reporte

Sistema de Control de Asistencia

Este módulo implementa un sistema de reportes estadísticos para el control de asistencia docente utilizando Streamlit, Plotly y FPDF. Permite generar visualizaciones y reportes PDF detallados sobre el cumplimiento de asistencia por profesores, materias y carreras.

Funcionalidades Principales

- Generación de estadísticas de cumplimiento por carrera, profesor y materia
- Visualización de datos mediante gráficas interactivas
- Exportación de reportes detallados en formato PDF
- Interfaz web interactiva con Streamlit

Dependencias

- streamlit
- pandas
- plotly
- fpdf2
- conexion.db

Funciones de Consulta

▶ View Source

def obtener_estadisticas_carrera(): ▶ View Source

Obtiene estadísticas detalladas de asistencia por carrera desde la base de datos.

Devuelve

lista Lista de diccionarios con las siguientes claves: - carrera (str): Nombre de la carrera - total_clases (int): Total de clases programadas - clases_impartidas (int): Total de clases efectivamente impartidas - porcentaje_cumplimiento (float): Porcentaje de cumplimiento

def obtener_estadisticas_profesor(): ▶ View Source

Obtiene estadísticas detalladas de asistencia por profesor desde la base de datos.

Devuelve

lista Lista de diccionarios con las siguientes claves: - profesor (str): Nombre completo del profesor - total_clases (int): Total de clases programadas - clases_impartidas (int): Total de clases efectivamente impartidas - porcentaje_cumplimiento (float): Porcentaje de cumplimiento - total_materias (int): Total de materias asignadas

def obtener_estadisticas_materia(): ▶ View Source

Obtiene estadísticas detalladas de asistencia por materia desde la base de datos.

Devuelve

lista Lista de diccionarios con las siguientes claves: - materia (str): Nombre de la materia - carrera (str): Nombre de la carrera a la que pertenece - total_profesores (int): Número de profesores que imparten la materia - total_clases (int): Total de clases programadas - clases_impartidas (int): Total de clases efectivamente impartidas - porcentaje_cumplimiento (float): Porcentaje de cumplimiento

def crear_grafica_carreras(df): ▶ View Source

Crea una gráfica de barras horizontales para visualizar estadísticas por carrera.

Parámetros

df : pandas.DataFrame DataFrame con las estadísticas de carreras

Devuelve

plotly.graph_objects.Figure Gráfica de barras horizontales con el porcentaje de cumplimiento por carrera

def crear_grafica_profesores(df): ▶ View Source

Crea una gráfica de barras horizontales para visualizar estadísticas por profesor.

Parámetros

df : pandas.DataFrame DataFrame con las estadísticas de profesores

Devuelve

plotly.graph_objects.Figure Gráfica de barras horizontales con el porcentaje de cumplimiento por profesor

def crear_grafica_materias(df): ▶ View Source

Crea una gráfica de barras horizontales para visualizar estadísticas por materia.

Parámetros

df : pandas.DataFrame DataFrame con las estadísticas de materias

Devuelve

plotly.graph_objects.Figure Gráfica de barras horizontales con el porcentaje de cumplimiento por materia

def generar_reporte_pdf(df, tipo_reporte): ▶ View Source

Genera un reporte PDF mejorado con estadísticas detalladas, gráficos y formato profesional

Parámetros

df : pandas.DataFrame DataFrame con los datos a incluir en el reporte tipo_reporte : str Tipo de reporte a generar ('Profesores', 'Materias' o 'Carreras')

Devuelve

str Ruta temporal del archivo PDF generado

Notas

El reporte incluye:

- Encabezado y pie de página personalizados
- Resumen general de estadísticas
- Tabla detallada de datos
- Notas y observaciones
- Formato profesional con colores alternados en las tablas

def generar_reportes(): ▶ View Source

Función principal para generar y mostrar reportes

Crea una interfaz web con tres pestañas:

1. Reporte por Profesor: Estadísticas y métricas de asistencia docente
2. Reporte por Materia: Análisis de cumplimiento por asignatura
3. Reporte por Carrera: Visión general del cumplimiento por programa académico

Cada pestaña incluye:

- Gráfica interactiva de cumplimiento
- Métricas clave en tiempo real
- Opción para exportar reporte PDF detallado

conexion

▶ View Source

class DatabaseConnection: ▶ View Source

Clase Singleton para manejar conexiones a PostgreSQL. Implementa el patrón Singleton para asegurar una única instancia de conexión.

DatabaseConnection() ▶ View Source

Constructor de la clase.

connection

def connect(self) -> Optional[psycopg2.extensions.connection]: ▶ View Source

Establece la conexión con la base de datos PostgreSQL.

Returns: Optional[psycopg2.extensions.connection]: Conexión a la base de datos o None si hay error

def get_cursor(self, dictionary: bool = **True**) -> Optional[psycopg2.extensions.cursor]:
▶ View Source

Obtiene un cursor de la base de datos.

Args: dictionary (bool): Si True, retorna resultados como diccionarios

Returns: Optional[psycopg2.extensions.cursor]: Cursor o None si hay error

def execute_query(
    self,
    query: str,
    params: Union[tuple, List, Dict, NoneType] = **None**,
    commit: bool = **False**
) -> Optional[List]: ▶ View Source

Ejecuta una consulta SQL y opcionalmente hace commit.

Args: query (str): Consulta SQL a ejecutar params (Optional[Union[tuple, List, Dict]]): Parámetros para la consulta commit (bool): Si True, realiza commit después de la consulta

Returns: Optional[List]: Resultados de la consulta o None si hay error

```
def close(self) -> None:
```

▶ View Source

Cierra la conexión a la base de datos.

db = <DatabaseConnection object>

vadmin

▶ View Source

def estilo_admin():

▶ View Source

Aplica estilos CSS modernos y profesionales al panel de administración, con énfasis en la legibilidad del texto. Esta función debe ser llamada al inicio de la aplicación.

Esta configura la apariencia visual de toda la interfaz administrativa, incluyendo colores, tipografía, espaciado y efectos visuales. Define variables CSS globales y estilos para todos los componentes de la interfaz.

def update_database():

▶ View Source

Actualiza el estado de la base de datos en session_state y fuerza una recarga de la página.

Esta función se utiliza después de realizar operaciones de modificación en la base de datos para asegurar que los cambios se reflejen inmediatamente en la interfaz.

def execute_db_operation(query, params, operation_name):

▶ View Source

Ejecuta una operación en la base de datos con manejo de errores mejorado.

Parámetros: query (str): Consulta SQL a ejecutar params (tuple): Parámetros para la consulta operation_name (str): Nombre de la operación para mensajes de error

Devuelve: bool: True si la operación fue exitosa, False en caso contrario

def mostrar_campos_comunes():

▶ View Source

Muestra y recopila los campos comunes para todos los tipos de usuarios.

Devuelve: dict: Diccionario con los valores de los campos comunes: - usuario: str - contrasenia: str - nombre: str - apellidos: str - fecha_nacimiento: date - direccion: str - telefono: str

def grupos():

▶ View Source

Retorna una lista de grupos predefinidos disponibles.

Devuelve: list: Lista de cadenas con los grupos en formato [1-9][A-D,G]

def mostrar_campos_jefe_grupo(): ▶ View Source

Muestra y recopila los campos específicos para jefes de grupo.

Intenta obtener los grupos desde la base de datos y si falla, utiliza una lista predefinida.

Devuelve: str: Grupo seleccionado para asignar al jefe de grupo

def alta_usuario(): ▶ View Source

Gestiona el proceso de registro de nuevos usuarios en el sistema.

Funcionalidades: - Permite crear usuarios de tipo: * Profesor * Administrador * Jefe de grupo

```
- Realiza validaciones de:
  * Campos obligatorios
  * Usuario existente
  * Grupo ya asignado (para jefes de grupo)

- Inserta los datos en las tablas correspondientes según el tipo de usuario
```

Nota: No requiere parámetros ya que obtiene los datos mediante la interfaz de Streamlit

def baja_usuario(): ▶ View Source

Gestiona el proceso de dar de baja usuarios del sistema.

Funcionalidades: - Muestra una lista de usuarios activos - Permite seleccionar usuario a dar de baja - Requiere especificar motivo de la baja - Actualiza el estatus del usuario a 'inactivo'

Nota: No requiere parámetros ya que obtiene los datos mediante la interfaz de Streamlit

def modificar_usuario(): ▶ View Source

Gestiona el proceso de modificación de datos de usuarios existentes.

Funcionalidades: - Permite modificar: * Datos personales (nombre, apellidos, teléfono, dirección) * Contraseña (opcional) * Grupos asignados (solo para jefes de grupo)

```
- Actualiza los datos en las tablas correspondientes según el tipo de usuario
```

Nota: No requiere parámetros ya que obtiene los datos mediante la interfaz de Streamlit

def alta_asignatura(): ▶ View Source

Gestiona el proceso de registro de nuevas asignaturas.

Funcionalidades: - Permite ingresar nombre de la asignatura - Permite seleccionar carrera asociada - Registra la asignatura en la base de datos

Nota: No requiere parámetros ya que obtiene los datos mediante la interfaz de Streamlit

def asignar_materia(): ▶ View Source

Gestiona el proceso de asignación de materias a profesores.

Funcionalidades: - Permite: * Seleccionar profesor activo * Seleccionar materia disponible * Especificar grupo y ciclo escolar

---

```
- Realiza validaciones de:
  * Campos completos
  * Asignación existente

- Registra la asignación en la tabla profesor_materia
```

Nota: No requiere parámetros ya que obtiene los datos mediante la interfaz de Streamlit

def admin(): ▶ View Source

Función principal que gestiona el panel de administración.

Proporciona acceso a todas las operaciones administrativas:

- Alta de Usuario
- Baja de Usuario
- Modificar Usuario
- Alta de Asignatura
- Asignación de Materias
- Reportes

Aplica los estilos del panel y gestiona la navegación entre operaciones.

login

Sistema de Autenticación y Control de Acceso.

Este módulo implementa el sistema de autenticación y control de acceso para la aplicación de control de asistencia. Maneja la interfaz de login, validación de usuarios y gestión de sesiones utilizando Streamlit.

Dependencias: - streamlit - hashlib - datetime - conexion (módulo local para manejo de base de datos) - vadmin (módulo local para interfaz de administrador) - alumno (módulo local para interfaz de jefe de grupo)

▶ View Source

def estilo():

▶ View Source

Aplica los estilos CSS personalizados a la interfaz de login.

Descripción: Configura la apariencia visual de la interfaz de login aplicando estilos CSS personalizados a través de Streamlit.

Elementos que personaliza: - Contenedor principal y fondo - Formulario de login - Campos de entrada - Botones - Títulos y textos - Efectos visuales y animaciones

Nota: Los estilos se aplican usando st.markdown con unsafe_allow_html=True

def init_session_state():

▶ View Source

Inicializa las variables de estado de la sesión.

Descripción: Configura las variables iniciales en st.session_state para controlar el estado de la sesión del usuario.

Variables que inicializa: - logged_in: Estado de la sesión del usuario - user_data: Datos del usuario autenticado

def mostrar_interface_jefe_grupo(user_data):

▶ View Source

Muestra la interfaz específica para jefes de grupo.

Parámetros: user_data (dict): Diccionario con la información del jefe de grupo. Debe contener: - username: Identificador del usuario - role: Rol del usuario - status: Estado del usuario - nombre: Nombre del usuario - apellidos: Apellidos del usuario

def validate_user(username: str, password: str): ▶ View Source

Valida las credenciales del usuario contra la base de datos.

Parámetros: username (str): Nombre de usuario a validar password (str): Contraseña del usuario

Retorna: tuple: (bool, dict) - bool: True si la validación es exitosa, False en caso contrario - dict: Datos del usuario si la validación es exitosa, None en caso contrario Contiene: username, role, status, nombre, apellidos

Excepciones: Exception: Si hay errores en la consulta a la base de datos

Nota: La contraseña se procesa con hash SHA-256 antes de la validación

def show_login_form(): ▶ View Source

def show_user_interface(): ▶ View Source

Muestra la interfaz correspondiente al rol del usuario autenticado.

Descripción: Determina y muestra la interfaz apropiada basada en el rol del usuario: - Administrador: Muestra la interfaz de administración - Jefe de grupo: Muestra la interfaz de jefe de grupo

Funcionalidades: - Muestra información básica del usuario - Proporciona botón de cierre de sesión - Redirige a la interfaz específica según el rol

Nota: Al cerrar sesión, limpia el estado y cierra la conexión a la base de datos

def login(): ▶ View Source

Función principal que maneja el flujo de autenticación.

Descripción: Controla el flujo principal de la autenticación y navegación del sistema. Inicializa el estado de la sesión y muestra la interfaz apropiada según el estado de autenticación del usuario.

Flujo de ejecución: 1. Muestra el título de la aplicación 2. Inicializa el estado de la sesión 3. Muestra el formulario de login o la interfaz de usuario según corresponda

alumno

Sistema de Registro de Asistencia de Profesores.

Este módulo implementa un sistema de registro y gestión de asistencias de profesores utilizando Streamlit como framework de interfaz y PostgreSQL como base de datos. El sistema está diseñado para ser utilizado por jefes de grupo para registrar y monitorear la asistencia de los profesores asignados a sus grupos.

Dependencies: - streamlit - pytz - datetime - conexion (módulo local para manejo de base de datos)

▶ View Source

def estilo_jefe(): ▶ View Source

Función para aplicar el estilo CSS en la interfaz de registro de asistencia para el jefe de grupo.

Los estilos incluyen: - Configuración de fuentes y colores base - Estilos para formularios y campos de entrada - Diseño de botones y estados de hover - Estilos para mensajes de éxito y error - Formato de textos informativos

Nota: Los estilos se aplican usando st.markdown con unsafe_allow_html=True

def jefe_grupo(user_data: Dict): ▶ View Source

def obtener_grupo_jefe(usuario: str) -> Optional[str]: ▶ View Source

Consulta y retorna el grupo asignado al jefe de grupo.

Parámetros: usuario (str): Identificador único del jefe de grupo.

Retorna: Optional[str]: Identificador del grupo asignado o None si no se encuentra.

Excepciones: Exception: Si hay errores en la consulta a la base de datos.

def obtener_materias_grupo(grupo: str) -> Optional[List[Dict]]: ▶ View Source

Obtiene las materias y profesores asignados a un grupo específico.

Parámetros: grupo (str): Identificador del grupo a consultar.

Retorna: Optional[List[Dict]]: Lista de diccionarios con la información de materias y profesores. Cada diccionario contiene: - profesor_usuario (str): ID del profesor - profesor_nombre (str): Nombre del

profesor - profesor_apellidos (str): Apellidos del profesor - materia_id (int): ID de la materia - materia_nombre (str): Nombre de la materia - dias_semana (str): Días programados - hora_inicio (time): Hora de inicio de clase - hora_fin (time): Hora de fin de clase - aula (str): Aula asignada

Excepciones: Exception: Si hay errores en la consulta a la base de datos.

```
def validar_dia_clase(dia_actual: str, dias_programados: List[str]) -> bool:
```
▶ View Source

Valida si un día específico corresponde a los días programados para la clase.

Parámetros: dia_actual (str): Día de la semana en inglés (ej: 'monday', 'tuesday'). dias_programados (List[str]): Lista de días permitidos en español.

Retorna: bool: True si el día es válido, False en caso contrario.

Ejemplo:

```
validar_dia_clase('monday', ['lunes', 'miércoles']) True
```

```
def verificar_registro_existente(
    fecha: datetime.datetime,
    profesor: str,
    materia: int,
    grupo: str
) -> bool:
```
▶ View Source

Verifica si ya existe un registro de asistencia para una combinación específica.

Parámetros: fecha (datetime): Fecha del registro a verificar. profesor (str): ID del profesor. materia (int): ID de la materia. grupo (str): ID del grupo.

Retorna: bool: True si existe un registro, False en caso contrario.

Excepciones: Exception: Si hay errores en la consulta a la base de datos.

```
def registrar_asistencia(
    fecha: datetime.datetime,
    hora_inicio: datetime.time,
    hora_fin: datetime.time,
    profesor_usuario: str,
    materia_id: int,
    grupo: str,
    estatus: str,
```

```
        observaciones: str
) -> bool:
```

▶ View Source

Registra la asistencia de un profesor en la base de datos.

Parámetros: fecha (datetime): Fecha de la clase. hora_inicio (time): Hora de inicio de la clase. hora_fin (time): Hora de finalización de la clase. profesor_usuario (str): ID del profesor. materia_id (int): ID de la materia. grupo (str): ID del grupo. estatus (str): Estado de la clase ('impartida', 'no_impartida', 'retardo'). observaciones (str): Comentarios adicionales sobre la clase.

Retorna: bool: True si el registro fue exitoso, False en caso contrario.

Excepciones: Exception: Si hay errores en la inserción en la base de datos.

Nota: El estatus solo puede ser uno de los siguientes valores: - 'impartida': Clase impartida normalmente - 'no_impartida': Clase no impartida - 'retardo': Clase impartida con retardo