SENATI

SERVICIO NACIONAL DE ADIESTRAMIENTO EN TRABAJO INDUSTRIAL

FORMACIÓN PROFESIONAL DUAL

INFORME MENSUAL DE FORMACIÓN PRÁCTICA EN EMPRESA

CÓDIGO N°

SENATI

DIRECCIÓN ZONAL

_____

FORMACIÓN PROFESIONAL DUAL

CFP/UCP/ESCUELA: _____

ESTUDIANTE: Romario solis rosas

ID: 001514833 BLOQUE: _____

CARRERA: Ingeniería de Software con Inteligencia

INSTRUCTOR: **JHON EDWARD FRANCIA MINAYA**

SEMESTRE: 5º DEL: 03/04/25 AL: 01/05/25

INSTRUCCIONES PARA EL USO DEL INFORME MENSUAL DE FORMACIÓN PRÁCTICA EN EMPRESA

1. PRESENTACIÓN.

El Informe mensual de Formación Práctica en Empresa es un documento de control, en el cual el estudiante, registra diariamente, durante el mes, las tareas y operaciones que ejecuta en su formación práctica Empresa.

2. INSTRUCCIONES PARA EL USO DEL INFORME MENSUAL DE FORMACIÓN PRÁCTICA.

2.1 En el cuadro de rotaciones, el estudiante, registrará el nombre de las áreas o secciones por las cuales rota durante su formación práctica en la Empresa, precisando la fecha de inicio y término.

2.2 Con base al PEA publicado en SINFO, el estudiante selecciona el PEA del semestre que está cursando y transcribe el PEA en el informe de práctica.
El estudiante registrará y controlará su avance, marcando en la columna que corresponda.

2.3 Si el PEA tiene menos operaciones (151) de las indicadas en el presente formato, puede eliminar alguna página.

2.4 En el REGISTRO SEMANAL DE TRABAJOS REALIZADOS, el estudiante anotará diariamente los trabajos que ejecuta en la empresa, indicando el tiempo correspondiente. El día de asistencia a SENATI para las sesiones de tecnología, registrará los contenidos que desarrolla en clase. Al término de cada semana totalizará las horas.
De las tareas realizadas durante el mes, el estudiante **seleccionará la tarea más significativa** y realizará una descripción del proceso de ejecución con esquemas y dibujos correspondientes que aclaren dicho proceso.
Una de las características de la comunicación técnica es que debe contener información relevante y fácil de entender.

2.5 Mensualmente, el estudiante **presentará en físico** el informe de la tarea más significativa al Monitor, quien revisará, anotará las observaciones, las recomendaciones que considere y validará con su firma el respectivo informe.
Se recomienda que el monitor solicite al estudiante que **explique o fundamente el informe** que ha elaborado.

2.6 El informe validado por el monitor será presentado al instructor correspondiente (mediante carga en el LMS Blackboard), quien revisará y calificará el Informe mensual de Formación Práctica en Empresa haciendo las observaciones y recomendaciones que considere convenientes, en los aspectos relacionados a la elaboración de un Informe Técnico.

SENATI

CUADRO DE ROTACIONES			
ÁREA / SECCIÓN / EMPRESA	PERÍODO		SEMANAS
	DESDE	HASTA	

SENATI

PLAN ESPECÍFICO DE APRENDIZAJE (PEA)
CONTROL DE AVANCE

Llenar según avance

Nº	OPERACIONES/TAREAS	OPERACIONES EJECUTADAS*				OPERACIONES POR EJECUTAR	OPERACIONES PARA SEMINARIO
		1	2	3	4		
01	Conoce y aplica prácticas de desarrollo de software colaborativo		X				
02	Conoce GitHub para desarrollo colaborativo			X			
03	Conoce las principales herramientas para el control de versiones.			x			
04	Crea repositorios, ramas y usa el control de versiones.		X				
05	Usa entorno de ejecución backend con JavaScrip		x				
06	Instala y configura entorno de desarrollo con Node JS.			X			
07	Crea programas de operaciones CRUD con Node JS y MySQL			x			
08	Usa tecnología frontend con JavaScript		X				
09	Define conceptos de web SPA y configura el entorno para React		x				
10	Crea proyecto con React.			X			
11	Usa componentes, Routing y Navegación en React.		X				
12	Eventos y formularios con React.		X				
13	Diseña y crea servicios API RESTfu			X			
14	Define los fundamentos de REST y SOAP		X				
15	Crea API REST con Node JS.		X				
16	Define los fundamentos de contenedores y Docker.			X			
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							

SENATI

REGISTRO SEMANAL DE TRABAJOS EFECTUADOS

DÍA	TRABAJOS EFECTUADOS	HORAS
LUNES D/M/A	Conoce GitHub para desarrollo colaborativo	5 horas
MARTES D/M/A		
MIÉRCOLES D/M/A		
JUEVES D/M/A	Conoce las principales herramientas para el control de versiones.	7 horas
VIERNES D/M/A		
SÁBADO D/M/A		
	TOTAL	12 horas

DÍA	TRABAJOS EFECTUADOS	HORAS
LUNES D/M/A	Instala y configura entorno de desarrollo con Node JS.	5 horas
MARTES D/M/A		
MIÉRCOLES D/M/A		
JUEVES D/M/A	Crea programas de operaciones CRUD con Node JS y MySQL	7 horas
VIERNES D/M/A		
SÁBADO D/M/A		
	TOTAL	12 horas

DÍA	TRABAJOS EFECTUADOS	HORAS
LUNES D/M/A	Define conceptos de web SPA y configura el entorno para React	5 horas
MARTES D/M/A		
MIÉRCOLES D/M/A		
JUEVES D/M/A	Eventos y formularios con React.	7 horas
VIERNES D/M/A		
SÁBADO D/M/A		
	TOTAL	12 horas

DÍA	TRABAJOS EFECTUADOS	HORAS
LUNES D/M/A	Define los fundamentos de REST y SOAP	5 horas
MARTES D/M/A		
MIÉRCOLES D/M/A		
JUEVES D/M/A	Define los fundamentos de contenedores y Docker.	7 horas
VIERNES D/M/A		
SÁBADO D/M/A		
	TOTAL	12 horas

INFORME MENSUAL

Tarea más significativa del mes: ¿Porqué eligió esta tarea y qué operaciones del PEA cumplió con su ejecución?

sí cumplimos con las tarea y operaciones que nos brindó el instructor con los ejercicios que nos brindó en este mes y tarea que nos brinda más información y diferente tipos de programs y código que podemos utilizar en nuestro proyectos

Descripción del proceso:(Secuencia lógica de la ejecución de la tarea: operaciones, pasos, sub pasos)

En desarrollo de la una tienda de vehículos

1.- Configuración del entorno de desarrollo

Se inició preparando el entorno de trabajo, instalando Node.js y creando dos estructuras principales: el servidor (backend) y la interfaz del usuario (frontend). Se utilizaron herramientas como Visual Studio Code y npm (Node Package Manager) para instalar las dependencias necesarias.

2.- Diseño y desarrollo del backend (Node.js + Express)

Se construyó un servidor con Express, encargado de gestionar las rutas y la lógica para manejar los datos. Se definieron las operaciones principales del modelo CRUD:

- GET para obtener todos los registros.
- POST para añadir nuevos elementos.
- PUT para actualizar información existente.
- DELETE para eliminar datos específicos.

3.- **Construcción de la interfaz (React)**

El frontend se implementó usando React. Se diseñó una interfaz sencilla pero funcional que permitiera al usuario interactuar con los datos fácilmente. Se crearon componentes para:

- Mostrar la lista de registros.
- Formularios para ingresar y modificar datos.
- Botones para eliminar elementos.

El uso de `useState` y `useEffect` facilitó el control del estado de la aplicación y la conexión con el servidor.

4.- **Integración del frontend con el backend**

Se estableció la comunicación entre ambas capas utilizando `fetch()` para realizar peticiones HTTP desde el cliente hacia el servidor. Esto permitió que las acciones del usuario (como añadir, editar o eliminar datos) se reflejaran en la base de datos del backend en tiempo real.

5.- **Pruebas funcionales**

Se realizaron pruebas continuas para validar cada funcionalidad. Se verificó que los datos se guardaran correctamente, que los formularios respondieran a entradas válidas y que los errores fueran gestionados adecuadamente.

6.- **Optimización y organización del código**

Antes de finalizar el proyecto, se revisó el código para asegurar su legibilidad, se modularizaron los componentes, y se eliminaron líneas innecesarias. Esta etapa fue clave para facilitar el mantenimiento futuro.

7.- **Consideraciones para la ampliación**

Aunque la versión actual cumple con los requisitos básicos de un CRUD, se dejó la estructura preparada para incorporar funciones adicionales como autenticación de usuarios, filtros por categoría, historial de actividad o paneles de administración.

Máquinas, equipos, herramientas y materiales (Listar lo utilizado especificando características, medidas, etc)

Laptop personal, PDF, computadora del instituto, git, visual Studio Code, Node.js, sql, Bootstrap

Seguridad e higiene industrial/ambiental (ATS, Charla de cinco minutos: SST/SGA)

Si es importante la charla de los cinco minutos para tener un mejor desarrollo de nuestras actividades y prevenir un accidente en nuestro entorno o practicas

Resultados de la ejecución de la tarea/Recomendaciones (¿Se logró el objetivo que motivó la ejecución de la tarea? Qué recomendaciones sugiere para garantizar la operatividad del bien o servicio realizado

Sí, se logró el objetivo. Se desarrolló e implementó correctamente una aplicación web CRUD utilizando Node.js, la cual permite realizar operaciones de creación, lectura, actualización y eliminación sobre un conjunto de datos. La estructura del proyecto es coherente y cumple con los principios básicos de diseño modular y uso de dependencias mediante `package.json`

HACER ESQUEMA, DIBUJO O DIAGRAMA

```
const bodyParser = require('body-parser')
const path = require('path')

//Acceso a rutas
const rutaVehiculo = require('./routes/vehiculo')
//const rutaMarca = require('./routes/marca')

//Iniciar la App
const app = express();
const PORT = process.env.PORT || 3000

//Configurar "middleware" => "capa de comunicación"
app.use(bodyParser.urlencoded({extended: true}))
app.use(bodyParser.json())
app.use(express.static(path.join(__dirname, 'public')))

//Motor de plantillas
app.set('view engine', 'ejs')
app.set('views', path.join(__dirname, 'views'))

//Configuración rutas
app.use('/', rutaVehiculo)          //Principal
//app.use('/api/marcas', rutaMarca)   //Suministrar datos

//Servidor Web
app.listen(PORT, () => {
    console.log(`Servidor iniciado en http://localhost:3000`)
```

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vehículos</title>

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-SgOJa3DmI

</head>
<body>

  <header>
    <nav class="navbar navbar-expand-lg bg-body-tertiary">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">NodeJS</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link" href="/create">Registro vehículo</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/">Lista vehículo</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
  </header>

  <main class="container">
```

```ejs
<div class="table-responsive">
  <table class="table table-sm">
    <thead>
      <tr>
        <th>ID</th>
        <th>Marca</th>
        <th>Modelo</th>
        <th>Color</th>
        <th>Combustible</th>
        <th>Año</th>
        <th>Condición</th>
        <th>Operaciones</th>
      </tr>
    </thead>
    <tbody>
      <% vehiculos.forEach(vehiculo => { %>
      <tr>
        <td><%= vehiculo.idvehiculo %></td>
        <td><%= vehiculo.marca %></td>
        <td><%= vehiculo.modelo %></td>
        <td><%= vehiculo.color %></td>
        <td><%= vehiculo.combustible %></td>
        <td><%= vehiculo.afabricacion %></td>
        <td><%= vehiculo.condicion %></td>
        <td>
          <a href="/edit/<%= vehiculo.idvehiculo %>" class="btn btn-sm btn-info">Editar</a>
          <a href="/delete/<%= vehiculo.idvehiculo %>" class="btn btn-sm btn-danger delete">Eliminar</a>
        </td>
      </tr>
      <% }) %>
    </tbody>
  </table>
</div>
```

```js
const express = require('express') //Framework
const router = express.Router() //Rutas
const db = require('../config/database') //Acceso BD

router.get('/', async (req, res) => {
  try{
    const query = `
    SELECT
      V.idvehiculo,
        M.marca,
        V.modelo,
        V.color,
        V.combustible,
        V.afabricacion,
        V.condicion
      FROM vehiculos V
        INNER JOIN marcas M ON V.idmarca = M.idmarca
    `
    const [vehiculos] = await db.query(query)
    res.render('index', {vehiculos})
  }catch(error){
    console.error(error)
  }
});

router.get('/create', async(req, res) => {
  try{
    const [datos] = await db.query("SELECT * FROM marcas")
    res.render('create', { marcas: datos })
  }
  catch(error){
    console.error(error)
  }
})
```

```
  try{
    const [datos] = await db.query("SELECT * FROM marcas")
    const [registro] = await db.query("SELECT * FROM vehiculos WHERE idvehiculo = ?", [req.params.id])

    if (registro.length > 0)
     res.render('edit', { marcas: datos, vehiculo: registro[0] })
    else
     res.redirect('/')
  }
  catch(error){
    console.error(error)
  }
})

router.post('/create', async(req, res) => {
  try{
    //Obtener los datos
    const {marcas, modelo, color, combustible, afabricacion, condicion} = req.body
    //Guardar registro
    await db.query(`INSERT INTO vehiculos (idmarca, modelo, color, combustible, afabricacion, condicion) VALUES (?,?,?,?,?,?)`,
      [marcas, modelo, color, combustible, afabricacion, condicion])
    res.redirect('/')
  }catch(error){
    console.error(error)
  }
})
router.post('/edit/:id', async(req, res) => {
  try{
    //Obtener los datos
    const {marcas, modelo, color, combustible, afabricacion, condicion} = req.body
    //Guardar registro
    await db.query("UPDATE vehiculos SET idmarca = ?, modelo=?, color=?, combustible=?, afabricacion=?, condicion=? WHERE idvehiculo=? ",
    [marcas, modelo, color, combustible, afabricacion, condicion, req.params.id])

    res.redirect('/')
  }catch(error){
    console.error(error)
  }
})

//eliminarcion
router.get('/delete/:id', async(req, res) => {
  try{
    //datos que ingresan por el <from></from> res.body.objeto
    //datos que ingresan por GET/URL req.params.atributos
   const [resultado] = await db.query("DELETE FROM vehiculos WHERE idvehiculo = ?", [req.params.id]);
   //res.send(resultado)
   res.redirect('/');
  }catch(error){
    console.error(error);
  };
});

module.exports = router
```

SENATI

```
const mysql = require('mysql2/promise')

//Crear pool de acceso
const pool = mysql.createPool({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'tallerchincha'
})

//Verificar la conexión
async function testConnection(){
  try{
    const connection = await pool.getConnection()
    console.log("Conexión MySQL exitosa")
    connection.release() //liberar
  }catch(error){
    console.error("Error: ", error)
  }
}

testConnection();
module.exports = pool;
```

```
<%- include('partials/header') %>

<form action="/edit/<%= vehiculo.idvehiculo %>" id="formulario-registro" autocomplete="off" method="post">
  <div class="card mt-2">
    <div class="card-header"><strong>Actualicaciones</strong></div>

    <div class="card-body">

      <div class="row g-2">
        <div class="col-md-6 mb-2">      You, last week • Proyecto CRUD Ver 2 parte 2 …
          <div class="form-floating">
            <select name="marcas" id="marcas" class="form-select" required>
              <option value="" selected disabled>Seleccione</option>
              <!-- Las marcas se cargan desde la BD -->
              <% marcas.forEach(element => { %>
                <option value="<%= element.idmarca %>" <%= element.idmarca == vehiculo.idmarca ? 'selected':'' %> ><%= element.marca %></option>
              <% }) %>
            </select>
            <label for="form-label">Marcas</label>
          </div>
        </div>
        <div class="col-md-6 mb-2">
          <div class="form-floating">
            <input type="text" id="modelo" value="<%= vehiculo.modelo %>" name="modelo" class="form-control" placeholder="Modelo" required>
            <label for="form-label">Modelo</label>
          </div>
        </div>
      </div>
```

```
    <div class="row g-2">
      <div class="col-md-4 mb-2">
        <div class="form-floating">
          <input type="text" id="color" value="<%= vehiculo.color %>" name="color" class="form-control" placeholder="Color" required>
          <label for="form-label">Color</label>
        </div>
      </div>
      <div class="col-md-4 mb-2">
        <div class="form-floating">
          <select name="combustible" id="combustible" class="form-select" required>
            <option value="" >Seleccione</option>
            <option value="Diesel"   <%= vehiculo.combustible == 'Diesel' ? 'selected':'' %> >Diesel</option>
            <option value="Gasolina" <%= vehiculo.combustible == 'Gasolina' ? 'selected':'' %> >Gasolina</option>
            <option value="GLP"      <%= vehiculo.combustible == 'GLP' ? 'selected':'' %> >GLP</option>
            <option value="GNV"      <%= vehiculo.combustible == 'GNV' ? 'selected':'' %> >GNV</option>
            <option value="Híbrido"  <%= vehiculo.combustible == 'Híbrido' ? 'selected':'' %> >Híbrido</option>
          </select>
          <label for="form-label">Combustible</label>
        </div>
      </div>
      <div class="col-md-4 mb-2">
        <div class="form-floating">
          <input type="text" minlength="4"  value="<%= vehiculo.fabricación %>" maxlength="4" pattern="[0-9]+" id="afabricacion" name="afabricacion"
          <label for="form-label">Año de fabricación</label>
        </div>
      </div>
    </div>
```

```
      <div class="row">
        <div class="col-md-12">
          <div class="form-floating">
            <select name="condicion" id="condicion" class="form-select" required>
              <option value="Nuevo" <%= vehiculo.Condición  == 'Nuevo' ? 'selected':''%> >Nuevo</option>
              <option value="Seminuevo" <%= vehiculo.Condición  == 'Seminuevo' ? 'selected':''%> >Seminuevo</option>
            </select>
            <label for="form-label">Condición</label>
          </div>
        </div>
      </div>
    </div>

    <div class="card-footer">
      <button type="submit" class="btn btn-sm btn-primary">Actualizar</button>
      <a href="/" class="btn btn-sm btn-secondary">Cancelar</a>
    </div>
  </div>
</form>
```

```
<script>
  document.addEventListener("DOMContentLoaded", () => {
    const formulario = document.querySelector("#formulario-registro")

    formulario.addEventListener("submit", (event) => {
      event.preventDefault()

      Swal.fire({
        title: 'Vehículos',
        text: '¿Está seguro de Actualizar?',
        icon: 'question',
        footer: 'SENATI - Ingeniería Software',
        confirmButtonText: 'Aceptar',
        confirmButtonColor: '#2980b9',
        showCancelButton: true,
        cancelButtonText: 'Cancelar',
        cancelButtonColor: '#c0392b'
      }).then((result) => {
        if (result.isConfirmed){
          formulario.submit()
        }
      })

    })
  })
</script>

<%- include('partials/footer') %>
```

EVALUACIÓN DEL INFORME DE TRABAJO MENSUAL	NOTA	

OBSERVACIONES Y RECOMENDACIONES DEL MONITOR DE EMPRESA:

FIRMA DEL ESTUDIANTE	FIRMA DEL MONITOR DE EMPRESA