“ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE SOFTWARE

Aplicaciones Informáticas II

Paralelo 8vo “A”

Víctor Ochoa – 7198

TEMA

Arquitectura

1. Estilo Arquitectónico: Modular con enfoque MVC

El sistema utilizará una arquitectura modular para dividir las funcionalidades principales, con el patrón MVC (Modelo-Vista-Controlador) para una separación lógica entre datos, interfaz de usuario y controladores de lógica de negocio.

2. Patrón de Comunicación: RESTful API

La comunicación entre el frontend y el backend se realizará mediante APIs RESTful, facilitando la escalabilidad y la integración con futuros módulos.

3. Componentes Clave

- **Front-End:** HTML, CSS, JavaScript para diseño y experiencia del usuario.
- **Controladores PHP:** Scripts PHP actuarán como controladores para manejar las solicitudes y enviar respuestas al frontend.
- **Modelo (Base de Datos):** PHP se conectará directamente a las bases de datos usando PDO para manejar las transacciones.
- **Lógica de Negocio**: Encapsulada en funciones o clases PHP

4. Detalles de cada Componente

Front-End

El frontend interactuará directamente con el backend a través de formularios o peticiones AJAX. No hay separación estricta entre frontend y backend; el HTML es generado dinámicamente desde PHP

- **Portal de Empleados:**
  Una interfaz donde los empleados pueden:
  - Ingresar solicitudes de permisos.
  - Consultar el estado de sus permisos.
  - Ver su historial de permisos.
- **Portal de Administradores/Jefes:**
  Diseñado para jefes y responsables de RRHH:
  - Aprobar o rechazar permisos.
  - Consultar reportes de permisos por empleado.
- **Interfaz interactiva:**
  Formatos dinámicos, como tablas actualizables, notificaciones de estado, y validaciones en tiempo real con JavaScript (AJAX).

Backend en PHP

- **Controladores PHP:**

  Cada funcionalidad clave tendrá su propio script PHP o clase. Por ejemplo:

  - empleados.php → Gestión de empleados.
  - permisos.php → Gestión de permisos.
  - auth.php → Autenticación y autorización.

- **Flujo de Trabajo**
    - **Entrada**: El frontend envía datos mediante formularios o peticiones AJAX (POST/GET).
    - **Lógica de negocios**: PHP valida los datos, aplica reglas de negocio y consulta la base de datos si es necesario.
    - **Salida**: PHP devuelve HTML generado dinámicamente o una respuesta JSON para el frontend (en el caso de AJAX).

Bases de Datos

El sistema manejará una conexión directa con las bases de datos a través de PDO. Las bases de datos se estructurarán de la siguiente manera

1. **Base de Datos de Empleados:**
    - Contendrá información básica de los empleados (nombres, roles, áreas asignadas).
    - Relación jerárquica (empleados y jefes).
2. **Base de Datos de Permisos:**
    - Registro de solicitudes de permisos:
    - ID del permiso.
    - Empleado solicitante.
    - Fecha de inicio y fin del permiso.
    - Estado (pendiente, aprobado, rechazado).
    - Justificación.
3. **Base de Datos de Usuarios:**
    - Credenciales y roles de usuario para gestionar accesos.
4. **Base de Datos de Reportes (opcional):**
    - Almacenará datos consolidados para generar informes.

Seguridad

1. **Autenticación**:
    - Gestión de sesiones con $_SESSION para manejar usuarios autenticados.
2. **Validación de Datos:**
    - Validación del lado del servidor para evitar inyecciones SQL y XSS.
3. **Roles y Permisos:**
    - Control de acceso basado en roles (empleado, administrador).

5. Diagrama de Componentes

1. **Cliente (Frontend):**
    - HTML, CSS, JavaScript (AJAX).
    - Formularios para envío de solicitudes.
    - Dashboard con reportes dinámicos (opcional).
2. **Servidor (PHP):**
    - Scripts PHP divididos por módulos funcionales:
        - empleados.php → Gestión de empleados.
        - permisos.php → Gestión de permisos.
        - auth.php → Autenticación y roles.
    - Respuestas dinámicas en HTML o JSON.

3. **Base de Datos (MySQL/PostgreSQL):**
    - Tablas para empleados, permisos y usuarios.

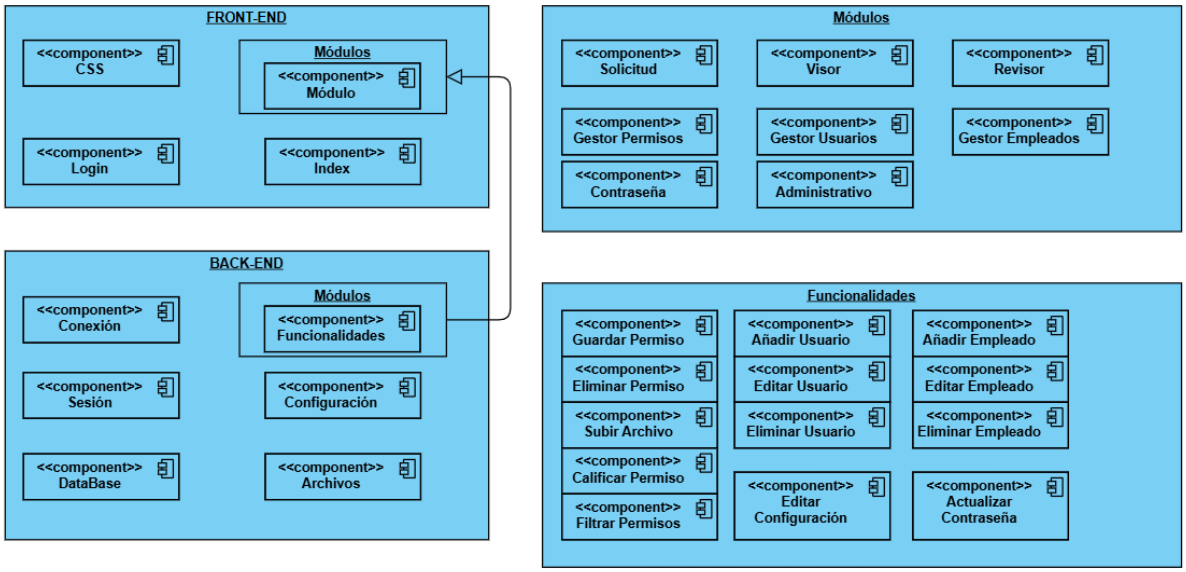
6. Ventajas de la Arquitectura

1. **Simplicidad:**
    - Todo el sistema está centralizado, lo que facilita el despliegue y la configuración.
2. **Modularidad Interna:**
    - Aunque es monolítico, los scripts PHP están organizados en módulos funcionales.
3. **Eficiencia:**
    - Menor latencia al no depender de sistemas distribuidos.

7. Desventajas de la Arquitectura

1. **Escalabilidad:**
    - Es menos flexible para manejar un gran volumen de usuarios o futuras expansiones.
2. **Mantención:**
    - Si el sistema crece demasiado, puede volverse difícil de mantener sin una separación más clara entre frontend y backend.

Diagrama de Componentes



Base de Datos