.1.12.- Tablas

En el contexto de las bases de datos relacionales, una tabla es una estructura de datos que organiza la información en filas y columnas. Cada fila representa un registro o entidad individual, y cada columna almacena un atributo o campo específico. Las tablas permiten almacenar y organizar grandes cantidades de datos de manera estructurada, lo que facilita la manipulación y el acceso eficiente a la información (Elmasri & Navathe, 2020, p. 110).

Una tabla de base de datos es similar en apariencia a una hoja de cálculo en cuanto a que los datos se almacenan en filas y columnas. Por ende, es bastante fácil importar una hoja de cálculo en una tabla de base de datos. La principal diferencia entre almacenar los datos en una hoja de cálculo y almacenarlos en una base de datos es la forma en la que están organizados los datos.

Cada fila de una tabla se denomina registro. En los registros se almacena información. Cada registro está formado por uno o varios campos. Los campos equivalen a las columnas de la tabla. Los campos deben designarse como un determinado tipo de datos, ya sea texto, fecha u hora, número o algún otro tipo.

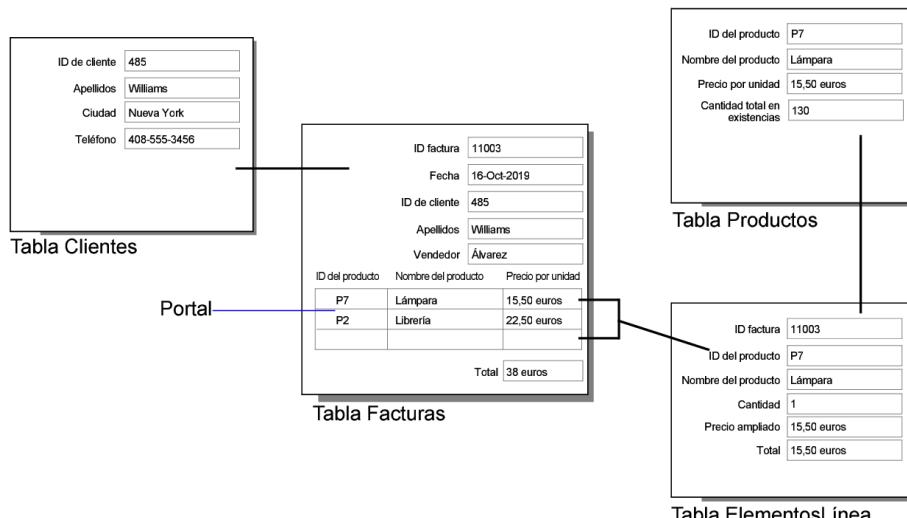


Ilustración 16 Tablas BD. Fuente: <https://fmhelp.filemaker.com/help/16/fmp/es/>

4.1.13.- Consultas

Las consultas en bases de datos son comandos o instrucciones que se utilizan para recuperar información de una base de datos. Permiten realizar búsquedas, filtrar datos y combinar información de diferentes tablas para obtener resultados específicos. Las consultas se basan en el lenguaje de consulta estructurado (SQL) y proporcionan una forma poderosa y flexible de interactuar con los datos almacenados en una base de datos relacional (Ramakrishnan & Gehrke, 2003, p. 75).

Consiste en una petición de información a una base de datos. La petición debe venir en una tabla de la base de datos o en una combinación de tablas utilizando un código conocido como lenguaje de consulta. De este modo, el sistema puede entender y procesar la consulta en cuestión. (Manual SQL Server , 2019)

Las consultas pueden realizar diversas funciones en una base de datos. La función más común es recuperar datos específicos de las tablas. Los datos que quiere ver generalmente se encuentran distribuidos en varias tablas y las consultas le permiten verlos en una sola hoja única de datos

En este listado se muestran los comandos más populares para realizar consultas.

- **SELECT:** Obtener datos de la base de datos. Este es uno de los comandos más usados ya que la mayoría de las peticiones comienzan con una consulta select.
- **AND:** Combina datos de una o de más tablas.
- **CREATE TABLE:** Crea diferentes tablas y especifica el nombre de cada una de las columnas que contiene.
- **ORDER BY:** Este ordena los resultados de los datos numéricamente o alfabéticamente.
- **SUM:** Resume los datos de una columna concreta.
- **UPDATE:** Modifica las filas ya existentes en una tabla.
- **INSERT:** Inserta nuevos datos filas a una tabla que ya existe.
- **WHERE:** Filtra los datos y obtiene su valor en función de una condición establecida. (Manual SQL Server , 2019)

4.1.14.- Backend

El backend, también conocido como el lado del servidor, es la parte de una aplicación web que se encarga de la lógica y el procesamiento detrás de escena. Esto incluye el manejo de solicitudes, la gestión de bases de datos, la autenticación de usuarios, la ejecución de algoritmos y la generación de respuestas dinámicas. El backend se desarrolla utilizando lenguajes de programación como PHP, Python o Java, y se comunica con el frontend a través de API para presentar los datos y la funcionalidad al usuario (Gosselin, 2019, p. 22).

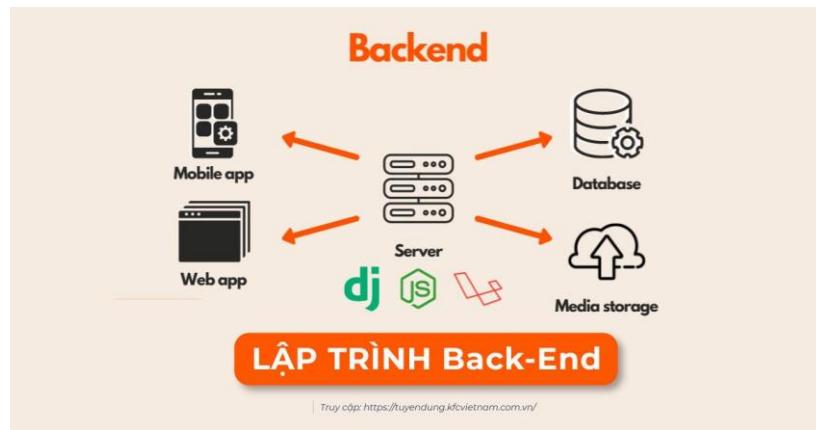


Ilustración 17 Back-End. Fuente: <https://tuyendung.kfcvietnam.com.vn/blog/lap-trinh-backend>

Este también se conoce como el lado del servidor, ya que todas las solicitudes del usuario son procesadas por el servidor antes de que se le entreguen los resultados al usuario. En contraste, el frontend se refiere a la parte de una aplicación de software que el usuario ve y con la que interactúa directamente, como la interfaz de usuario y la presentación de información. (The complete guide to E-commerce technology. Cengage Learning EMEA., 2003, pág. 80)

4.1.15.- Frontend

El frontend, también conocido como el lado del cliente, es la parte de una aplicación web que los usuarios ven y con la que interactúan directamente. Incluye la interfaz de usuario, la presentación de datos y la interacción del usuario. El frontend se desarrolla utilizando tecnologías web como HTML, CSS y JavaScript, y su objetivo principal es brindar una experiencia de usuario atractiva y amigable, garantizando la usabilidad y la accesibilidad (Terry, 2020, p. 78).

Es esa parte de la página con la que interactúan los usuarios de la misma, es todo el código que se ejecuta en el navegador de un usuario, al que se le denomina una aplicación cliente, es decir, todo lo que el visitante ve y experimenta de forma directa. (Terry, 2020, p. 80).



Ilustración 18 Frontend. Fuente: <https://www.servnet.mx/blog/backend-y-frontend/>

4.1.16.- Microservicios

Los microservicios son una arquitectura de software en la que una aplicación se descompone en servicios pequeños, autónomos e independientes que se comunican entre sí a través de API. Cada microservicio se enfoca en una función o tarea específica y puede ser desarrollado, implementado y escalado de forma individual. Esta arquitectura permite la flexibilidad, el mantenimiento y la escalabilidad eficientes de los sistemas, facilitando la colaboración y el despliegue continuo (Newman, 2015, p. 92).

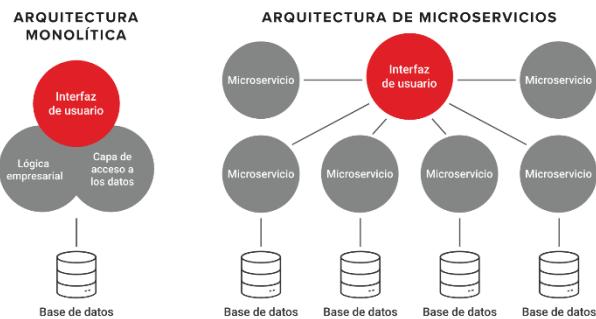


Ilustración 19 Arquitectura de Microservicios. Fuente: <https://www.chakray.com/es/experiencia/microservicios/>

Una regla importante de la arquitectura de microservicios es que cada microservicio debe ser propietario de sus datos de dominio y su lógica. Al igual que una aplicación completa posee su lógica y sus datos, cada microservicio debe poseer su lógica y sus datos en un ciclo de vida autónomo, con implementación independiente por microservicio. (Fowler, M. , 2014).

4.2 Metodología

4.2.1 Metodología Scrum.

El proyecto se realizará utilizando un marco de desarrollo ágil, tipo Scrum pues es la metodología que más se adapta a nuestro proyecto y la que nos permitirá asegurar el cumplimiento de los objetivos para que el proyecto se desarrolle de la mejor manera posible. (Schwaber, K., & Sutherland, 2023).

Los sprints elaborados durante este proyecto fueron 8 ya que estos se dividieron por fechas como se muestra en el cronograma de actividades elaborado antes de comenzar con la producción del mismo.

En la elaboración de este proyecto los roles ocupados fueron el back-end y front-end. Para el back-end se utilizó MariaDB como base de datos en su versión 10.1.2 y por el lado del front-end se utilizó el framework de Laravel en su versión 10 como se explica a continuación.

4.2.2 Cronograma de Actividades

Actividad programada	Semanas														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Definición del proyecto	1	2	3	4	5										
Datos generales de la empresa.	x														
Antecedentes de la empresa	x														
Planteamiento del problema.		x													
Justificación.		x													
Planteamiento de objetivos.			x												
Hipótesis.			x												
Marco teórico.				x											
Metodología.					x										
Desarrollo del proyecto.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Desarrollo del modelo de la base de datos y prueba de consultas en la base de datos						x									
Modelado de las interfaces en figma							x								
Revisión y observaciones de las interfaces							x								
Desarrollo de las interfaces en laravel								x							
Desarrollo del proyecto.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Integración de los dos sistemas cursos planta y internos								x							
Pruebas para verificar el funcionamiento del sistema									x						
Subir el sistema al servidor										x					
Conclusión del proyecto	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Análisis de resultados.											x				

Ilustración 20 cronograma de actividades

Redacción de conclusiones.											X	X		
Oficio de dictamen													x	
Impresión y entrega de memoria														x

Ilustración 21 cronograma de actividades

4.2.3 Product backlog del proyecto

Sprint 1: Diseño y creación de la base de datos.

En el desarrollo de este sprint se llevó a cabo el diseño y la creación de la base de datos en donde se tomó en cuenta los datos requeridos por la Consultora de Procesos y la Gerente del área de TIC.

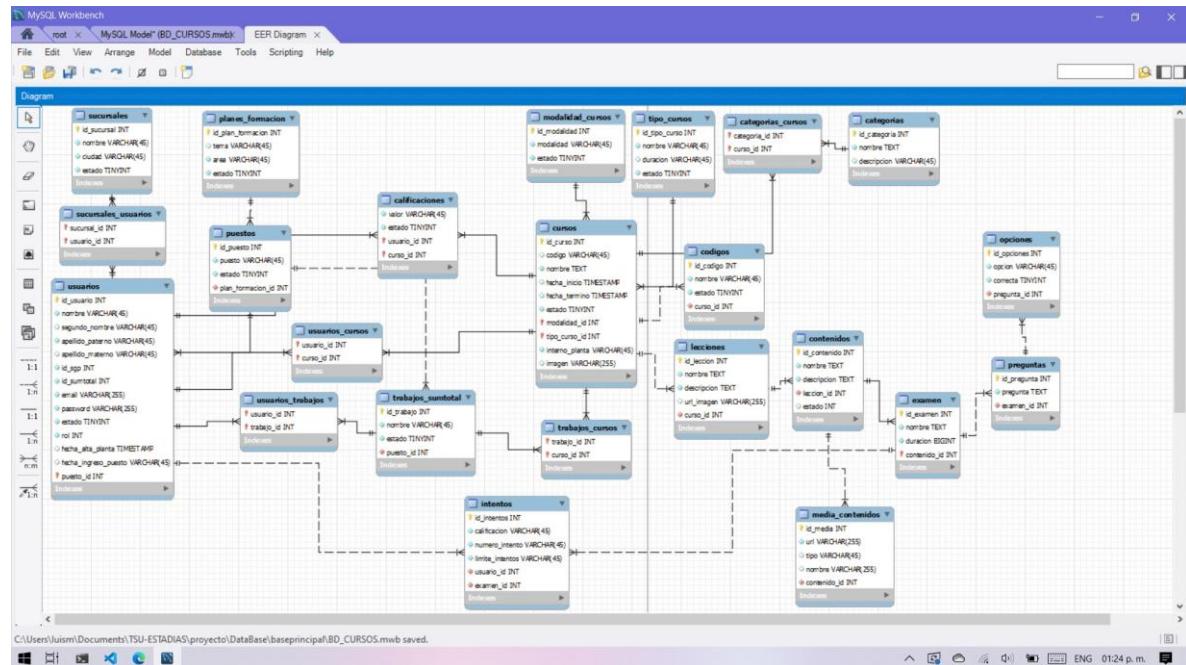


Ilustración 22 modelo de la Base de datos.

sprint 2: Diseño y programación de las primeras interfaces administrador.

El segundo sprint se basó en la creación y programación y revisión de las interfaces:

- Login
- Vista inicio administrador
- Vista curso interno individual
- Vista curso interno

Desarrollo de la interfaz del login:

- Diseño:

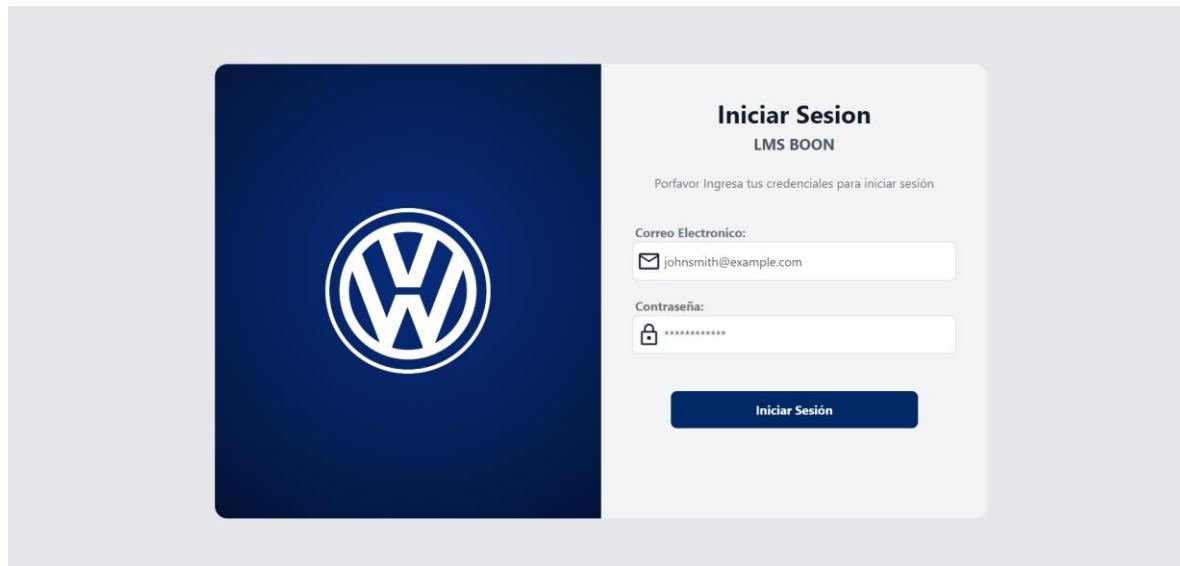


Ilustración 23 interfaz login.

- Funcionabilidad:

El login cuenta con los campos correo electrónico y contraseña el cual se validarán con los datos que se encuentran almacenadas en la base de datos y si los datos ingresados por el usuario son idénticas a las almacenadas se validará y el usuario podrá iniciar sesión de acuerdo con el rol asignado por el administrador.

- Programación:

El login está programado a través de un controlador (LoginController) de la siguiente manera:

- Función showLoginForm():

```
public function showLoginForm()
{
    /**
     * retorna ala vista del login para
     * que el usuario se autentique
     */
    return view('auth.login');
}
```

Ilustración 24 función showLoginForm(LoginController).

Esta función retorna al usuario ala vista login por defecto donde el usuario podrá autenticarse con su correo y contraseña.

- Función logout():

```
public function logout(Request $request)
{
    Auth::guard('web')->logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();

    // // Evitar que el usuario retroceda en el historial
    // $request->headers->set('Cache-Control', 'no-store, no-cache, must-revalidate, max-age=0');
    // $request->headers->set('Pragma', 'no-cache');
    // $request->headers->set('Expires', 'Sat, 01 Jan 2000 00:00:00 GMT');

    return to_route('home')
        ->with('status', 'Cerrando sesion');
}
```

Ilustración 25 función logout(LoginController).

Esta función permite que el usuario cierre su sesión y no pueda volver o retroceder en el historial de navegación hasta que no vulva a autenticarse en el login.

- Función login():

```

public function login(Request $request)
{
    $credentials = $request->validate([
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string']
    ]);

    /**
     * va a verificar email y password
     * con la de la base de datos
     * y devuelve un booleano, como segundo parametro
     * recibe un boolean para indicarle si
     * queremos recordar la sesion o no
     * para eso utilizamos el checkbox de recordarme
     */
    if (!Auth::attempt($credentials)) {
        throw ValidationException::withMessages([
            'email' => __('auth.failed')
        ]);
    }

    $user = Auth::user();
    if (!$user->estado) {
        // El usuario no está activo, mostrar mensaje de error
        return back()->withErrors([
            'email' => 'El usuario no está activo.',
        ]);
    }

    if (auth()->user()->role == 0) {
        // retornar a vista administrador
        return to_route('home');
    }

    $request->session()->regenerate();

    return redirect()->intended()
        ->with('status', 'Inicio de sesion correcto');
}

```

Ilustración 26 función login(LoginController)

Esta función validara los datos ingresados por el usuario (correo y contraseña) en el login el cual serán enviados a través de un formulario con el método POST, en donde posteriormente se verificarán si concuerdan con la base de datos en el cual devolverá un dato booleano el cual si es verdadero ingresara a la validación de acuerdo con su rol, si su rol es “0” (administrador = 0 y empleado = 1) redireccionara a la vista administrador y si no redireccionara a la vista empleado.

Desarrollo de la interfaz de la vista curso interno:

- Diseño:

The screenshot shows a web application interface for managing internal courses. At the top, there's a header with the title 'CATÁLOGO DE CURSOS'. Below the header, there are three buttons: 'Agregar curso' (Add course), 'Usuarios internos' (Internal users), and 'Reportes' (Reports). A search bar with placeholder text 'Buscar por Categoría, Nombre, Código' and a magnifying glass icon is located on the right. The main area is titled 'Cursos' and contains a table with course data. The columns are: IMAGEN (Thumbnail), NOMBRE COMPLETO DEL CURSO (Full course name), CÓDIGO DEL CURSO (Course code), CATEGORÍA DEL CURSO (Category), CANTIDAD DE USUARIOS INSCRITOS (Number of registered users), and CONFIGURACIÓN (Configuration). The table lists five courses: 'HHHJUG', 'PRUEBA DE LIVERIWE', 'CURSO DE INDUCCION VOLKSWAGEN 3', 'CURSO DE INDUCCION VOLKSWAGEN', and 'PRUEBA FINAL ADMINISTRADOR'. Each row has a configuration button. At the bottom of the table, it says 'Mostrando 1 a 8 de 5 resultados' and includes navigation arrows.

Ilustración 27 interfaz cursos internos.

- Funcionabilidad:

La vista cursos internos tendrá las siguientes funcionalidades: visualizar los diferentes cursos que ya han sido dados de alta mostrando su información más relevante como su imagen, nombre, código, categoría, fecha inicio y termino, de igual manera contará con un botón en el cual el administrador podrá dar de alta nuevos cursos, y así mismo contará con un buscador el cual podrá buscar un curso por nombre, código o categoría.

- Programación:

La vista cursos internos está programado a través de un controlador (CursosInternos/CursosController) de la siguiente manera:

- Función index():

```
public function index(Request $request)
{
    // $cursos = Curso::where('interno_planta', '=', 1)->orderBy('id_curso', 'desc')->paginate('10');
    $buscar = $request->buscador;
    $cursos = Curso::where(function ($query) use ($buscar) {
        $query->where('nombre', 'like', $buscar . "%")
            ->orWhere('codigo', 'like', $buscar . "%")
            ->orWhere('fecha_inicio', 'like', $buscar . "%");
    })
        ->where('interno_planta', '=', 1)
        ->orWhereHas('categoria', function ($query) use ($buscar) {
            $query->where('nombre', 'like', $buscar . "%");
        })
        ->orderBy('id_curso', 'desc')
        ->paginate(8);
    $autores = User::all();
    $categorias = Categoria::all();
    return view('cursosinternos.cursos.catalogo', compact('cursos', 'autores', 'categorias'));
}
```

Ilustración 28 función index (CursosInternos/CursoController).

Esta función permite renderizar todos los datos necesarios de los cursos internos a la vista para que el usuario administrador pueda visualizarlos, de igual manera en esta función es donde se implementa el buscador donde obtenemos el dato que queremos buscar el cual lo trabajamos con un “request” y posteriormente si encuentra alguna coincidencia lo retornamos ala vista.

Desarrollo de la interfaz del Vista curso interno individual:

- Diseño:

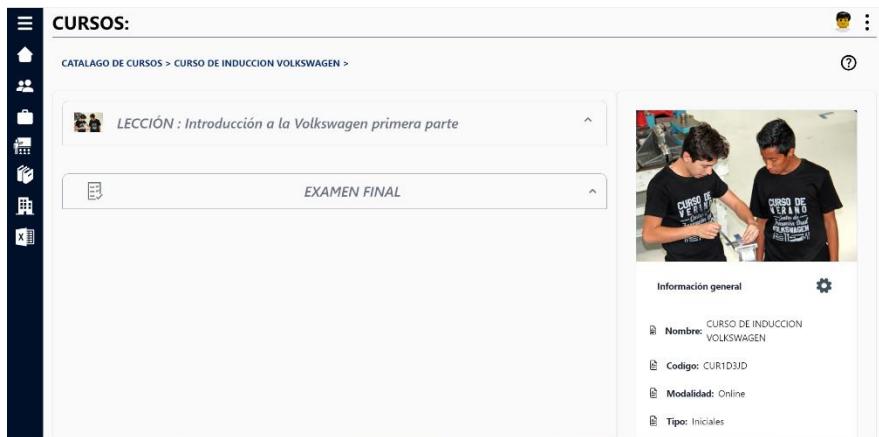


Ilustración 29 interfaz cursos internos por curso.

- Funcionabilidad:

En esta interfaz el usuario administrador tendrá un botón donde asignará las lecciones correspondientes al curso creado.

Una vez asignada las lecciones el administrador tendrá un botón donde podrá agregar los contenidos a cada lección y de igual manera su examen correspondiente a cada respectivo contenido.

Una vez terminado los procesos anteriores al administrador le aparecerá un botón donde podrá agregar a los usuarios al curso.

- Programación:

La interfaz del Vista curso interno está programado a través de un controlador (CursosInternos/CursoController) de la siguiente manera:

- Función show():

```
public function show(Request $request, string $id)
{
    $curso = Curso::find($id);
    $categoria = Categoria::all();
    $modalidad = ModalidadCurso::all();
    $tipo = TipoCurso::all();
    $usuarios = User::all();
    return view('CursosInternos.cursos.configurarCursos', compact('curso', 'modalidad', 'tipo', 'usuarios', 'categoria'));
}
```

Ilustración 30 función show (CursosInternos/CursoController).

En esta función obtenemos los datos de cada curso por medio de un id que se obtiene al momento de ingresar a esta interfaz.

Una vez obtenida la información del curso lo retornamos a la vista para mostrar los datos.

Sprint 3: Diseño y programación de las interfaces administrador.

El tercer sprint se basó en la creación y programación y revisión de las siguientes interfaces:

- Vista creación curso interno
- Vista creación lección

Desarrollo de la interfaz de la creación curso interno:

- Diseño:

Ilustración 31 interfaz creación de cursos internos pagina 1.

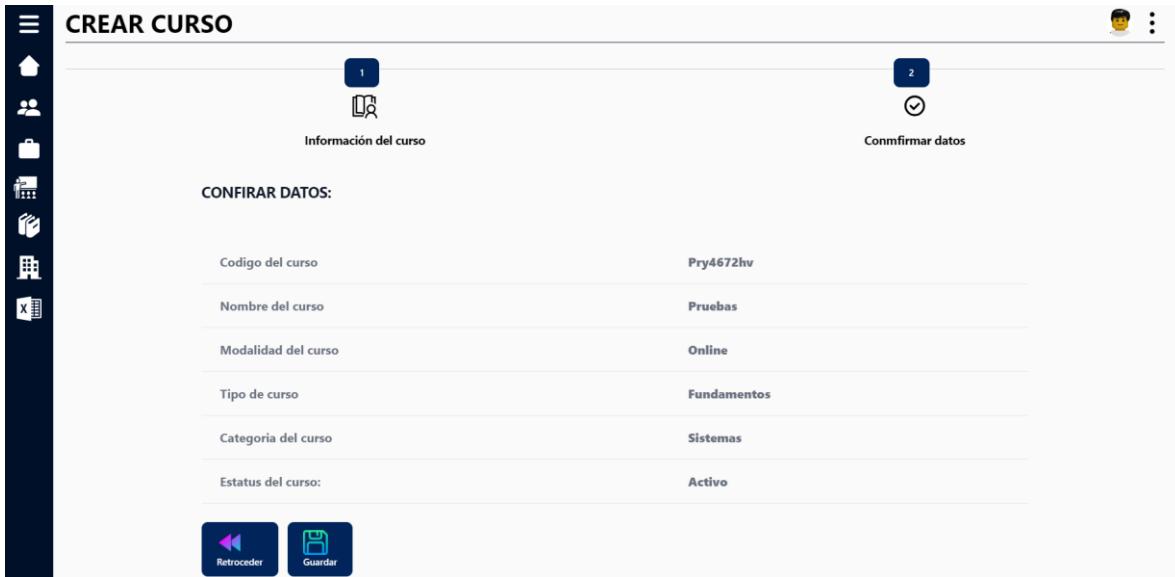


Ilustración 32 interfaz creación de cursos internos pagina 2.

- Funcionabilidad:

El administrador podrá dar de alta nuevos cursos internos donde insertará datos esenciales para el curso como son nombre, código, imagen del curso en la primera página, en la segunda página el administrador pondrá la fecha de inicio y la fecha de término, en la tercera página el administrador asignará modalidad, tipo, categoría. En la ultima pagina el administrador antes de guardar el curso tendrá que verificar la información ingresada en cada pagina si esta es correcta continuará con el botón guardar y si no podrá retroceder a cambiar el dato incorrecto.

- Programación:

La vista cursos internos está programado a través de un plugin de laravel llamado livewire (livewire/wizard) de la siguiente manera:

- Variables publicas:

```
use WithFileUploads;

public $currentStep = 1;
public $codigo, $nombre, $fecha_inicio, $fecha_termino, $curso_id,
       $estado = 1, $categoria_id, $modalidad_id, $tipo_curso_id,
       $interno_planta = 1, $imagen;
public $successMessage = '';
```

Ilustración 33 variables públicas(livewire/wizard).

- Función render():

```
public function render()
{
    $modalidad = ModalidadCurso::all();
    $tipo = TipoCurso::all();
    $categoria = Categoria::all();
    return view('livewire.wizard', compact('modalidad', 'tipo', 'categoria'));
}
```

Ilustración 34 función render (livewire/wizard).

En esta función se encarga de renderizar todos los datos que el administrador vaya guardando en cada página así mismo renderiza la información adicional que el administrador necesite.

- Función firstStepSubmit():

```
public function firstStepSubmit()
{
    $validatedData = $this->validate([
        'codigo' => 'required|string|min:5|unique:cursos',
        'nombre' => 'required|string|min:5',
        'imagen' => 'required|image|mimes:jpg,png',
    ]);

    $this->currentStep = 2;
}
```

Ilustración 35 función firstStepSubmit (livewire/wizard).

Esta función es la que controla la primera pagina de la creación del curso se encarga de validar la información ingresada por el usuario, que sean datos validos si es así la variable publica currentStep declarada al inicio su valor pasara a ser igual a 2.

- Función secondStepSubmit():

```
public function secondStepSubmit()
{
    $validatedData = $this->validate([
        'fecha_inicio' => 'required|date',
        'fecha_termino' => 'required|date',
    ]);

    $this->currentStep = 3;
}
```

Ilustración 36 función secondStepSubmit (livewire/wizard).

- Función threeStepSubmit():

```

public function threeStepSubmit()
{
    $validatedData = $this->validate([
        'modalidad_id' => 'required',
        'tipo_curso_id' => 'required',
        'categoria_id' => 'required',
    ]);

    $this->currentStep = 4;
}

```

Ilustración 37 función threeStepSubmit (livewire/wizard).

Estas dos funciones se encargan de almacenar la información faltante que el administrador ingrese en las páginas 2 y 3. De igual manera se encarga de validar cada uno de la información ingresada.

- Función messages():

```

public function messages()
{
    return [
        // VALIDACION DE CODIGO
        'codigo.required' => 'El campo codigo es obligatorio.',
        'codigo.string' => 'El campo codigo debe ser una cadena.',
        'codigo.min' => 'El campo codigo debe ser mínimo 5 caracteres.',
        'codigo.unique' => 'El campo codigo es unico.',
        // VALIDACION DE NOMBRE
        'nombre.required' => 'El campo nombre es obligatorio.',
        'nombre.string' => 'El campo nombre debe ser una cadena.',
        'nombre.min' => 'El campo nombre debe ser mínimo 5 caracteres.',
        // VALIDACION DE IMAGEN
        'imagen.required' => 'El campo imagen es obligatorio.',
        'imagen.image' => 'El campo imagen debe ser formato png,jpg.',
        'imagen.mimes' => 'El campo imagen debe ser formato png,jpg.',
        // VALIDACION DE FECHA INICIO
        'fecha_inicio.required' => 'El campo fecha inicio es obligatorio.',
        'fecha_inicio.date' => 'El campo fecha inicio debe ser una fecha válida.',
        // VALIDACION DE FECHA TERMINO
        'fecha_termino.required' => 'El campo fecha termino es obligatorio.',
        'fecha_termino.date' => 'El campo fecha termino debe ser una fecha válida.',
        // VALIDACION DE MODALIDAD DEL CURSO
        'modalidad_id.required' => 'El campo Modalidad termino es obligatorio.',
        // VALIDACION DE TIPO CURSO
        'tipo_curso_id.required' => 'El campo Tipo termino es obligatorio.',
        // VALIDACION DE CATEGORIA CURSO
        'categoria_id.required' => 'El campo Categoria termino es obligatorio.',

    ];
}

```

Ilustración 38 función messages (livewire/wizard).

En esta función se valida los tipos de campos que el usuario pueda ingresar y así mismo regresar un mensaje de error diciendo que tipos de datos admite.

- Función submitform():

```
public function submitForm()
{
    DB::transaction(function () {
        $imgCurso = $this->imagen->store('public/imagenes');
        $url = Storage::url($imgCurso);

        $curso = Curso::create([
            'codigo' => $this->codigo,
            'nombre' => $this->nombre,
            'fecha_inicio' => $this->fecha_inicio,
            'fecha_termino' => $this->fecha_termino,
            'modalidad_id' => $this->modalidad_id,
            'tipo_curso_id' => $this->tipo_curso_id,
            'interno_planta' => $this->interno_planta,
            'imagen' => $url,
            'estado' => $this->estado,
        ]);

        $this->curs0_id = $curso->id_curso;

        Categoria_curso::create([
            'categoria_id' => $this->categoria_id,
            'curso_id' => $curso->id_curso
        ]);
    });
    $this->clearForm();

    return to_route("curs.show", $this->curs0_id)->with('agregado', 'Agregado Correctamente');
}
```

Ilustración 39 función submitform (livewire/wizard).

Esta función es la encargada de guardar los datos en la base de datos de una manera exitosa. Para este caso utilizamos una propiedad de laravel llamada “BD” que nos permite verificar si existe un error ala hora de guardar la información, ya que estamos guardando datos en dos tablas distintas al mismo tiempo. Lo cual con la ayuda de esta propiedad nos permite a detener la operación si existiera un error, mas explicado se refiere a que si al momento de guardar la información en la primera o en la segunda tabla tabla existiera un error la operación se cancelaria y no se guardaría nada en ninguna de las dos tablas.

- función clearForm() y Función back():

```

public function back($step)
{
    $this->currentStep = $step;
}

public function clearForm()
{
    $this->codigo = '';
    $this->nombre = '';
    $this->fecha_inicio = '';
    $this->fecha_termino = '';
    $this->modalidad_id = '';
    $this->tipo_curso_id = '';
    // $this->interno_planta = '';
    $this->estado = '';
}

```

Ilustración 40 función back y clearForm (livewire/wizard).

En la función back nos sirve para poder avanzar y retroceder en las páginas de la creación del curso y la función clearForm se encarga de reiniciar las variables una vez que el administrador allá guardado la información antes guardada anteriormente.

Desarrollo de la interfaz de la creación lecciones:

- Diseño:

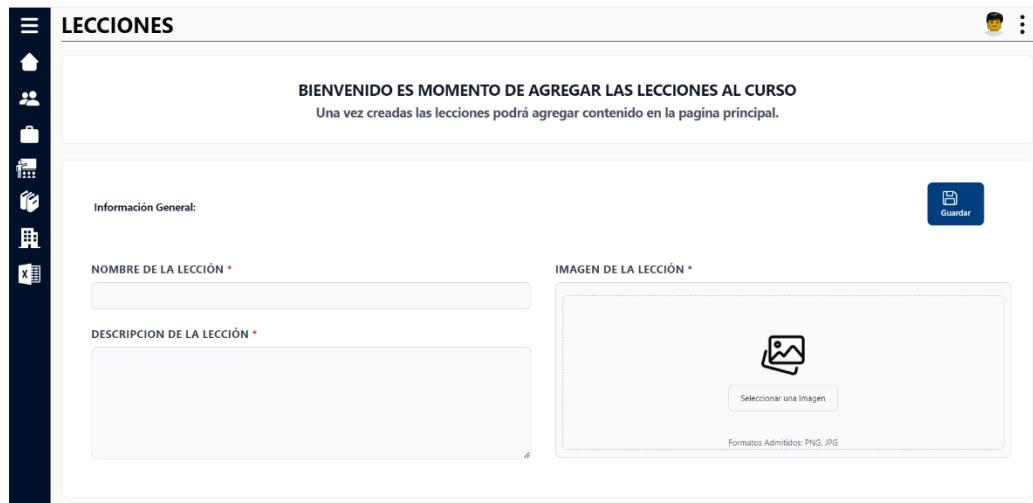


Ilustración 41 interfaz creación de lecciones.

- Funcionabilidad:

En esta interfaz el usuario administrador podrá agregar lecciones al curso donde podrá agregar un nombre, una descripción, y una imagen referente a la lección.

- Programación:

La interfaz del Vista crear lección está programado a través de un controlador (CursosInternos/LeccionController) de la siguiente manera:

- Función show():

```
public function show(string $id)
{
    return view('cursosinternos.lecciones.agregar', compact('id'));
}
```

Ilustración 42 función show (CursosInternos/LeccionController).

En esta función obtenemos el id de la lección para posteriormente retornarla a la vista de agregar una lección.

- Función store():

```
public function store(SaveLeccionRequest $request)
{
    $leccione = new Leccion();
    $leccione->nombre = $request->post('nombre');
    $leccione->descripcion = $request->post('descripcion');
    $leccione->curso_id = $request->post('curso_id');
    $lecciones = $request->file('url_imagen')->store('public/leccion');
    $url = Storage::url($lecciones);
    $leccione->url_imagen = $url;
    $leccione->saveOrFail();
    $leccione = $leccione->curso_id;
    $curso = $leccione->curso_id;
    return to_route("curs.show", $curso)->with('agregado', 'Lección Agregado Correctamente');
}
```

Ilustración 43 función store (CursosInternos/LeccionController).

En esta función store es la encargada de guardar los datos obtenidos a través de la variable request para obtener los datos y con la ayuda del modelo de lección podremos guardar los datos correctamente asignando cada dato obtenido en los inputs y con el método saveOrFail guardarlos en la base de datos. Una vez llevado a cabo este proceso retornará a la vista curso interno con un mensaje de lección agregada correctamente.

sprint 4: Diseño y programación de las interfaces administrador.

El cuarto sprint se basó en la creación y programación y revisión de las siguientes interfaces:

- Vista creación contenido
- Vista edición contenido
- Vista edición lección
- Vista edición información del curso interno

Desarrollo de la interfaz de la creación contenido:

- Diseño:

Ilustración 44 interfaz creación de contenido .

- Funcionabilidad:

En esta interfaz el usuario administrador podrá agregar contenidos a las lecciones donde podrá agregar un nombre, una descripción, y un archivo ya sea PDF, IMG, MP4, TXT.

- Programación:

La interfaz del Vista crear contenido está programado a través de un controlador (CursosInternos/ContenidoController) de la siguiente manera:

- Funcion show():

```
public function show(string $id)
{
    return view('cursosinternos.contenido.contenido', compact('id'));
}
```

Ilustración 45 función show (CursosInternos/ContenidoController).

En esta función obtenemos el id del contenido que mandamos a través de la ruta para poder agregar un contenido y posteriormente redireccionar a la vista donde el usuario rellenara el formulario de agregación de los datos del contenido.

- Función store():

```
public function store(SaveContenidoRequest $request)
{
    //Este apartado es para crear un contenido
    $contenido = new Contenido();
    $contenido->nombre = $request->post('nombre');
    $contenido->descripcion = $request->post('descripcion');
    $contenido->leccion_id = $request->post('leccion_id');
    $contenido->estado = 0;
    $contenido->saveOrFail();
    //Esta parte es para obtener el id del contenido
    $id_cont = $contenido->id_contenido;
    // crear en otra tabla
    $media = new Media_contenido();
    $contenidos = $request->file('url')->store('public/contenido');
    $url = Storage::url($contenidos);
    $media->url = $url;
    $media->contenido_id = $id_cont;
    $media->saveOrFail();
    $leccion = Leccion::find($request->leccion_id);
    $curso = $leccion->curso_id;

    return to_route("curs.show", $curso)->with('agregado', 'Contenido Agregado Correctamente');
}
```

Ilustración 46 función store (CursosInternos/ContenidoController).

En esta función store es en donde se guardará la información a la base de datos con la ayuda del modelo contenido y el modelo media_contenido ya que en este caso guardaremos información en dos tablas diferentes cada una con sus respectivos datos.

Desarrollo de la interfaz de la edición contenido:

- Diseño:

Ilustración 47 interfaz edición de contenido.

- Funcionabilidad:

En esta interfaz el usuario administrador podrá editar la información respecto al contenido seleccionado.

- Programación:

La interfaz del Vista edición contenido está programado a través de un controlador (CursosInternos/ContenidoController) de la siguiente manera:

- Funcion update():

```

public function update(Request $request, string $id)
{
    $contenido = Contenido::find($id);
    if(is_null($contenido)){
        return redirect()->back();
    }
    $contenido->nombre = $request->post('nombre');
    $contenido->descripcion = $request->post('descripcion');
    $contenido->leccion_id = $request->post('leccion_id');
    $contenido->saveOrFail();
    //Esta parte es para obtener el id del contenido
    $id_cont = $contenido->id_contenido;
    // crear en otra tabla
    $id_media = $contenido->media[0]->id_media;
    $media = Media_contenido::find($id_media);
    if(is_null($media)){
        return redirect()->back();
    }
    if ($request->hasFile('url')) {
        $medias = $request->file('url')->store('public/contenido');
        $url = Storage::url($medias);
        $media->url = $url;
    }
    $media->contenido_id = $id_cont;
    $media->saveOrFail();
    $leccion = Leccion::find($request->leccion_id);
    if(is_null($leccion)){
        return redirect()->back();
    }
    $curso = $leccion->curso_id;

    return to_route("curs.show", $curso)->with('actualizado', 'Contenido Actualizada Correctamente');
}

```

Ilustración 48 función update (CursosInternos/ContenidoController)

Primero, se busca el contenido en la base de datos utilizando el modelo Contenido y el \$id proporcionado. Si no se encuentra el contenido con ese ID, redirecciona hacia atrás (la página anterior).

Se actualizan los campos del contenido con los valores proporcionados en la solicitud (\$request) usando el método post (asumiendo que son campos enviados a través de una solicitud POST). Los campos que se actualizan son nombre, descripcion y leccion_id'.

Luego, el contenido se guarda en la base de datos utilizando el método saveOrFail() para asegurarse de que la operación se realice con éxito. Si hay algún error al guardar, se generará una excepción y se redirecciará hacia atrás (la página anterior).

A continuación, el código obtiene el ID del contenido actualizado para utilizarlo en las operaciones siguientes.

Luego, se busca el primer elemento de la relación "media" del contenido (\$contenido->media[0]). Es probable que exista una relación definida en el modelo Contenido para acceder a los medios asociados.

Si no se encuentra una "media" asociada, se redirecciona hacia atrás (la página anterior).almacena en la carpeta "public/contenido" utilizando el método store() de

Laravel. Luego, se obtiene la URL pública del archivo usando `Storage::url()` y se actualiza el campo `url` de la "media" con la nueva URL.

A continuación, se actualiza el campo `contenido_id` de la "media" con el ID del contenido actualizado.

Se guarda la "media" en la base de datos utilizando `saveOrFail()` para asegurarse de que la operación se realice correctamente.

Luego, se busca la lección relacionada con el contenido a través del ID proporcionado en la solicitud (`lección_id`).

Si no se encuentra la lección, se redirecciona hacia atrás (la página anterior)

Se obtiene el ID del curso asociado con la lección.

Finalmente, se redirecciona a la ruta "curs.show" con el ID del curso como parámetro y se muestra un mensaje de éxito en la sesión (with('actualizado', 'Contenido Actualizada Correctamente')).

Desarrollo de la interfaz de la edición lección:

- Diseño:

EDITAR LECCIÓN

BIENVENIDO ES MOMENTO DE EDITAR LA LECCION DEL CURSO
Una vez creadas las Lecciones podrá agregar contenido en la pagina principal

Información General:

NOMBRE DE LA LECCIÓN *

hsjvgjcgjh

DESCRIPCION DE LA LECCIÓN *

hxjkchsjhcgshjcj

IMAGEN DE LA LECCIÓN *

Seleccionar una Imagen

Formatos Admitidos: PNG, JPG

Guardar

Ilustración 49 interfaz edición de lección.

- Funcionabilidad:

En esta interfaz el usuario administrador podrá editar la información de la lección y actualizarla a la que dese.

- Programación:

La interfaz esta programada de la siguiente manera:

- Funcion update():

```

public function update(Request $request, string $id)
{
    $leccione = Leccion::find($id);
    if(is_null($leccione)){
        return redirect()->back();
    }
    if ($request->hasFile('url_imagen')) {
        $lecciones = $request->file('url_imagen')->store('public/leccion');
        $url = Storage::url($lecciones);
        $leccione->url_imagen = $url;
    }
    $leccione->nombre = $request->post('nombre');
    $leccione->descripcion = $request->post('descripcion');
    $leccione->curso_id = $request->post('curso_id');
    $leccione->saveOrFail();
    $curso = $leccione->curso_id;
    return to_route("curs.show", $curso)->with('agregado', 'Leccion Actualizada Correctamente');
}

```

Ilustración 50 función update() (cursosinternos/LeccionesController)

Se busca la lección en la base de datos utilizando el modelo **Leccion** y el \$id proporcionado. Si no se encuentra la lección con ese ID, redirecciona hacia atrás (la página anterior).

Si hay un archivo de imagen (**url_imagen**) en la solicitud (**\$request**), se almacena en la carpeta "public/leccion" utilizando el método **store()** de Laravel. Luego, se obtiene la URL pública de la imagen usando **Storage::url()** y se actualiza el campo **url_imagen** de la lección con la nueva URL.

A continuación, se actualizan los campos de la lección con los valores proporcionados en la solicitud (**\$request**) utilizando el método **post** (asumiendo que son campos enviados a través de una solicitud **POST**). Los campos que se actualizan son nombre, descripcion y curso_id'.

Luego, la lección se guarda en la base de datos utilizando el método **saveOrFail()** para asegurarse de que la operación se realice con éxito. Si hay algún error al guardar, se generará una excepción y se redireccionará hacia atrás (la página anterior).

Se obtiene el ID del curso asociado con la lección actualizada.

Finalmente, se redirecciona a la ruta "**curs.show**" con el ID del curso como parámetro y se muestra un mensaje de éxito en la sesión (**with('agregado', 'Leccion Actualizada Correctamente')**)

Desarrollo de la interfaz de la edición contenido:

- Diseño:

The screenshot displays the 'EDITAR CURSO' (Edit Course) interface. On the left, there is a vertical sidebar with several dropdown menus for course modalities: 'Modalidad del Curso' (avanzado), 'Modalidad del Curso' (Administrativo), 'Modalidad del Curso' (online), and 'Modalidad del Curso' (avanzado). The main content area contains fields for course information: 'Nombre del Curso' (PRUEBA FINAL ADMINISTRADOR), 'Codigo del Curso' (y7868gbihgik), 'Fecha de Inicio:' (05/07/2023), 'Fecha de Término:' (19/07/2023), 'Estatus del Curso:' (Curso Activo selected), and 'Imagen del Curso' (a placeholder image of a blue circle with a white 'W'). Below the image is a file selection button ('Seleccionar archivo') and a message ('Ninguno archivo selec.'). At the bottom right is a large blue 'Guardar' (Save) button with a document icon.

Ilustración 51 interfaz edición de curso.

- Funcionabilidad

En esta interfaz el usuario podrá modificar la información general del curso.

Programación:

Esta interfaz está programada de la siguiente manera:

- Función update():

```

public function update(Request $request, string $id)
{
    $curso = Curso::find($id);
    if (is_null($curso)) {
        return redirect()->back();
    }
    if ($request->hasFile('imagen')) {
        $img = $request->file('imagen')->store('public/imagenes');
        $url = Storage::url($img);
        $curso->imagen = $url;
    }
    $curso->codigo = $request->post('codigo');
    $curso->nombre = $request->post('nombre');
    $curso->estado = $request->post('estado');
    $curso->modalidad_id = $request->post('modalidad_id');
    $curso->tipo_curso_id = $request->post('tipo_curso_id');
    $categorias = $curso->categoria()->detach($curso->categoria->pluck('id_categoria'));
    $categorias = $curso->categoria()->attach($request->post('categoria_id'));
    $curso->saveOrFail();
    return redirect()->back()->with('actualizado', 'Actualizado Correctamente');
}

```

Ilustración 52 funcion update() (cursosinternos/CursoController)

Se busca el curso en la base de datos utilizando el modelo Curso y el **\$id proporcionado**. Si no se encuentra el curso con ese ID, redirecciona hacia atrás (la página anterior).

Si hay un archivo de imagen (**imagen**) en la solicitud (**\$request**), se almacena en la carpeta "public/imagenes" utilizando el método **store()** de Laravel. Luego, se obtiene la URL pública de la imagen usando **Storage::url()** y se actualiza el campo **imagen** del curso con la nueva URL.

A continuación, se actualizan los campos del curso con los valores proporcionados en la solicitud (**\$request**) utilizando el método **post** (asumiendo que son campos enviados a través de una solicitud POST). Los campos que se actualizan son código, nombre, estado, **modalidad_id** y **tipo_curso_id**.

Se maneja la relación "categoría" del curso. Primero, se eliminan todas las relaciones existentes utilizando el método **detach()** con el resultado de **pluck('id_categoria')** en la relación **categoria**. Luego, se crean nuevas relaciones utilizando el método **attach()** con el ID de la nueva categoría proporcionada en la solicitud (**\$request->post('categoria_id')**).

Luego, el curso se guarda en la base de datos utilizando el método **saveOrFail()** para asegurarse de que la operación se realice con éxito. Si hay algún error al guardar, se generará una excepción y se redireccionará hacia atrás (la página anterior).

Finalmente, se redirecciona hacia atrás (la página anterior) y se muestra un mensaje de éxito en la sesión (**with('actualizado', 'Actualizado Correctamente')**).

sprint 5: Diseño y programación de las interfaces administrador.

El quinto sprint se basó en la creación y programación y revisión de las siguientes interfaces:

- Vista creación examen
- Vista simulación de contenido Admin
- Vista visualización simulación examen Admin
- Vista visualización simulación calificación Admin
- Vista edición examen

Desarrollo de la interfaz de creación del examen para cada contenido y examen final:

- Diseño:

Ilustración 53 interfaz creación del examen.

- Funcionabilidad:

En esta interfaz el usuario administrador podrá asignar los exámenes correspondientes de cada contenido y de igual manera es la misma funcionalidad que para agregar el examen final.

- Programación:

Esta interfaz está programada de la siguiente manera tanto para el examen de cada contenido como para el examen final la única diferencia es la forma en la que regresa a la página principal:

- Función store():

```

public function store(Request $request)
{
    $dataExamen = [
        "nombre" => $request->nombre,
        "duracion" => $request->duracion,
        "contenido_id" => $request->contenido_id,
    ];

    $examen = Examen::create($dataExamen);

    foreach ($request->arreglo as $preguntas) {

        $dataPregunta = [
            "pregunta" => $preguntas[0],
            "examen_id" => $examen->id_examen
        ];

        $pregunta = Pregunta::create($dataPregunta);

        $dataOpciones = [];
        $opcionCorrecta = $preguntas['respuesta'];
        for ($i = 0; $i < count($preguntas['opciones']); $i++) {
            if ($opcionCorrecta == $i) {
                $consulta = [
                    'opcion' => $preguntas['opciones'][$i],
                    "pregunta_id" => $pregunta->id_pregunta,
                    "correcta" => true,
                ];
                array_push($dataOpciones, $consulta);
                continue;
            }

            $consulta = [
                'opcion' => $preguntas['opciones'][$i],
                "pregunta_id" => $pregunta->id_pregunta,
                "correcta" => false,
            ];
            array_push($dataOpciones, $consulta);
        }
        DB::table("opciones")->insert($dataOpciones);
    }

    $conT = Contenido::find($request->contenido_id);
    $LecC = $conT->leccion_id;
    $LecCI = Leccion::find($LecC);
    $idCuS = $LecCI->curso_id;
    $CuRsO = Curso::find($idCuS);
    // return "HOLIS SI GUARDO";
    return to_route("curs.show", $CuRsO)->with('agregado', 'Examen Agregado Correctamente');
}

```

Ilustración 54 función store() (cursosinternos/ExamenController)

- Función ExamenFinal():

```

public function ExamenFinal(Request $request)
{
    $dataExamen = [
        "nombre" => $request->nombre,
        "duracion" => $request->duracion,
        "curso_id" => $request->curso_id,
    ];

    $examen = Examen::create($dataExamen);

    foreach ($request->arreglo as $preguntas) {

        $dataPregunta = [
            "pregunta" => $preguntas[0],
            "examen_id" => $examen->id_examen
        ];

        $pregunta = Pregunta::create($dataPregunta);

        $dataOpciones = [];
        $opcionCorrecta = $preguntas['respuesta'];
        for ($i = 0; $i < count($preguntas['opciones']); $i++) {
            if ($opcionCorrecta == $i) {
                $consulta = [
                    'opcion' => $preguntas['opciones'][$i],
                    "pregunta_id" => $pregunta->id_pregunta,
                    "correcta" => true,
                ];
                array_push($dataOpciones, $consulta);
                continue;
            }

            $consulta = [
                'opcion' => $preguntas['opciones'][$i],
                "pregunta_id" => $pregunta->id_pregunta,
                "correcta" => false,
            ];
            array_push($dataOpciones, $consulta);
        }
        DB::table("opciones")->insert($dataOpciones);
    }

    $CuRs0 = Curso::find($request->curso_id);
    // return "HOLIS SI GUARDO";
    return to_route("curs.show", $CuRs0)->with('agregado', 'Examen Agregado Correctamente');
}

```

Ilustración 55 función ExamenFinal() (cursosinternos/ExamenController)

Se crea un arreglo **\$dataExamen** con la información del examen proporcionada en la solicitud (**\$request**). El arreglo contiene las claves "**nombre**", "**duracion**" y "**contenido_id**" con los respectivos valores extraídos de la solicitud.

Se crea un nuevo examen en la base de datos utilizando el modelo Examen y el método **create()** con el arreglo **\$dataExamen**. La variable **\$examen** almacenará el nuevo objeto de examen recién creado.

Se recorre el arreglo **\$request->arreglo**, que contiene información sobre las preguntas del examen.

Por cada pregunta, se crea un nuevo objeto de pregunta en la base de datos utilizando el modelo Pregunta y el método **create()**. Se crea un arreglo **\$dataPregunta** con la información de la pregunta, como el texto de la pregunta y el ID del examen al que pertenece. El texto de la pregunta se obtiene del índice 0 del arreglo **\$preguntas**.

Se crea un conjunto de opciones para cada pregunta y se almacenan en la base de datos utilizando el modelo **Opcion** y el método **insert()**. Las opciones se obtienen del arreglo **\$preguntas['opciones']** y se asocian a la pregunta recién creada. Además, se establece una opción como correcta (indicado por el valor true) según el índice proporcionado en **\$preguntas['respuesta']**.

En el caso de que la opción coincida con el índice de la respuesta correcta, se crea un arreglo **\$consulta** con los datos de la opción, y se agrega al arreglo **\$dataOpciones** utilizando **array_push()**. Si la opción no es correcta, se realiza el mismo proceso, pero se establece "**correcta** => **false**".

Una vez que se han creado y almacenado todas las opciones de la pregunta, se pasa al siguiente elemento del arreglo **\$request->arreglo** y se repite el proceso para crear las preguntas y sus opciones correspondientes.

Desarrollo de la interfaz de Edición del examen y examen final:

- Diseño:

The screenshot displays the 'INICIO' (Home) screen of an exam editing application. On the left, a vertical sidebar features icons for navigation: Home, User, Content, Exams, and Settings. The main area is titled 'INICIO' and contains the following fields:

- Información General del Contenido**
- Título del examen:** *
Examen prueba
- Duración del examen:** *
30
- Preguntas:**
- Pregunta:** *
ESTA ES UNA PRUEBA DE LA FUNCIONALIDAD DEL EXAMÉN?
- Opciones:** *
Respuesta 1 OPCIÓN 1
 OPCIÓN 3 OPCIÓN 4
- Respuesta correcta:** *
Respuesta 1

At the bottom are two blue buttons: 'Agregar Pregunta' (Add Question) and 'Editar examen' (Edit Exam). The top right corner includes a user icon and a vertical ellipsis.

Ilustración 56 interfaz edición del examen.

- Funcionalidad:

En esta interfaz el usuario podrá cambiar o modificar la información del examen de cada contenido o del examen final del curso.

- Programación:

Estas interfaces están programadas de la siguiente manera a través de su controlador:

- Funcion update():

```

public function update(Request $request, string $id)
{
    // dd($request);
    $examen = Examen::find($request->id_examen);
    if(is_null($examen)){
        return redirect()->back();
    }

    // Validar los datos enviados desde el formulario de edición
    $validatedData = $request->validate([
        'nombre' => 'required',
        'duración' => 'required',
        'contenido_id' => 'required'
        // Agrega las reglas de validación para los demás campos del examen
    ]);

    // Actualizar los datos del examen con los valores proporcionados en el formulario
    $examen->update($validatedData);

    foreach ($request->preguntas as $preguntaId => $preguntaData) {
        $pregunta = Pregunta::find($preguntaId);
        if(is_null($pregunta)){
            return redirect()->back();
        }
        $nombrePregunta = $preguntaData['titulo'];
        $pregunta->pregunta = $nombrePregunta;
        $pregunta->saveOrFail();

        $opcionCorrecta = $preguntaData['respuesta'];
        // dd($opcionCorrecta);

        foreach ($preguntaData['opciones'] as $opcionId => $opcionData) {
            // dd($opcionCorrecta);
            if ($opcionCorrecta == $opcionId) {

                $opcion = Opcion::find($opcionId);
                if(is_null($opcion)){
                    return redirect()->back();
                }
                $nombreOpcion = $opcionData['titulo'];
                // dd($nombreOpcion);
                $opcion->opcion = $nombreOpcion;
                $opcion->correcta = true;
                $opcion->saveOrFail();
                continue;
            }
            $opcion = Opcion::find($opcionId);
            if(is_null($opcion)){
                return redirect()->back();
            }
            $nombreOpcion = $opcionData['titulo'];
            $opcion->opcion = $nombreOpcion;
            $opcion->correcta = false;
            $opcion->saveOrFail();
        }

        if($request->arreglo){
            // ESTA PARTE ES PARA GUARDAR LAS NUEVAS PREGUNTAS
            foreach ($request->arreglo as $preguntas) {
                $dataPregunta = [
                    "pregunta" => $preguntas[0],
                    "examen_id" => $examen->id_examen
                ];

                $pregunta = Pregunta::create($dataPregunta);
                $dataOpciones = [];
                $opcionCorrecta = $preguntas['respuesta'];
                for ($i = 0; $i < count($preguntas['opciones']); $i++) {
                    if ($opcionCorrecta == $i) {
                        $consulta = [
                            'opcion' => $preguntas['opciones'][$i],
                            'pregunta_id' => $pregunta->id_pregunta,
                            'correcta' => true,
                        ];
                        array_push($dataOpciones, $consulta);
                    }
                }
                $consulta = [
                    'opcion' => $preguntas['opciones'][$i],
                    'pregunta_id' => $pregunta->id_pregunta,
                    'correcta' => false,
                ];
                array_push($dataOpciones, $consulta);
            }
            DB::table("opciones")->insert($dataOpciones);
        }
    }

    // return "holis";
    $cont = Contenido::find($request->contenido_id);
    $lecc = $cont->lección_id;
    $leccCI = Lección::find($lecc);
    $idCuS = $leccCI->curso_id;
    $cuSO = Curso::find($idCuS);
    // return "HOLIS SI GUARDO";
    return to_route("curs.show", $cuSO)->with('actualizado', 'Examen actualizado Correctamente');
}

```

Ilustración 57 función update().

- Funcion actualizar():

```

public function actualizar(Request $request, string $id)
{
    // dd($request);
    $examen = Examen::find($request->id_examen);
    if(is_null($examen)){
        return redirect()->back();
    }
    // Validar los datos enviados desde el formulario de edición
    $validatedData = $request->validate([
        'nombre' => 'required',
        'duracion' => 'required',
        'curso_id' => 'required'
        // Agrega las reglas de validación para los demás campos del examen
    ]);

    // Actualizar los datos del examen con los valores proporcionados en el formulario
    $examen->update($validatedData);

    foreach ($request->preguntas as $preguntaId => $preguntaData) {
        $pregunta = Pregunta::find($preguntaId);
        if(is_null($pregunta)){
            return redirect()->back();
        }
        $nombrePregunta = $preguntaData['titulo'];
        $pregunta->pregunta = $nombrePregunta;
        $pregunta->saveOrFail();

        $opcionCorrecta = $preguntaData['respuesta'];
        // dd($opcionCorrecta);

        foreach ($preguntaData['opciones'] as $opcionId => $opcionData) {
            // dd($opcionCorrecta);
            if ($opcionCorrecta == $opcionId) {

                $opcion = Opcion::find($opcionId);
                if(is_null($opcion)){
                    return redirect()->back();
                }
                $nombreOpcion = $opcionData['titulo'];
                // dd($nombreOpcion);
                $opcion->opcion = $nombreOpcion;
                $opcion->correcta = true;
                $opcion->saveOrFail();
                continue;
            }
            $opcion = Opcion::find($opcionId);
            if(is_null($opcion)){
                return redirect()->back();
            }
            $nombreOpcion = $opcionData['titulo'];
            $opcion->opcion = $nombreOpcion;
            $opcion->correcta = false;
            $opcion->saveOrFail();
        }
    }
    if($request->arreglo){
        // ESTA PARTE ES PARA GUARDAR LAS NUEVAS PREGUNTAS
        foreach ($request->arreglo as $preguntas) {
        }
    }

    // return "holis";
    // $com = Contenido::find($request->contenido_id);
    // $lecC = $com->leccion_id;
    // $lecCI = Leccion::find($lecC);
    // $idCuS = $lecCI->curso_id;
    // $CURSO = Curso::find($request->curso_id);
    // return "HOLIS SI GUARDO";
    return to_route("curs.show", $CURSO)->with('actualizado', 'Examen actualizado Correctamente');
}

```

Ilustración 58 función actualizar().

se intenta encontrar un registro específico en la tabla **examenes** (asumiendo que hay un modelo Examen) utilizando el **id_examen** que viene en la solicitud (**\$request**). Si el examen no se encuentra (es nulo), la función redirige al usuario a la página anterior (**redirect()->back()**). Esto puede suceder si el examen con el **id_examen** proporcionado no existe en la base de datos.

A continuación, se valida la información enviada en el formulario utilizando la función **validate()**. Se asegura de que los campos nombre, **duracion**, y **contenido_id** estén presentes en la solicitud (formulario). Si la validación falla, la función arrojará automáticamente errores de validación y redirigirá al usuario a la página anterior.

Si la validación es exitosa, se procede a actualizar los datos del examen (**\$examen**) con los valores proporcionados en el formulario (**\$validatedData**). Esto asume que el modelo Examen tiene un método **update()** que realiza la actualización.

A continuación, el código entra en un bucle **foreach** que recorre el arreglo **\$request->preguntas**, que contiene información sobre las preguntas enviadas en el formulario de edición. Por cada pregunta, se busca el registro correspondiente en la tabla preguntas (asumiendo que hay un modelo **Pregunta**). Si no se encuentra la pregunta, se redirige al usuario a la página anterior.

Se actualiza el título de la pregunta (**\$pregunta->pregunta**) con el valor proporcionado en el formulario (**\$preguntaData['titulo']**), y luego se guarda en la base de datos.

Luego, se itera sobre las opciones de la pregunta en el arreglo

\$preguntaData['opciones']. Se verifica si la opción es la correcta comparándola con **\$opcionCorrect** (que se obtiene de **\$preguntaData['respuesta']**). Si la opción es la correcta, se actualiza el título de la opción y se marca como correcta en la base de datos. Después de actualizar las preguntas existentes, se verifica si hay nuevas preguntas en el arreglo **\$request->arreglo** (esto asume que en el formulario de edición hay un campo con el nombre "arreglo" que contiene información sobre las nuevas preguntas).

Si hay nuevas preguntas, se itera sobre ellas y se crea un nuevo registro en la tabla **preguntas** (asumiendo que hay un modelo **Pregunta**) con el título y el **id_examenes proporcionados**.

A continuación, se crea un conjunto de opciones para cada nueva pregunta. Se itera sobre las opciones del arreglo **\$preguntas['opciones']**, y para la opción correcta, se marca como correcta antes de guardarla en la base de datos.

Finalmente, después de actualizar las preguntas y opciones existentes y agregar las nuevas preguntas y opciones, se redirige al usuario a la página de detalles del curso correspondiente.

Desarrollo de la interfaz de simulación administrador contenido:

- Diseño:

The screenshot shows a web-based simulation interface for a Cisco course. On the left, there's a vertical sidebar with icons for navigation. The main area has a header "CURSOS:" and a sub-header "PRUEBA FINAL ADMINISTRADOR > lección 1 prueba > contenido 1 prueba". Below this, a section titled "Contenido 1 prueba:" displays a message from Chuck Robbins, Chief Executive Officer of Cisco. The message reads:

preparado para encontrar magníficas oportunidades profesionales.
Ha obtenido la credencial de nivel estudiante por completar CCNA v7: Introducción a Redes y adquirió las siguientes capacidades:

- Configurar los routers, los switches y los dispositivos finales para proporcionar acceso a recursos de red locales y remotos y para habilitar la conectividad integral entre dispositivos remotos.
- Explicar cómo los protocolos físicos y de capa de enlace de datos admiten el funcionamiento de Ethernet en una red conmutada
- Configurar los routers para habilitar la conectividad de extremo a extremo entre dispositivos remotos.
- Crear esquemas de direccionamiento IPv4 e IPv6 y verificar la conectividad de red entre dispositivos.
- Explicar cómo las capas superiores del modelo OSI admiten aplicaciones de red.
- Configurar una red pequeña con prácticas recomendadas de seguridad.
- Solucionar problemas de conectividad en una red pequeña.

En el mundo actual, la capacitación técnica es más importante que nunca, y Cisco se enorgullece de proporcionarle con los conocimientos y aptitudes necesarios para construir y mantener redes digitales. Continúe con este excelente trabajo. Le deseamos lo mejor y esperamos que disfrute de un éxito continuo en el futuro.

Cordialmente,

Chuck Robbins

LECCIÓN: lección 1 prueba
CONTENIDOS DE LA LECCIÓN
contenido 1 prueba

Ilustración 59 interfaz simulación del contenido administrador.

The screenshot shows another view of the Cisco simulation interface. It features a vertical sidebar with icons. The main content area includes a signature from Chuck Robbins, Chief Executive Officer of Cisco, followed by a "DESCRIPCIÓN DEL CURSO:" section. This section contains placeholder text (Lorem ipsum) and a "Realizar Examén" button at the bottom.

Cordialmente,
Chuck Robbins
Chuck Robbins
Chief Executive Officer
Cisco
www.netacad.com

DESCRIPCIÓN DEL CURSO:

Lorem ipsum es el texto que se usa habitualmente en diseño gráfico en demostraciones de tipografías o de borradores de diseño para probar el diseño visual antes de insertar el texto final. Aunque no posee actualmente fuentes para justificar sus hipótesis, el profesor de filología clásica Richard Mc Clintock asegura que su uso se remonta a los impresores de comienzos del siglo xvi.¹ Su uso en algunos editores de texto muy conocidos en la actualidad ha dado al texto lorem ipsum nueva popularidad. El texto en sí no tiene sentido aparente, aunque no es aleatorio, sino que deriva de un texto de Cicerón en lengua latina, a cuyas palabras se les han eliminado silabas o letras. El significado del mismo no tiene importancia, ya que solo es una demostración o prueba. El texto procede de la obra De finibus bonorum et malorum (Sobre los límites del bien y del mal) que comienza con:

Realizar Examén

Ilustración 60 interfaz simulación del contenido administrador.

- Programación:

Esta interfaz está programada de la siguiente manera a través de su controlador:

- Función ver():

```
public function ver(string $id)
{
    $contenido = Contenido::find($id);
    if(is_null($contenido)){
        return redirect()->back();
    }
    $archivo = public_path($contenido->media[0]->url); // Ruta al archivo que deseas obtener la extensión
    $extension = File::extension($archivo);
    $lección = Lección::find($contenido->lección_id);
    return view('Cursosinternos.vistaPrevia.index', compact('contenido', 'lección', 'extension'));
}
```

Ilustración 61 función ver().

Se busca un registro específico en la tabla contenidos (asumiendo que hay un modelo **Contenido**) utilizando el id proporcionado en la variable **\$id**. Si no se encuentra ningún registro con el id proporcionado, la función redirige al usuario a la página anterior (**redirect()->back()**). Esto puede suceder si el contenido con el id especificado no existe en la base de datos.

Se obtiene la ruta completa al archivo multimedia asociado al contenido (**\$contenido->media[0]->url**). Esta línea asume que el contenido tiene una relación con la tabla de medios (media), y que el primer elemento ([0]) de esa relación contiene el archivo multimedia.

Se utiliza la clase File para obtener la extensión del archivo en la ruta proporcionada. La función **File::extension(\$archivo)** extraerá la extensión del archivo, que es la parte después del último punto en el nombre del archivo.

Se busca el registro de la lección asociada al contenido (**\$contenido->lección_id**). Esto asume que hay una relación entre los contenidos y las lecciones, y que el campo **lección_id** en el contenido almacena el ID de la lección asociada.

Finalmente, la función devuelve una vista llamada '**Cursosinternos.vistaPrevia.index**' y pasa como datos la variable **\$contenido**, **\$lección** y **\$extension**, que serán accesibles en la vista para mostrar la información correspondiente.

Desarrollo de la interfaz de simulación administrador examen:

- Diseño:

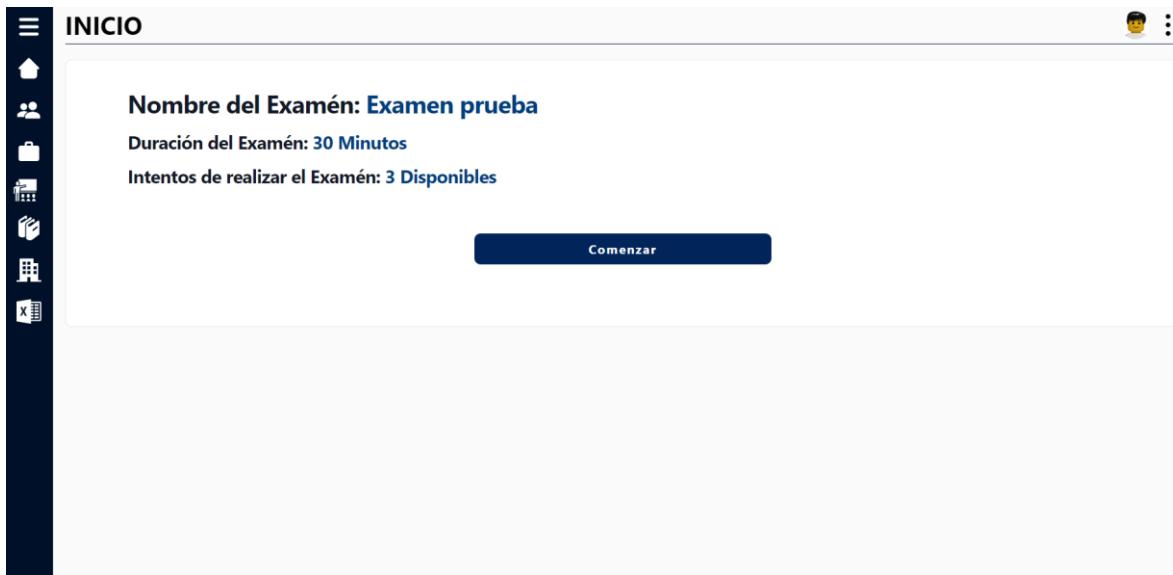


Ilustración 62 interfaz simulación del examen administrador.

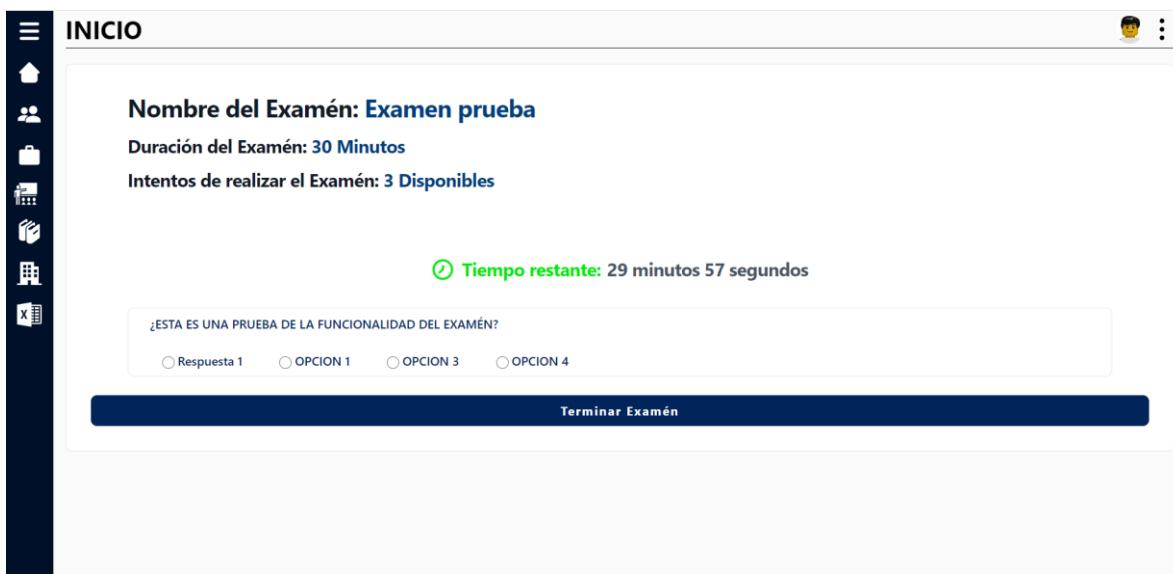


Ilustración 63 interfaz simulación del examen administrador.

- Programación:

Para esta interfaz se utilizaron dos funciones uno donde se muestra la información y otra para validar las respuestas correctas de cada pregunta del examen.

- Función verExam():

```

public function verExM(string $id)
{
    $intentos = 3;
    $contenido = Contenido::find($id);
    if(is_null($contenido)){
        return redirect()->back();
    }
    $examen = $contenido->examen;
    $totalPreguntas = $examen[0]->preguntas->count();
    return view('cursosinternos.examenes.vistaExamen', compact('examen', 'intentos', 'totalPreguntas'));
}

```

Ilustración 64 función verEx().

- Función validarExam() y función isAnswerCorrect():

```

public function validarExam(Request $request, string $id)
{
    $validatedData = $request->except('_token', 'total_pregunta');
    // Procesar las respuestas del test
    $totalQuestions = $request->total_pregunta;
    $correctAnswers = 0;

    foreach ($validatedData as $pregunta => $opcion) {
        if ($this->isAnswerCorrect($opcion)) {
            ++$correctAnswers;
        }
    }
    // Calcular el promedio
    $promedio = ($correctAnswers / $totalQuestions) * 100;

    return view('cursosinternos.examenes.calificacion', compact('promedio'));
}

private function isAnswerCorrect($par)
{
    $opcion = Opcion::find($par);
    if ($opcion->correcta == 1) {
        return true;
    } else {
        return false;
    }
    // return $opcion[0]->correcta === $correcta;
}

```

Ilustración 65 función validarExam() y función isAnswerCorrect().

- Función verEx():

Se establece una variable \$intentos con un valor de 3. Esta variable representa el número máximo de intentos que un usuario puede tener para completar el examen.

Se busca un registro específico en la tabla contenidos (asumiendo que hay un modelo Contenido) utilizando el id proporcionado en la variable \$id. Si no se encuentra ningún registro con el id proporcionado, la función redirige al usuario a la página anterior (redirect()->back()). Esto puede suceder si el contenido con el id especificado no existe en la base de datos.

Se accede al examen asociado con el contenido mediante la relación examen en el modelo Contenido. Asumiendo que existe una relación entre los contenidos y los exámenes y que el examen asociado se obtiene a través de esta relación.

Se obtiene el número total de preguntas en el examen. Suponiendo que la relación entre el examen y las preguntas se denomina preguntas, \$examen[0]->preguntas->count() cuenta cuántas preguntas hay en el examen.

Finalmente, la función devuelve una vista llamada

'Cursosinternos.examenes.vistaExamen' y pasa como datos las variables \$examen, \$intentos, y \$totalPreguntas. Estas variables estarán disponibles en la vista para mostrar la información correspondiente.

- Función validarExam():

Se utiliza la función **except()** del objeto **\$request** para obtener todos los datos enviados en la solicitud, excepto el token **CSRF (_token)** y un campo llamado **total_pregunta**. La función **except()** devuelve un arreglo con los datos de la solicitud, excluyendo los campos mencionados.

Se obtiene el número total de preguntas en el examen a partir del campo **total_pregunta** enviada en la solicitud.

Se inicializa una variable **\$correctAnswers** con un valor de 0. Esta variable se utilizará para contar el número de respuestas correctas proporcionadas por el usuario.

Se itera sobre el arreglo **\$validatedData**, que contiene las respuestas enviadas por el usuario. Cada clave del arreglo representa el nombre de la pregunta y el valor representa la opción seleccionada por el usuario para esa pregunta.

Para cada respuesta proporcionada por el usuario, se llama a la función **isAnswerCorrect(\$opcion)** para verificar si la opción seleccionada es correcta o no.

Si la respuesta es correcta, se incrementa el contador **\$correctAnswers** en 1.

Después de contar el número total de respuestas correctas, se calcula el promedio dividiendo **\$correctAnswers** entre **\$totalQuestions** y multiplicándolo por 100.

Esto da como resultado el porcentaje de preguntas respondidas correctamente.

Finalmente, se carga una vista llamada '**cursosinternos.examenes.calificacion**' y se pasa como dato el valor de **\$promedio**. Esta vista mostrará la calificación obtenida por el usuario en el examen.

- Función isAnswerCorrect(\$opcion):

Se utiliza el modelo **Opcion** para buscar un registro específico en la tabla opciones (asumiendo que existe un modelo llamado **Opcion**). La función **find(\$par)** busca el registro en la tabla opciones cuyo id coincide con el valor proporcionado en la **variable \$par**.

Luego, se verifica si el valor del campo correcta en el registro de la opción es igual a 1. Si el **campo correcta** tiene el valor 1, significa que esta opción es la correcta para la pregunta correspondiente. Si es así, la función devuelve true, lo que indica que la respuesta es correcta.

Si el **campo correcta** no tiene el valor 1, la función devuelve false, lo que indica que la respuesta no es correcta.

Desarrollo de la interfaz de simulación administrador calificación:

- Diseño:

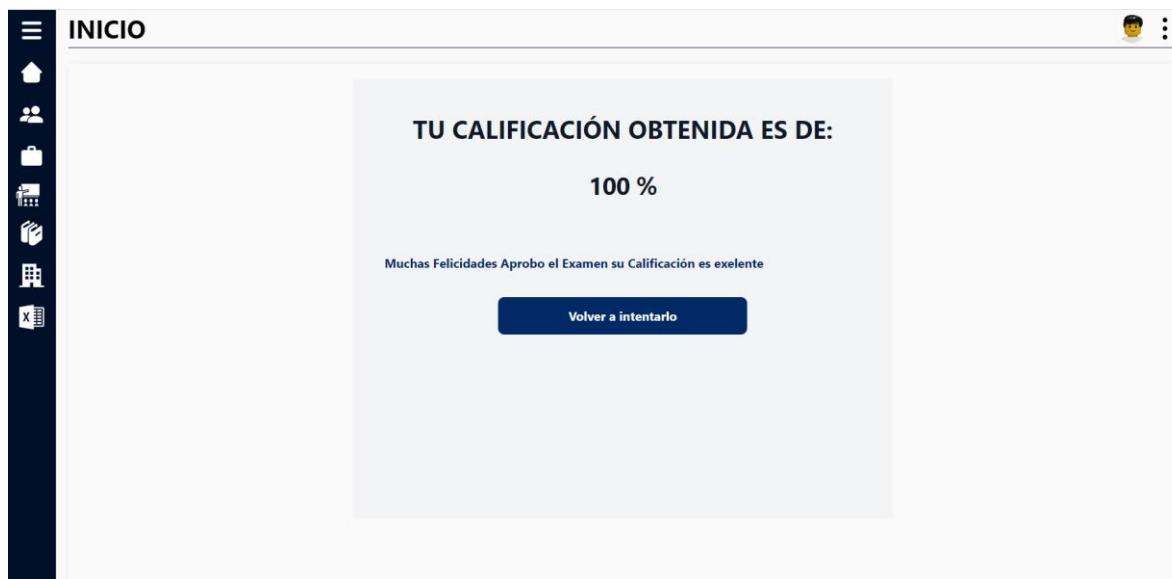


Ilustración 66 interfaz simulación de la calificación administrador.

- Programación:

En esta interfaz todo lo validamos en la vista calificaciones ya que todo lo obtenemos desde la función validarExam() que se encuentra explicada anteriormente.

sprint 6: Diseño y programación de las interfaces administrador.

El sexto sprint se basó en la creación y programación y revisión de las siguientes interfaces:

- Vista agregar usuario
- Vista eliminar usuario

Desarrollo de la interfaz de ver usuarios inscritos al curso:

- Diseño:

The screenshot shows a web-based application interface titled "Usuarios inscritos al curso". At the top right is a user icon with a plus sign. Below the title is a search bar with a magnifying glass icon and placeholder text "Buscar usuario por nombre...". The main area displays a table of registered users with the following columns: C. SGP, NOMBRE, FECHA DE TERMINÓ, and Opciones. There are five rows of data, each representing a user with their name, registration date, termination date (all listed as 27/07/2023), and an "Eliminar" button. At the bottom left, it says "Mostrando 1 a 5 de 5 resultados". At the bottom right, there are navigation buttons for page 1.

C. SGP	NOMBRE	FECHA DE TERMINÓ	Opciones
377855	Wilson Elouise Renner Gislason	Fecha terminó 27/07/2023	Eliminar
521614	Trevor Madilyn Wisozk	Fecha terminó 27/07/2023	Eliminar
99465557	Cora Stamm	Fecha terminó 27/07/2023	Eliminar
8	Mary Johnson Murray	Fecha terminó 27/07/2023	Eliminar
868197	Fae Hiram Jakubowski Cronin	Fecha terminó 27/07/2023	Eliminar

Ilustración 67 interfaz de vista usuarios.

- Programación:

Esta interfaz está programada desde la vista una vez haya usuarios inscritos al curso.

Desarrollo de la interfaz de agregar usuarios:

- Diseño:

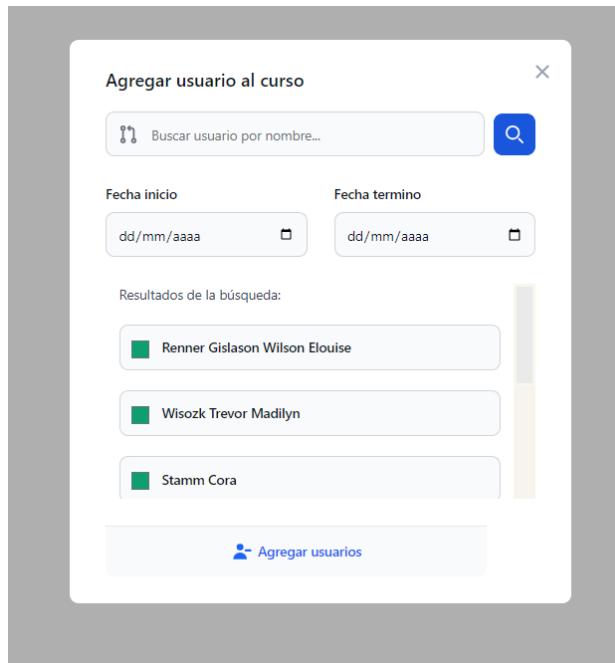


Ilustración 68 interfaz de vista agregar usuarios.

- Programación:

Esta interfaz esta programada de la siguiente manera dentro del controlador (cursosinternos/CursoController) de la siguiente manera:

- Función store():

```
public function store(Request $request)
{
    $request->validate(['usuarios' => 'array|required']);
    $dataUsuarios = [];
    foreach ($request->usuarios as $usuario) {

        $consulta = [
            "usuario_id" => $usuario,
            "curso_id" => $request->curso_id,
            "fecha_inicio" => $request->fecha_inicio,
            "fecha_termino" => $request->fecha_termino
        ];
        array_push($dataUsuarios, $consulta);
    }
    DB::table("usuarios_cursos")->insert($dataUsuarios);
    return redirect()->back()->with('agregado', 'Usuario agregado a curso');
}
```

Ilustración 69 función store()

Se utiliza la función **validate()** del objeto **\$request** para asegurarse de que el campo **usuarios** esté presente en la solicitud y sea un arreglo. Esto asegura que los usuarios proporcionados estén en un formato adecuado antes de procesarlos. Si el campo **usuarios** no está presente o no es un arreglo, la validación fallará y se mostrarán los errores correspondientes.

Se inicializa un arreglo vacío llamado **\$dataUsuarios**, que se utilizará para almacenar la información de los usuarios que se agregarán al curso.

Se itera sobre el arreglo **\$request->usuarios**, que contiene los **ID** de los usuarios que se agregarán al curso. Por cada usuario en el arreglo, se crea un arreglo **\$consulta** que contiene la información necesaria para agregar al usuario al curso. Esta información incluye el **usuario_id**, el **curso_id**, **fecha_inicio** y **fecha_termino**, que se obtienen de la solicitud (**\$request**). La variable **\$usuario** representa el **ID** del usuario actual que está siendo procesado en el bucle.

Se agrega cada **\$consulta** al arreglo **\$dataUsuarios** utilizando la función **array_push()**. Esto acumula la información de todos los usuarios que se agregarán al curso en el arreglo **\$dataUsuarios**.

Una vez que se han recopilado los datos de los usuarios a agregar al curso en el arreglo **\$dataUsuarios**, se utiliza la consulta **DB::table("usuarios_cursos")->insert(\$dataUsuarios);** para insertar los registros en la tabla **usuarios_cursos**, los registros se agregarán correctamente a esta tabla.

Después de insertar los registros en la tabla, la función redirige al usuario de vuelta a la página anterior (**redirect()->back()**) y agrega un mensaje de sesión con el nombre 'agregado' y el valor 'Usuario agregado a curso'. Esto mostrará un mensaje de éxito al usuario indicando que los usuarios fueron agregados correctamente al curso.

Desarrollo de la interfaz de eliminar usuarios inscritos al curso:

- Diseño:

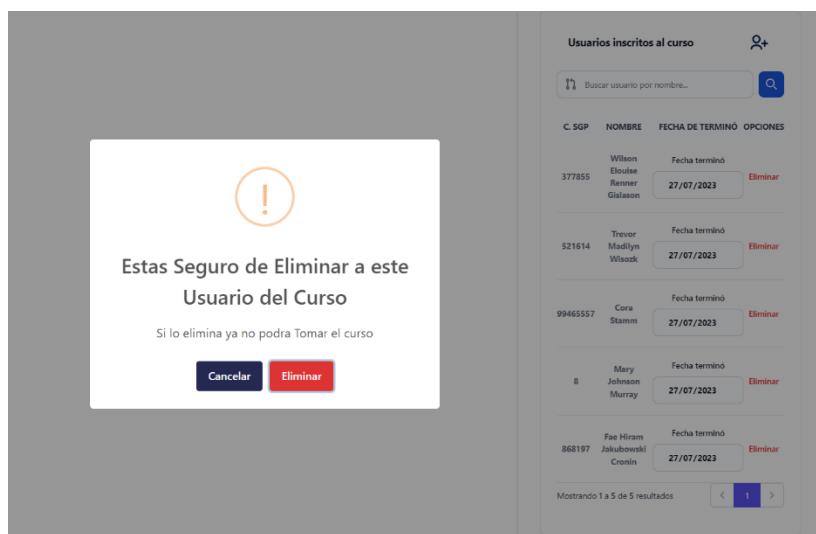


Ilustración 70 interfaz de eliminar usuarios.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (cursosinternos/CursoController) una vez confirmado que está seguro de eliminarlo:

- Función destroyUser():

```
public function destroyUser(Request $request, string $id)
{
    $user = User::find($id);
    if (is_null($user)) {
        return redirect()->back();
    }
    $curso_id = $request->curso_id;
    foreach ($user->cursos as $curso) {
        if ($curso->id_curso == $curso_id) {
            $user->cursos()->detach($curso_id);
            return redirect()->back()->with('eliminado', 'Eliminado Correctamente');
        } else{
            return redirect()->back()->with('error', 'No se pudo eliminar, vuelve a intentarlo');
        }
    }
}
```

Ilustración 71 función destroyUser()

Se busca un registro específico en la tabla **users** (asumiendo que hay un modelo **User**) utilizando el id proporcionado en la variable **\$id**. Si no se encuentra ningún registro con el id proporcionado, la función redirige al usuario a la página anterior (**redirect()>back()**). Esto puede suceder si el usuario con el id especificado no existe en la base de datos.

Se obtiene el valor del campo **curso_id** enviado en la solicitud (**\$request->curso_id**) y se guarda en la variable **\$curso_id**.

Se itera sobre los cursos asociados con el usuario utilizando la relación cursos en el modelo **User**. Se asume que existe una relación muchos a muchos entre los usuarios y los cursos, donde un usuario puede estar inscrito en varios cursos.

Dentro del bucle, se verifica si el **id_curso** del curso actual es igual a **\$curso_id** (el curso que se quiere eliminar). Si es así, se utiliza la función **detach(\$curso_id)** en la relación **cursos()** del usuario para eliminar la relación entre el usuario y el curso específico.

Si el curso se elimina correctamente, la función redirige al usuario de vuelta a la página anterior (**redirect()>back()**) y muestra un mensaje de éxito con el nombre '**eliminado**' y el valor '**Eliminado Correctamente**'.

Si el curso no se encuentra en la lista de cursos del usuario, se redirige al usuario de vuelta a la página anterior y muestra un mensaje de error con el nombre '**error**' y el valor '**No se pudo eliminar, vuelve a intentarlo**'.

Sprint 7: Diseño y programación de las interfaces Inicio cursos internos (ver avances por usuario en PDF), Reporte general.

Desarrollo de la interfaz de inicio cursos internos (ver avances por usuario en PDF):

- Diseño:

The screenshot shows a user interface titled "INFORMACIÓN POR EMPLEADO". On the left is a vertical sidebar with icons for navigation. The main area has a search bar at the top. A table below lists six employees with their course statistics. The columns are: EMPLEADO, T. CURSOS, APROBADOS, REPROBADOS, PENDIENTE POR INICIAR, EN CURSO, PROGRESO TOTAL, and VER DETALLES. Each row includes a small profile icon next to the employee name.

EMPLEADO	T. CURSOS	APROBADOS	REPROBADOS	PENDIENTE POR INICIAR	EN CURSO	PROGRESO TOTAL	VER DETALLES
WILSON ELOUISE RENNER GISLASON	1	0	0	1	0	0%	
TREVOR MADILYN WISOZK	1	0	0	1	0	0%	
CORA STAMM	1	0	0	1	0	0%	
MARY JOHNSON MURRAY	1	0	0	1	0	0%	
DEMETRIUS JULIET CHAMPLIN BOTSFORD	0	0	0	0	0	0%	
FAE HIRAM JAKUBOWSKI CRONIN	1	0	0	1	0	0%	

Ilustración 72 interfaz de inicio cursos internos.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (cursosinternos/AvanceUsuarioController) donde se utilizan 4 funciones las cuales son las siguientes:

○ Función index():

```

public function index(Request $request)
{
    $buscar = $request->buscador;
    $usuarios = User::where(function ($query) use ($buscar) {
        $query->where('nombre', 'like', $buscar . "%")
            ->orWhere('segundo_nombre', 'like', $buscar . "%")
            ->orWhere('apellido_paterno', 'like', $buscar . "%")
            ->orWhere('apellido_materno', 'like', $buscar . "%")
            ->orWhere('id_sgp', 'like', $buscar . "%");
    })
        ->where('rol', '=', 1)
        ->orderBy('id_usuario', 'asc')
        ->get();
    if (is_null($usuarios)) {
        return redirect()->back();
    }
    $user = $this->getCursosUsuarios($usuarios);
    $datosUsuarios = [];
    foreach ($user as $UserCursos) {
        $cursosUsuario = []; // Arreglo para almacenar los cursos del usuario actual
        foreach ($UserCursos['cursos'] as $curso) {
            $calificacionContenido = 0;
            $totalContenido = 0;
            $contenidosCompletados = 0;
            $totalProgresoExamen = 0;
            foreach ($curso['lecciones'] as $seccion) {
                $totalContenido += $seccion['contenidos']->count(); // cantidad de contenidos
                foreach ($seccion['contenidos'] as $contenido) {
                    foreach ($contenido['examenes'] as $examen) {
                        foreach ($examen['calificaciones'] as $calificacion) {
                            $calificacionContenido += $calificacion['calificacion'];
                            if ($calificacion['calificacion'] >= 0) {
                                $contenidosCompletados++;
                            }
                        }
                    }
                }
            }
            $calificacionExamenFinal = 0;
            $totalProgresoExamen = 0;
            $totalExamenFinal = $curso['examenfinal']->count(); // cantidad real de exámenes finales disponibles
            if ($totalExamenFinal > 0) {
                foreach ($curso['examenfinal'] as $examenFinal) {
                    $calificacionExamenFinal += $examenFinal['calificacion'];
                }
                $calificacionExamenFinal = $calificacionExamenFinal / $totalExamenFinal;
                $totalProgresoExamen += $calificacionExamenFinal;
            }
            $cursosUsuario[] = [
                'id' => $UserCursos['usuario_id'],
                'nombre' => $UserCursos['usuario_N'],
                'nombreS' => $UserCursos['usuario_S'],
                'nombreAP' => $UserCursos['usuario_AP'],
                'nombreAM' => $UserCursos['usuario_AM'],
                'cursos' => $UserCursos,
                'aprobados' => $aprobados,
                'reprobados' => $reprobados,
                'pendientes' => $pendientes,
                'encurso' => $encurso,
                'progresoTotal' => $totalProgresoExamen
            ];
        }
    }
    $datosUsuarios[] = [
        'cursosUsuario' => $cursosUsuario,
        'aprobados' => $aprobados,
        'reprobados' => $reprobados,
        'pendientes' => $pendientes,
        'encurso' => $encurso,
        'progresoTotalCurso' => $totalProgresoExamen
    ];
    return view('cursosinternos.index', compact('datosUsuarios'));
}

```

Ilustración 73 función index()

- Función getExamenUsuario (\$examen, \$id_usuario):

```
private function getExamenUsuario($examen, $id_usuario) //funcion para obtener los datos del examen por contenido
{
    $caliContenido = 0;
    $contenidoCompleto = 0;
    return $resultado = $examen->map(function ($examen) use ($id_usuario, &$caliContenido, &$contenidoCompleto) [
        $usuarios = $examen->usuarios->map(function ($examUser) use ($id_usuario, &$caliContenido, &$contenidoCompleto) {
            if ($examUser->id_usuario == $id_usuario) {
                $calificacion = $examUser->pivot->calificacion ?? 0;
                $caliContenido = $calificacion;
                $contenidoCompleto++;
                return [
                    'usuario_id' => $id_usuario,
                    'usuario_nombre' => $examUser->nombre,
                    'calificacion' => $caliContenido,
                    'contenidoCompleto' => $contenidoCompleto
                ];
            }
            return [
                'usuario_id' => $examUser->id_usuario,
                'usuario_nombre' => $examUser->nombre,
                'calificacion' => 0,
                'contenidoCompleto' => 0
            ];
        });
        return [
            'examen_id' => $examen->id_examen,
            'calificacionContenido' => $usuarios->where('usuario_id', '=', $id_usuario)
        ];
    ]);
}
```

Ilustración 74 función getExamenUsuario(\$examen, \$id_usuario)

- Función getExamenfinal(\$examen, \$id_usuario):

```
private function getExamenfinal($examen, $id_usuario) //funcion para obtener los datos del examen final
{
    $caliexam = 0;
    $examencompleto = 0;
    return $resultado = $examen->map(function ($examen) use ($id_usuario, &$caliexam, &$examencompleto) [
        $usuariosexamen = $examen->usuarios->map(function ($examuserfinal) use ($id_usuario, &$caliexam, &$examen, &$examencompleto) {
            if ($examuserfinal->id_usuario == $id_usuario) {
                $calificacion = $examuserfinal->pivot->calificacion ?? 0;
                $caliexam = $calificacion;
                $examencompleto++;
                return [
                    'usuario_id' => $id_usuario,
                    'usuario_nombre2' => $examuserfinal->nombre,
                    'calificacionexamenFinal' => $caliexam,
                    'progresoexamen' => $examencompleto
                ];
            }
            return [
                'usuario_id' => $examuserfinal->id_usuario,
                'usuario_nombre2' => $examuserfinal->nombre,
                'calificacionexamenFinal' => 0,
                'progresoexamen' => $examencompleto
            ];
        });
        return [
            'examen_idfinal' => $examen->id_examen,
            'calificacionExamen' => $usuariosexamen->where('usuario_id', '=', $id_usuario)
        ];
    ]);
}
```

Ilustración 75 función getExamenfinal(\$examen, \$id_usuario)

- Función getCursosUsuarios(\$usuarios):

```

private function getCursosUsuarios($usuarios) //funcion que obtiene todos los datos del usuario mapeado
{
    return $result = $usuarios->map(function ($usuario) {
        $usuario_id = $usuario->id_usuario;
        $usuario_nombre = $usuario->nombbre;
        $usuario_nombreS = $usuario->segundo_nombbre;
        $usuario_nombreAP = $usuario->apellido_paterno;
        $usuario_nombreAM = $usuario->apellido_materno;
        $rol = $usuario->rol;

        $cursos = $usuario->cursos->map(function ($curso) use ($usuario_id) {
            $curexamefina = $curso->examen;
            $finalexam = $this->getExamenfinal($curexamefina, $usuario_id);
            $lecciones = $curso->lecciones->map(function ($lección) use ($usuario_id) {
                $contenidos = $lección->contenido->map(function ($contenido) use ($usuario_id) {
                    $examen1 = $contenido->examen;
                    $cursUsuario = $this->getExamenUsuario($examen1, $usuario_id);

                    return [
                        'contenido_id' => $contenido->id_contenido,
                        'nombre' => $contenido->nombbre,
                        'examen' => $cursUsuario,
                    ];
                });
                return [
                    'lección_id' => $lección->id_lección,
                    'nombre_lección' => $lección->nombbre,
                    'contenidos' => $contenidos,
                ];
            });
            return [
                'curso_id' => $curso->id_curso,
                'curso_nombre' => $curso->nombbre,
                'examenfinal' => $finalexam,
                'lección' => $lecciones,
            ];
        });

        return [
            'usuario_id' => $usuario_id,
            'usuario_N' => $usuario_nombre,
            'usuario_S' => $usuario_nombreS,
            'usuario_AP' => $usuario_nombreAP,
            'usuario_AM' => $usuario_nombreAM,
            'rol' => $rol,
            'cursos' => $cursos,
        ];
    });
}

```

Ilustración 76 función getCursosUsuarios(\$usuarios)

- Función getCursosUsuarios(\$usuarios):

La función comienza con un **return** que indica que va a devolver un resultado.

\$result = \$usuarios->map(function (\$usuario) { ... });: Aquí se está usando el método **map()** en la colección **\$usuarios**. El método **map()** itera sobre cada elemento del array **\$usuarios** y ejecuta la función anónima definida dentro de **map()** para cada elemento.

Dentro del primer **map()**, se realiza el mapeo de cada **\$usuario** de la colección **\$usuarios** y se extraen algunas propiedades de cada usuario, como su **id_usuario**, **nombre**, **segundo_nombre**, **apellido_paterno**, **apellido_materno** y **rol**.

Luego, se accede a la propiedad cursos de cada **\$usuario**, que es una colección de datos asociada al usuario, y se aplica otro **map()** para cada curso.

En el segundo **map()** de cursos, se extraen propiedades del curso actual, como **id_curso** y **nombre**, y se realiza una llamada a la función **getExamenfinal()** pasando el valor de **\$curexamefina** y **\$usuario_id** como argumentos para obtener el resultado en la variable **\$finalexam**.

Dentro del segundo **map()** también se accede a la propiedad lecciones de cada curso y se aplica otro **map()** para cada lección.

En el tercer **map()**, se extraen propiedades de la lección actual, como **id_leccion** y **nombre_leccion**. Luego, se accede a la propiedad contenido de la lección, y se aplica otro **map()** para cada contenido.

En el cuarto **map()**, se extraen propiedades del contenido actual, como **id_contenido** y **nombre**. También se realiza una llamada a la función **getExamenUsuario()** pasando el valor de **\$examen1** y **\$usuario_id** como argumentos para obtener el resultado en la variable **\$cursUsuario**.

Finalmente, en el último **map()** de contenidos, se devuelve un array con la información del contenido actual, incluyendo **id_contenido**, **nombre** y el resultado de **\$cursUsuario**.

Todas las iteraciones anteriores resultan en la construcción de un array estructurado con información relevante de cada **usuario**, sus **cursos**, **lecciones** y **contenidos**. Al final de la función, se devuelve este array completo.

La función se repite para cada usuario en la colección **\$usuarios**, y cada resultado parcial se agrega a un array llamado **\$result**.

Una vez que se procesan todos los usuarios, la función devuelve el array **\$result** que contiene la estructura completa con la información de todos los **usuarios**, **cursos**, **lecciones** y **contenidos**.

- Función getExamenfinal(\$examen, \$id_usuario):

Se definen dos variables **\$caliexam** y **\$examencompleto** inicializadas en **0**. Estas variables se utilizan para llevar el registro de la calificación total del examen y el progreso del examen para un usuario específico.

\$resultado = \$examen->map(function (\$examen) use (\$id_usuario, &\$caliexam, &\$examencompleto) { ... });: Aquí se está usando el método **map()** en la colección **\$examen**. El método **map()** itera sobre cada elemento del array **\$examen** y ejecuta la función anónima definida dentro de **map()** para cada elemento.

Dentro del primer **map()**, se realiza el mapeo de cada **\$examen** de la colección **\$examen** y se accede a la propiedad **usuarios** de cada examen, que es otra colección de datos asociada al examen.

Luego, se aplica otro **map()** en la colección de usuarios del examen actual (**\$usuariosexamen = \$examen->usuarios->map(function (\$examuserfinal) use (\$id_usuario, &\$caliexam, &\$examen, &\$examencompleto) { ... });**) para procesar cada usuario del examen.

Dentro del segundo **map()** de usuarios del examen, se verifica si el **\$id_usuario** pasado como argumento coincide con el **id_usuario** del usuario actual (**\$examuserfinal**). Si es así, se obtiene la calificación del usuario en ese examen a través de **\$examuserfinal->pivot->calificacion** y se almacena en la variable **\$calificacion**.

Se actualiza el valor de **\$caliexam** con la calificación obtenida y se incrementa en 1 el valor de **\$examencompleto**, que representa el progreso del examen para el usuario actual.

Se devuelve un array que contiene la información del usuario, incluyendo su **id_usuario**, **nombre**, **calificacionexamenFinal** (que es la calificación obtenida) y **progresosexamen** (que representa el progreso del examen para el usuario).

Si el **id_usuario** del usuario actual no coincide con **\$id_usuario**, se devuelve un array con la información del usuario, pero con **calificacionfinalexamen** en **0** (ya que no es el usuario para el que se busca la calificación).

Al final del segundo **map()** de usuarios del examen, se obtiene una colección de usuarios procesados con su información correspondiente.

Luego, se devuelve un array que contiene la información del examen actual, incluyendo su **id_examen** y la colección de usuarios del examen que coinciden con el **\$id_usuario**.

La función se repite para cada examen en la colección **\$examen**, y cada resultado parcial se agrega a un array llamado **\$resultado**.

Una vez que se procesan todos los exámenes, la función devuelve el array **\$resultado** que contiene la estructura completa con la información de los exámenes y las calificaciones correspondientes de los usuarios.

- Función getExamenUsuario (\$examen, \$id_usuario):

Se definen dos variables **\$caliContenido** y **\$contenidocompleto** inicializadas en **0**. Estas variables se utilizan para llevar el registro de la calificación total del contenido y el progreso del contenido para un usuario específico.

\$resultado = \$examen->map(function (\$examen) use (\$id_usuario, &\$caliContenido, &\$contenidocompleto) { ... });: Aquí se está usando el método **map()** en la colección **\$examen**. El método **map()** itera sobre cada elemento del array **\$examen** y ejecuta la función anónima definida dentro de **map()** para cada elemento.

Dentro del primer **map()**, se realiza el mapeo de cada **\$examen** de la colección **\$examen** y se accede a la propiedad **usuarios** de cada examen, que es otra colección de datos asociada al examen.

Luego, se aplica otro **map()** en la colección de usuarios del examen actual (**\$usuarios = \$examen->usuarios->map(function (\$examUser) use (\$id_usuario, &\$caliContenido, &\$contenidocompleto) { ... });**) para procesar cada usuario del examen.

Dentro del segundo **map()** de usuarios del examen, se verifica si el **\$id_usuario** pasado como argumento coincide con el **id_usuario** del usuario actual (**\$examUser**). Si es así, se obtiene la calificación del usuario en ese examen a través de **\$examUser->pivot->calificacion** y se almacena en la variable **\$calificacion**.

Se actualiza el valor de **\$caliContenido** con la calificación obtenida y se incrementa en **1** el valor de **\$contenidocompleto**, que representa el progreso del contenido para el usuario actual.

Se devuelve un array que contiene la información del usuario, incluyendo su **id_usuario**, **nombre**, **calificacion** (que es la calificación obtenida en el examen) y **contenidocompleto** (que representa el progreso del contenido para el usuario).

Si el **id_usuario** del usuario actual no coincide con **\$id_usuario**, se devuelve un array con la información del usuario, pero con **calificacion** y **contenidocompleto** en **0** (ya que no es el usuario para el que se busca la calificación).

Al final del segundo **map()** de usuarios del examen, se obtiene una colección de usuarios procesados con su información correspondiente.

Luego, se devuelve un array que contiene la información del examen actual, incluyendo su **id_examen** y la colección de usuarios del examen que coinciden con el **\$id_usuario**.

La función se repite para cada examen en la colección **\$examen**, y cada resultado parcial se agrega a un array llamado **\$resultado**.

Una vez que se procesan todos los exámenes, la función devuelve el array **\$resultado** que contiene la estructura completa con la información de los exámenes y las calificaciones correspondientes de los usuarios para el contenido.

- Función index():

La función comienza extrayendo un valor llamado **\$buscar** del objeto **\$request**, que es el valor ingresado por el usuario para realizar una búsqueda en la base de datos. A continuación, se realiza una consulta en la tabla '**users**' (usuarios) usando el modelo **User**. Se aplica una cláusula **where** con varias condiciones, utilizando el operador lógico **OR**. Se busca un usuario cuyo **nombre**, **segundo nombre**, **apellido paterno**, **apellido materno**, o el campo '**id_sgp**' coincida con el valor de búsqueda **\$buscar**. Además, se agrega una condición **where** para filtrar solo los usuarios con el campo '**rol**' igual a 1.

Se ordenan los resultados por el campo '**id_usuario**' en orden ascendente.

Se ejecuta la consulta con **->get()**, y los resultados se almacenan en la variable **\$usuarios**. Si no se encuentra ningún usuario que cumpla con las condiciones de búsqueda, la variable **\$usuarios** será **null**, y en ese caso, se redireccionará a la página anterior.

Si se encuentran usuarios que cumplan con los criterios de búsqueda, se procede a procesar la información.

Se llama a una función llamada **\$this->getCursosUsuarios(\$usuarios)** para obtener datos adicionales sobre los cursos internos de los usuarios encontrados. Esta función se encuentra en la misma clase explicada anteriormente.

Se inicializa una variable llamada **\$datosUsuarios** como un array vacío. Esta variable se usará para almacenar la información detallada de los usuarios y sus cursos internos.

Se utiliza un bucle **foreach** para iterar sobre el resultado de la función **\$this->getCursosUsuarios(\$usuarios)** (almacenado en **\$user**). Cada elemento de **\$user** representa un usuario con sus cursos internos.

Se inicializa una variable llamada **\$cursosUsuario** como un array vacío. Esta variable se usará para almacenar los cursos del usuario actual.

Se utiliza otro bucle **foreach** para iterar sobre los cursos del usuario actual. Dentro de este bucle, se realiza una serie de cálculos relacionados con las calificaciones y el progreso de los contenidos y exámenes de cada curso.

Se calcula el promedio de calificación para los contenidos (**\$promediocontenidofinal**) y el promedio de calificación para el examen final (**\$promedioExamenFinal**) para el curso actual.

Se calcula el progreso del contenido (**\$progresofinalcontenido**) y el progreso del examen final (**\$progresoexamenFINAL**) para el curso actual.

Se calcula el progreso total del curso, que combina el progreso del contenido y el progreso del examen final (**\$progesototalcurso**).

Se agrega la información del curso actual al array **'\$cursosUsuario'**.

Después de procesar todos los cursos para el usuario actual, se realiza otro conjunto de cálculos para determinar la cantidad de cursos aprobados, reprobados, pendientes y en curso. También se calcula el progreso total del usuario en todos sus cursos (**\$progresoTotalUsuario**).

Finalmente, se agrega la información del usuario actual, incluyendo sus cursos, a la variable **\$datosUsuarios**.

Después de procesar todos los usuarios, se devuelve la vista '**cursosinternos.index**', pasando la variable **\$datosUsuarios** como datos para ser utilizados en la vista.

Para pasar la información al PDF utilizamos las mismas funciones explicadas anteriormente dentro de un controlador llamado **ReporteAvancesController** y de igual manera mandamos el mismo array **\$datosUsuarios** dentro de la función **PDF** y renderizamos la información como en una vista.

- Función para mandar la información a un PDF:

```
$pdf = PDF::loadView('pdfs.reporteAvances', ['datosUsuarios' => $datosUsuarios]);
$pdf->setOptions(['defaultFont' => 'poppins']);
$pdf->setPaper('letter', 'landscape');

return $pdf->stream("avances.pdf");
}
```

Ilustración 77 función para mandar los datos al PDF.

PDF::loadView('pdfs.reporteAvances', ['datosUsuarios' => \$datosUsuarios]):

Aquí se utiliza el método **loadView()** del paquete **PDF** para cargar una vista llamada '**pdfs.reporteAvances**'. Además, se pasa un array asociativo que contiene datos a ser utilizados en la vista, y estos datos están almacenados en la variable **\$datosUsuarios**.

\$pdf->setOptions(['defaultFont' => 'poppins']): Se utiliza el método **setOptions()** del objeto **\$pdf** para establecer opciones adicionales. En este caso, se establece la opción '**defaultFont**' con el valor '**poppins**', lo que significa que el tipo de fuente predeterminada para el **PDF** será '**poppins**'.

\$pdf->setPaper('letter', 'landscape'):: Se utiliza el método **setPaper()** del objeto **\$pdf** para definir el tamaño y la orientación del papel en el **PDF**. En este caso, se establece el tamaño del papel en '**letter**' (carta, 8.5 x 11 pulgadas) y la orientación en '**landscape**' (horizontal).

return \$pdf->stream("avances.pdf");: Finalmente, se llama al método **stream()** del objeto **\$pdf** para generar el archivo **PDF** y mostrarlo en el navegador como una respuesta. El nombre del archivo **PDF** resultante se establece como "**avances.pdf**". Al usar **stream()**, el archivo **PDF** se muestra en el navegador para que el usuario pueda verlo o descargarlo directamente.

Desarrollo de la interfaz de Reporte general:

- Diseño:

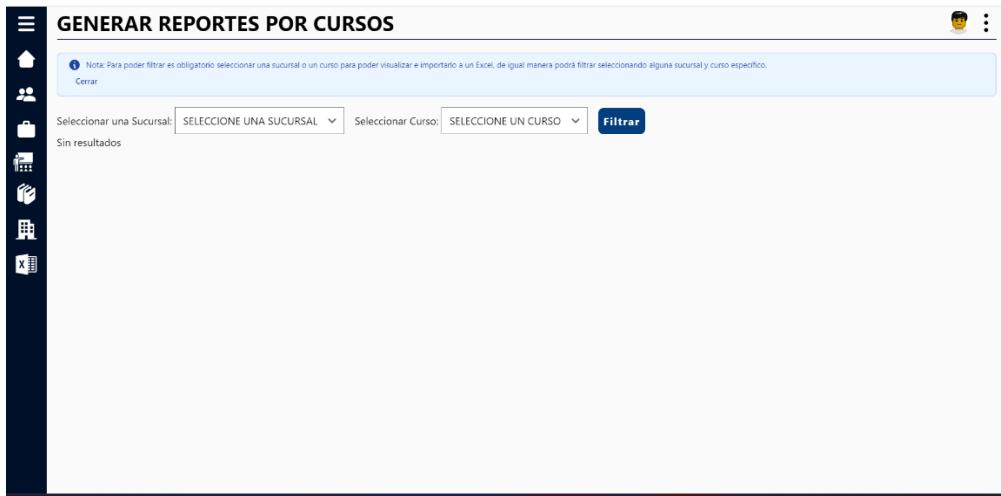


Ilustración 78 interfaz de reporte general.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/ReporteCursoInternoController), de la misma manera que la interfaz anterior.

Sprint 8: Diseño y programación de las interfaces empleado

El octavo sprint se basó en la creación y programación y revisión de las siguientes interfaces:

- Vista inicio
- Vista del curso interno
- Vista contenido
- Vista lección

Desarrollo de la interfaz de inicio empleado:

- Diseño:

The screenshot shows a user interface titled 'INICIO' (Login). At the top, there is a navigation bar with a home icon, the word 'INICIO', a search bar ('Buscar curso por nombre...'), and a 'Perfil' (Profile) icon. Below the navigation, a section titled 'Cursos: 2' (Courses: 2) is displayed, with a note 'Lista de los cursos inscritos' (List of registered courses). There are four tabs: 'Todos los cursos' (All courses), 'Cursos aprobados' (Approved courses), 'Cursos pendientes' (Pending courses), and 'Cursos reprobados' (Failed courses). The 'Todos los cursos' tab is selected. Below the tabs is a table with course details:

Imagen del curso	Nombre del curso	N.º Lecciones	Estatus del curso	Estado del curso	Inicia el:	Termina el:	Calificación del curso	Progreso del curso	Opciones
	Prueba	1	Expirado	Aprobado	2023-07-24	2023-07-27	100.00	<div style="width: 100%;">100.00%</div>	Curso expirado Descargar certificado
	Prueba 2	1	Pendiente por iniciar	Aprobado	2023-10-18	2023-10-26	100.00	<div style="width: 100%;">100.00%</div>	Pendiente por iniciar Descargar certificado

Ilustración 79 interfaz de inicio empleado.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/EmpleadoController), de la siguiente manera:

- Función index(\$request);

```

public $intentos = 3;
public $intentoHecho = 1;

public function index(Request $request)
{
    $calificacionescursos = [];
    $buscar = $request->buscador1;
    $estadoseleccionado = $request->estado ?? "Todos";
    $cursosfiltrados = [];
    foreach (Auth::user()->userCursos as $curso) {
        if (stripos($curso->nombre, $buscar) !== false) {
            $progresoleccion2 = 0;
            $calificacionTotal = 0;
            $examenCalifinal = 0;
            $calificacionMaxima = 0;
            $progresoTotal = 0;
            $progresototalexamen = 0;
            $totalLeccionesFinal = $curso->lecciones->count();
            foreach ($curso->lecciones as $lección) {
                $progresolección = 0;
                $progre = 0;
                $calificacionLección = 0;
                $calificacionLección2 = 0;
                $totalContenido = $lección->contenido->count();
                foreach ($lección->contenido as $contenido) {
                    $examen = $contenido->examen()->first(); // Retrieve the first exam
                    $examenCurso = $curso->examen()->first();
                    if (count($examen->usuarios ?? []) > 0) {
                        foreach ($examen->usuarios as $calificacion) {
                            if ($calificacion->pivot->usuario_id == Auth::user()->id_usuario) {
                                $calificacion = $calificacion->pivot->calificacion ?? 0; // Get the "calificación" property
                                $calificacionLección += $calificacion;
                            }
                        }
                    }
                }
            }
            $calificacionescursos[] = [
                'curso' => $curso,
                'calificacionTotal' => $calificacionTotal,
                'examenCalifinal' => $examenCalifinal,
                'calificacionMaxima' => $calificacionMaxima,
                'progresoTotal' => $progresoTotal,
                'progresototalexamen' => $progresototalexamen,
                'totalLeccionesFinal' => $totalLeccionesFinal,
                'calificacionLección' => $calificacionLección
            ];
        }
    }
    $request->session()->put('cursosfiltrados', $cursosfiltrados);
    return view('empleados.index', compact('cursosfiltrados', 'calificacionescursos'));
}

```

Ilustración 80 función index(\$request).

```

        }
        if ($examen_usuarios[]->where('examen_id', $examen->examen()->first()->id_examen)->exists() and $examen->usuarios()->where('usuario_id', Auth::user()->id_usuario)->exists()) {
            $progresoseleccion1;
        }
    }
}

if (count($examsCursos_usuarios) > 1) {
    foreach ($examsCursos_usuarios as $calCu) {
        if ($calCu->pivot->usuario_id == Auth::user()) {
            $calificacion2 = $calCu->pivot->calificacion ?? 0; // Get the "calificacion" property
            $calificacioneleccion2 = $calificacion2;
            if ($examsCursos_usuarios()->where('examen_id', $curso->examen()->first()->id_examen)->exists() and $examen->usuarios()->where('usuario_id', Auth::user()->id_usuario)->exists()) {
                $progresoseleccion2 = 20;
            }
        }
    }
}
$calificacionTotal += $calificacioneleccion;
$calificacionContenido = $calificacionTotal;
$progresototal = $progresoseleccion;
$progresototalexamen = $progresoseleccion2;
$examenCalifinal = $calificacioneleccion;

$promedioCalificacion = $calificacionMaxima > 0 ? ($calificacionTotal * 300) / ($calificacionMaxima * 300) : 0;
$promedioContenido = $promedioCalificacion > 0 ? ($promedioCalificacion * 60) / 100 : 0;
$promedioIncompleto = $promedioContenido * $promedioFinal;
$promedioProgreso = $calificacionMaxima > 0 ? ($progresototal / $calificacionMaxima) * 100 : 0;
$promedioProgresoContenido = $promedioProgreso > 0 ? ($promedioProgreso * 80) / 100 : 0;
$promedioInadecuado = $promedioIncompleto + $promedioProgresoContenido;
$numeroFormato = number_format($promedioInadecuado, 2);
$numeroFormato = number_format($promedioInALCURSOCOMPLETO, 2);

// Averiguar la condición para determinar el estado del curso
$estado = "";
if ($promedioInALCURSOCOMPLETO >= 80 and $promedioInadecuado < 100) {
    $estado = "Aprobado";
} else if ($promedioInadecuado < 100) {
    $estado = "Reprobado";
}

$fechaInicioCurso = $curso->pivot->fecha_inicio;
$fechaTerminoCurso = $curso->pivot->fecha_termino;
$fechaActual = date("Y-m-d");

if ($fechaActual > $fechaInicioCurso && $fechaActual <= $fechaTerminoCurso) {
    // El curso está vigente
    // Pasa la variable a la vista para indicar que el curso está vigente
    $cursoVigente = true;
} else {
    // El curso no está vigente
    // Pasa la variable a la vista para indicar que el curso no está vigente
    $cursoVigente = false;
}

$calificacionesCursos[] = [
    'curso' => $curso,
    'calificacion' => $numeroFormato,
    'progreso' => $numero,
    'estado' => $estado,
    'lecciones' => $totalleccionesfinal,
    'estatus' => $cursoVigente,
    'fecha' => $fechaActual,
    'fecha_inicio' => $fechaInicioCurso,
    'fecha_termino' => $fechaTerminoCurso,
];
}

return view('VistasEmpleados.inicio', compact('calificacionesCursos', 'estadoSeleccionado'));
}

```

Ilustración 81 función index(\$request).

Comenzamos inicializando una matriz vacía llamada **\$calificacionesCursos**, que contendrá información sobre los cursos del usuario.

Recupera la consulta de búsqueda (**\$buscar**) y el estado del curso seleccionado (**\$estadoSeleccionado**) de la solicitud HTTP.

Inicializa una matriz vacía llamada **\$cursosFiltrados**.

Luego, la secuencia de comandos itera a través de cada uno de los cursos del usuario mediante un **foreachbucle**. Comprueba si el nombre del curso contiene la consulta de búsqueda utilizando la **striposfunción**.

Dentro del bucle, el script calcula varias métricas y datos relacionados con el progreso y el rendimiento del usuario en cada curso. Estas métricas incluyen:

\$progresoLeccion2: una variable para realizar un seguimiento del progreso general del curso (porcentaje).

\$calificacionTotal: Puntuación total obtenida por el usuario en todas las lecciones del curso.

\$examenCalifinal: Puntuación obtenida por el usuario en el examen final del curso.

\$calificacionMaxima: Puntuación máxima posible en el curso (calculada contando el número de contenidos/lecciones).

\$progresoTotal: Progreso total del usuario en términos de lecciones completadas (recuento de lecciones completadas).

\$progresototalexamen: Avance del usuario específicamente en el examen final (porcentaje).

\$totalLeccionesfinal: Número total de lecciones en el curso.

Después de calcular las métricas, el script calcula varios promedios y porcentajes relacionados con el rendimiento, el progreso y la calificación final del usuario en cada curso. Las variables utilizadas para estos cálculos incluyen:

\$promedioCalificacion: Puntuación media en porcentaje en todas las lecciones.

\$promedioContenido: Media ponderada de la puntuación del contenido (60 % de la nota total).

\$promedioExaFinal: Media ponderada de la nota del examen final (40% de la nota total).

\$promedioFINALCURSOCOMPLETO: Promedio ponderado total de las calificaciones del contenido y del examen final.

\$promedioProgreso: Progreso promedio en porcentaje basado en lecciones completadas.

\$promedioProgresoContenido: Promedio ponderado del progreso del contenido (80% de la nota total).

\$promedioFinaldelcurso: Promedio ponderado total del progreso y calificación del examen final.

Con base en los promedios ponderados calculados, el guión determina el estado del curso (**\$estado**). Los posibles estados son:

"**Aprobado**" (Aprobado): Si el promedio ponderado es mayor o igual a 80 y el progreso en el examen final es del 100%.

"**Pendiente**" (Pending): Si el progreso en el examen final es inferior al 100%.

"**Reprobado**" (Failed): Si el promedio ponderado es menor a 80.

Luego, el script verifica si la fecha actual se encuentra dentro de las fechas de inicio y finalización del curso (**\$fechalinicioCurso** **\$fechaTerminoCurso**) para determinar si el curso está actualmente activo o no. Establece la **\$cursoVigentevariable** en consecuencia.

Finalmente, el script agrega toda la información calculada a la **\$calificacionesCursos matriz**, incluidos los detalles del curso, las métricas de rendimiento y el estado del curso.

Una vez realizado esto, renderiza la vista **vistasEmpleados.inicio**, pasando la **calificacionesCursosmatriz** y la **estadoSeleccionadovariable** como datos que se utilizarán en la vista.

Desarrollo de la interfaz curso interno empleado:

- Diseño:

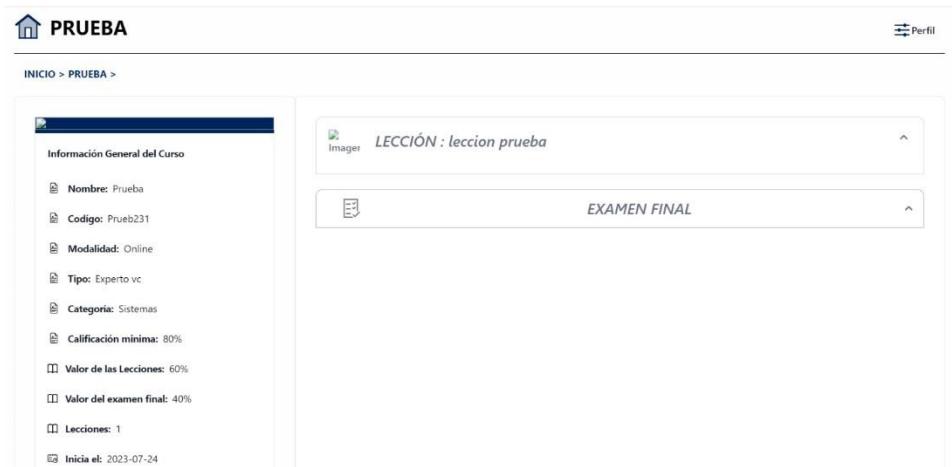


Ilustración 82 interfaz de curso interno empleado.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/EmpleadoController), de la siguiente manera:

- Función show(\$id):

```
public function show(string $id)
{
    $curso = Curso::find($id);
    if (is_null($curso)) {
        return redirect()->back();
    }
    $categoria = Categoria::all();
    $modalidad = ModalidadCurso::all();
    $tipo = TipoCurso::all();
    $usuarios = User::all();

    return view('vistasEmpleados.cursosEmpleados.indexCurso', compact('curso', 'modalidad', 'tipo', 'usuarios', 'categoria'));
}
```

Ilustración 83 funcion show(\$id).

Esta función comienza con la variable `$curso = Curso::find($id);`: Aquí se obtiene un curso específico de la base de datos utilizando el modelo Curso y su método estático `find()`. El `find()` toma como argumento el `$id` del curso que se quiere obtener. Si el curso no existe (es nulo), se detiene la ejecución del código y se redirige al usuario a la página anterior utilizando `redirect()->back()`.

Después de asegurarse de que el curso existe, se procede a obtener datos adicionales de otras tablas de la base de datos relacionadas con el curso:

`$categoria = Categoria::all();`: Aquí se obtienen todas las categorías de la tabla `categorias` y se asignan a la variable `$categoria`.

`$modalidad = ModalidadCurso::all();`: Se obtienen todas las modalidades de curso de la tabla `modalidad_cursos` y se asignan a la variable `$modalidad`.

`$tipo = TipoCurso::all();`: Se obtienen todos los tipos de curso de la tabla `tipo_cursos` y se asignan a la variable `$tipo`.

`$usuarios = User::all();`: Se obtienen todos los usuarios de la tabla `users` y se asignan a la variable `$usuarios`.

Finalmente, se carga una vista llamada `vistasEmpleados.cursosEmpleados.indexCurso` y se le pasan los datos obtenidos a través de la función `compact()`. La función `compact()` crea un `array` asociativo con el nombre de las variables (`$curso`, `$modalidad`, `$tipo`, `$usuarios` y `$categoria`) como claves y sus valores como valores. Estos datos estarán disponibles en la vista para ser utilizados en la presentación de la información relacionada con el curso y los datos relacionados.

Desarrollo de la interfaz curso interno empleado:

- Diseño:

The screenshot displays a web application interface for managing content. At the top, there's a header with a home icon, the text 'CONTENIDO PRUEBA', and a 'Perfil' link. Below the header, the URL 'PRUEBA > LECCION PRUEBA > CONTENIDO PRUEBA' is visible. The main area is divided into two sections: 'CONTENIDO PRUEBA:' on the left and 'LECCIÓN: LECCION PRUEBA' on the right.

CONTENIDO PRUEBA:

- INICIO** tab is selected.
- Cursos:** 2 (List of courses inserted).
- Buttons: Todos los cursos, Cursos aprobados, Cursos pendientes, Cursos reprobados.
- Table headers: Imagen del curso, Nombre del curso, N.º Lecciones, Estatus del curso, Estado del curso, Inicia el:, Termina el:, Califica del cu.
- Table data for one course: Prueba, 1, Activo, Aprobado, 2023-07-24, 2023-08-27, 100%.

LECCIÓN: LECCION PRUEBA

- CONTENIDOS DE LA LECCIÓN** section.
- CONTENIDO PRUEBA** button.

Ilustración 84 interfaz de contenido empleado.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/EmpleadoController), de la siguiente manera:

- Función ver(\$id):

```
public function ver(string $id)
{
    $contenido = Contenido::find($id);
    if (is_null($contenido)) {
        return redirect()->back();
    }
    $archivo = public_path($contenido->media[0]->url); //Ruta al archivo que deseas obtener la extensión
    $extension = File::extension($archivo);
    $lección = Lección::find($contenido->lección_id);
    return view('vistasEmpleados.vistacontenido.index', compact('contenido', 'lección', 'extension'));
}
```

Ilustración 85 función ver(\$id).

Esta función comienza con la variable **\$contenido = Contenido::find(\$id);**: Se obtiene un contenido específico de la base de datos utilizando el modelo **Contenido** y su método estático **find()**. El **find()** toma como argumento el **\$id** del contenido que se desea obtener. Si el contenido no existe (es nulo), se detiene la ejecución del código y se redirige al usuario a la página anterior utilizando **redirect()->back()**.

Luego de asegurarse de que el contenido existe, se procede a obtener la extensión del archivo multimedia asociado al contenido:

\$archivo = public_path(\$contenido->media[0]->url);: Se construye la ruta completa al archivo multimedia utilizando la URL almacenada en el primer elemento de la relación media del contenido. **public_path()** es una función de **Laravel** que proporciona la ruta completa al directorio **public** del proyecto.

\$extension = File::extension(\$archivo);: Se utiliza la clase **File** de **Laravel** para obtener la extensión del archivo multimedia. Esto se logra utilizando el método **extension()** de la clase **File** y pasando la ruta del archivo como argumento.

Después de obtener la extensión del archivo multimedia, se busca la lección asociada al contenido y se obtiene la información relacionada:

\$lección = Lección::find(\$contenido->lección_id);: Se busca una lección específica en la base de datos utilizando el modelo **Lección** y el **lección_id** almacenado en el contenido. Finalmente, se carga una vista llamada **vistasEmpleados.vistacontenido.index** y se le pasan los datos obtenidos a través de la función **compact()**. La función **compact()** crea un **array** asociativo con el nombre de las variables (**\$contenido**, **\$lección** y **\$extension**) como claves y sus valores como valores. Estos datos estarán disponibles en la vista para ser utilizados en la presentación de la información relacionada con el contenido y la lección.

Desarrollo de la interfaz examen final empleado:

- Diseño:

Si el usuario no ha completado los 3 intentos máximos permitidos le aparecerá la siguiente interfaz:



Ilustración 86 interfaz de examen empleado.

Si el usuario ya ha completado los 3 intentos máximos permitidos le aparecerá la siguiente interfaz o si ha llegado a la calificación aprobatoria:



Ilustración 87 interfaz de examen empleado.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/EmpleadoController), de la siguiente manera:

- Función verExam(\$id);

```
public function verExam(string $id)
{
    $intentos = $this->intentos;
    $examen = Examen::find($id);
    if (is_null($examen)) {
        return redirect()->back();
    }
    $totalPreguntas = $examen->preguntas->count();
    return view('vistasEmpleados.vistaExamenEmpleado.index', compact('examen', 'intentos', 'totalPreguntas'));
}
```

Ilustración 88 función verExam(\$id).

Esta función comienza con la variable **\$intentos = \$this->intentos;**: Se asigna el valor de la propiedad \$intentos al objeto actual (probablemente el controlador). Esta propiedad puede ser una variable o valor que representa el número de intentos realizados por un empleado en un examen.

\$examen = Examen::find(\$id);: Se busca un examen específico en la base de datos utilizando el modelo Examen y su método estático **find()**. El **find()** toma como argumento el \$id del examen que se desea obtener. Si el examen no existe (es nulo), se detiene la ejecución del código y se redirige al usuario a la página anterior utilizando **redirect()->back()**.

Luego de asegurarse de que el examen existe, se procede a obtener información adicional sobre el examen:

\$totalPreguntas = \$examen->preguntas->count();: Se cuenta el número total de preguntas asociadas al examen accediendo a la relación preguntas del examen y utilizando el método **count()** para obtener el número total de elementos en la colección de preguntas. Finalmente, se carga una vista llamada **vistasEmpleados.vistaExamenEmpleado.index**, y se le pasan los datos obtenidos a través de la función **compact()**. La función **compact()** crea un array asociativo con el nombre de las variables (**\$examen**, **\$intentos** y **\$totalPreguntas**) como claves y sus valores como valores. Estos datos estarán disponibles en la vista para ser utilizados en la presentación de la información relacionada con el examen y los datos asociados.

Desarrollo de la interfaz preguntas del examen empleado:

- Diseño:



Ilustración 89 interfaz de preguntas examen empleado.

- Programación:

Esta interfaz está programada de la siguiente manera dentro del controlador (empleados/EmpleadoController), de la siguiente manera:

- Función validarExam(\$request, \$id):

```
public function validarExam(Request $request, string $id)
{
    $validatedData = $request->except('_token', 'total_pregunta');
    $totalQuestions = $request->total_pregunta;
    $id_user = Auth::user()->id_usuario;
    $user = User::find($id_user);
    if ($user == null) {
        return redirect()->back();
    }
    $examenn = Examen::find($id);
    if ($examenn == null) {
        return redirect()->back();
    }
    $id_examen = $examenn->id_examen;
    $correctAnswers = 0;
    foreach ($validatedData as $pregunta => $opcion) {
        if ($this->isAnswerCorrect($opcion)) {
            ++$correctAnswers;
        }
    }
    // Calcular el promedio
    $promedio = ($correctAnswers / $totalQuestions) * 100;
    if ($user->examen()->where('examen_id', $id_examen)->exists()) {
        foreach ($user->examen as $exaaa) {
            if ($exaaa->id_examen == $id_examen) {
                $intentoHecho = $exaaa->pivot->numero_intento;
                $user->examen()->updateExistingPivot($id_examen, ['numero_intento' => ++$intentoHecho, 'limite_intentos' => '3', 'calificacion' => $promedio]);
            }
        }
    } else {
        $user->examen()->attach($id_examen, ['numero_intento' => $this->intentoHecho, 'limite_intentos' => '3', 'calificacion' => $promedio]);
    }
    $examenID = $examenn->contenido_id;
    $examenDI = $examenn->curso_id;
    return view('vistasEmpleados.vistaExamenEmpleado.calificacionExamen', compact('promedio', 'examenID', 'examenDI', 'id_examen'));
}
```

Ilustración 90 función validarExam(\$id).

- Función insanswerCorrect(\$par):

```

private function isAnswerCorrect($par)
{
    $opcion = Opcion::find($par);
    if (is_null($opcion)) {
        return redirect()->back();
    }
    if ($opcion->correcta == 1) {
        return true;
    } else {
        return false;
    }
    // return $opcion[0]->correcta === $correcta;
}

```

Ilustración 91 función insanswerCorrect (\$par).

- Función insanswerCorrect (\$par):

Esta función comienza con la variable **\$opcion = Opcion::find(\$par);**: Se busca una opción específica en la base de datos utilizando el modelo **Opcion** y su método estático **find()**. El **find()** toma como argumento **\$par**, que probablemente es el identificador de la opción que se desea verificar.

if (is_null(\$opcion)) { return redirect()->back(); }: Se verifica si la opción existe en la base de datos. Si la opción no se encuentra (es nula), se detiene la ejecución del código y se redirige al usuario a la página anterior utilizando **redirect()->back()**. Esto podría ocurrir si el identificador proporcionado no corresponde a una opción válida. Luego de asegurarse de que la opción existe, se procede a verificar si es correcta o no:

if (\$opcion->correcta == 1) { return true; }: Se verifica si la propiedad correcta de la opción es igual a 1. Esto implica que la opción es correcta, y se devuelve el valor **true** para indicar que la opción es correcta.

else { return false; }: Si la opción no es correcta (la propiedad correcta no es igual a 1), se devuelve el valor **false** para indicar que la opción no es correcta.

- Función validarExam(\$id):

Se utiliza la función **except()** del objeto **\$request** para obtener todos los datos enviados en la solicitud, excepto el token **CSRF (_token)** y un campo llamado **total_pregunta**. La función **except()** devuelve un arreglo con los datos de la solicitud, excluyendo los campos mencionados.

Se obtiene el número total de preguntas en el examen a partir del campo **total_pregunta** enviada en la solicitud.

Se inicializa una variable **\$correctAnswers** con un valor de 0. Esta variable se utilizará para contar el número de respuestas correctas proporcionadas por el usuario.

Se itera sobre el arreglo **\$validatedData**, que contiene las respuestas enviadas por el usuario. Cada clave del arreglo representa el nombre de la pregunta y el valor representa la opción seleccionada por el usuario para esa pregunta.

Para cada respuesta proporcionada por el usuario, se llama a la función **isAnswerCorrect(\$opcion)** para verificar si la opción seleccionada es correcta o no.

Si la respuesta es correcta, se incrementa el contador **\$correctAnswers** en 1.

Después de contar el número total de respuestas correctas, se calcula el promedio dividiendo **\$correctAnswers** entre **\$totalQuestions** y multiplicándolo por 100.

Esto da como resultado el porcentaje de preguntas respondidas correctamente.

Finalmente, se carga una vista llamada '**cursosinternos.examenes.calificacion'** y se pasa como dato el valor de **\$promedio**. Esta vista mostrará la calificación obtenida por el usuario en el examen.

RESULTADOS

El proyecto "Sistema Web de Cursos Internos" ha sido un éxito para la Automotriz Bonn S.A DE C.V, brindando una forma efectiva y moderna de capacitar a sus empleados. A continuación, se presentan algunos de los resultados obtenidos:

- Aumento de la participación y la adopción:

Gracias a la accesibilidad y facilidad de uso del sistema web, se ha logrado un aumento significativo en la participación de los empleados en los cursos internos. Ahora, más colaboradores se sienten motivados para capacitarse y desarrollar sus habilidades, lo que ha generado un impacto positivo en su desempeño laboral.

- Personalización y flexibilidad en el aprendizaje:

El sistema web de cursos internos ha permitido a los administradores elegir entre una amplia gama de cursos para sus empleados según sus necesidades y preferencias. La posibilidad de acceder a los contenidos ha brindado a los empleados la flexibilidad de aprender a su propio ritmo, adaptándose a sus horarios laborales y responsabilidades personales.

- Mejora en la retención de conocimientos:

El enfoque interactivo y práctico de los cursos ha demostrado ser altamente efectivo en la retención de conocimientos. Los empleados pueden aplicar directamente lo que aprenden en su trabajo diario, lo que ha llevado a una mayor eficiencia y calidad en las tareas laborales.

- Reducción de costos y tiempo:

El sistema web ha permitido a la Automotriz Bonn S.A. DE C.V, optimizar el tiempo de capacitación al eliminar la necesidad de asignar a una persona física para llevar a cabo dicha capacitación, lo que ha permitido que los empleados inviertan más tiempo en su desarrollo profesional.

- Seguimiento y evaluación del progreso:

El sistema web ofrece herramientas avanzadas para realizar un seguimiento detallado del progreso de cada empleado en los cursos. Los responsables de capacitación y los gerentes pueden monitorear el rendimiento y la evolución de los colaboradores, lo que facilita la identificación de áreas de mejora y la toma de decisiones basadas en datos.

En resumen, con el Sistema Web de Cursos Internos se espera revolucionar la forma en que la Automotriz Bonn capacita a sus empleados, proporcionando un entorno de aprendizaje más accesible, interactivo y flexible. Los resultados obtenidos inicialmente muestran un aumento en la productividad, la satisfacción del personal y el rendimiento general de la empresa.

CONCLUSIONES

En conclusión, el proyecto de cursos internos para la Automotriz Bonn S.A DE C.V. ha demostrado ser una iniciativa sumamente valiosa y efectiva. A lo largo de este proceso, hemos identificado la necesidad de mejorar las habilidades y conocimientos del personal en el sector automotriz. Gracias a la implementación de programas de capacitación y formación, se ha logrado empoderar a los empleados y fortalecer sus competencias técnicas y profesionales.

Los cursos que se han diseñado para cubrir las áreas más relevantes del campo automotriz, abarcando desde aspectos técnicos hasta habilidades de servicio al cliente. Mediante la participación activa y el compromiso de nuestros colaboradores, que serán publicados en el sistema web laborado durante este periodo de tiempo ha alcanzado un nivel significativo de adquisición de conocimientos, lo que se ha traducido en un aumento de la eficiencia, la calidad de servicio y la satisfacción de nuestros clientes.

Además de los beneficios tangibles, también se ha podido observar un cambio positivo en la motivación y el compromiso del equipo de trabajo.

Sin embargo, este proyecto no debe considerarse como una meta aislada, sino como el inicio de un proceso continuo de mejora y aprendizaje constante. Es fundamental mantener y actualizar regularmente los programas de formación para estar a la vanguardia de los avances tecnológicos y las tendencias de la industria automotriz.

En resumen, los cursos para la Automotriz Bonn han sido un gran paso, generando un impacto positivo tanto en la empresa como en el desarrollo profesional de sus empleados.

LITERATURA CITADA

- Referencia: Castells, M. (2001). *The Internet Galaxy: Reflections on the Internet, Business, and Society*. Oxford University Press.
- Referencia: Van Vliet, H. (2008). *Software Engineering: Principles and Practice* (3rd ed.). Wiley.
- Referencia: Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). *Human-Computer Interaction* (3rd ed.). Pearson Education.
- Referencia: Laudon, K. C., & Laudon, J. P. (2020). *Sistemas de información gerencial*. Pearson.
- Referencia: Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 249-256.
- Referencia: Otwell, T. (2011). *Laravel: Up & Running*. O'Reilly Media.
- Referencia: The PHP Group. (2021). *PHP: Documentation*. Recuperado de <https://www.php.net/docs.php>
- Referencia: MariaDB Foundation. (2021). *Maria DB: Documentation*. Recuperado de <https://mariadb.org/documentation/>
- Referencia: Matthew, N., & Stones, R. (2020). *Linux Server Security: Hack and Defend*. Apress.
- Referencia: Elmasri, R., & Navathe, S. B. (2020). *Fundamentals of Database Systems* (8th ed.). Pearson.
- Referencia: Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd ed.). McGraw-Hill.
- Referencia: Gosselin, B. (2019). *Web Development with MongoDB and Node - Third Edition*. Packt Publishing.