Documentación de la Aplicación Web de Gestión de FarmaFac

Introducción

Esta aplicación web está diseñada para gestionar compras, manejar inventarios de productos y generar facturas. Utiliza Flask como framework web y cx_Oracle para interactuar con una base de datos Oracle.

Requisitos

- Python 3.x
- Flask
- cx_Oracle
- num2words
- weasyprint
- Oracle Database

Configuración Inicial

- Instalar las dependencias necesarias:

  pip install Flask cx_Oracle num2words weasyprint

- Configurar la conexión a la base de datos Oracle en la función get_db.

Rutas de la Aplicación

Función get_db()

Propósito: Esta función se encarga de obtener una conexión a la base de datos. Si la conexión no existe en el contexto global (`g`), intenta obtener las credenciales de inicio de sesión de la sesión del usuario y establece la conexión. Luego, devuelve la conexión y un cursor para ejecutar consultas.

Parámetros:
- Ninguno.

Operaciones:

- Verifica si la conexión ya existe en el contexto global (`g`). Si no existe, procede a obtener las credenciales de inicio de sesión de la sesión del usuario.

- Intenta establecer una conexión a la base de datos utilizando las credenciales proporcionadas.
- Crea un cursor asociado con la conexión.
- Devuelve la conexión y el cursor.

Consideraciones:

- Gestión de Sesión: Utiliza la sesión del usuario para almacenar y recuperar las credenciales de inicio de sesión.
- Seguridad de Datos: Asegura que las credenciales de inicio de sesión se transmitan de manera segura y se utilicen para establecer la conexión de manera segura.
- Validación de Entradas: Es importante validar las credenciales de inicio de sesión antes de utilizarlas para evitar posibles ataques de inyección de SQL u otras vulnerabilidades.

Inicialización de la aplicación Flask (app)

Propósito: Esta parte del código inicializa la aplicación Flask y configura la clave secreta para la gestión de sesiones.

Parámetros:
- Ninguno.

Operaciones:
- Importa el módulo `Flask`.
- Inicializa una instancia de la aplicación Flask.
- Configura la clave secreta para la gestión de sesiones.

Consideraciones:

- Gestión de Sesión: La clave secreta es crucial para la seguridad de la gestión de sesiones en Flask. Debe ser lo suficientemente segura para prevenir ataques de secuestro de sesión.

Función index()

Propósito: Esta función maneja las solicitudes GET y POST en la ruta '/login' y autentica a los usuarios en función de su nombre de usuario y contraseña proporcionados.

Parámetros:
- Ninguno.

Operaciones:
- Si la solicitud es POST, verifica las credenciales de inicio de sesión proporcionadas por el usuario.

- Si las credenciales son válidas, establece la sesión del usuario y redirige según el rol del usuario.
- Si la solicitud es GET, renderiza la plantilla 'login.html'.

Consideraciones:

- Autenticación de Usuarios: Se deben verificar cuidadosamente las credenciales de inicio de sesión para evitar posibles brechas de seguridad.
- Seguridad de Contraseñas: Las contraseñas deben almacenarse de forma segura utilizando técnicas de hashing y salting para protegerlas contra ataques de fuerza bruta y de diccionario.

Función admin()

Propósito: Esta función maneja las solicitudes GET y POST en la ruta '/admin' y muestra la página de administrador con la lista de usuarios y los movimientos registrados.

Parámetros:
- Ninguno.

Operaciones:
- Verifica si el usuario está autenticado.
- Si está autenticado, obtiene la lista de usuarios y los movimientos registrados en la base de datos.
- Renderiza la plantilla 'admin.html' con los datos obtenidos.

Consideraciones:

- Seguridad de Acceso: Asegura que solo los usuarios autenticados y autorizados puedan acceder a la página de administrador.
- Protección de Datos: Los datos de los usuarios y los movimientos deben manejarse de manera segura y protegerse contra posibles fugas de información.

Función auditor()

Propósito:
Esta función maneja las solicitudes GET y POST en la ruta '/auditor' y muestra la página del auditor con la lista de artículos y permite modificarlos.

Parámetros:
- Ninguno.

Operaciones:
- Verifica si el usuario está autenticado.
- Si está autenticado, obtiene la lista de artículos de la base de datos.

- Si la solicitud es POST, modifica el artículo según los datos proporcionados.
- Renderiza la plantilla 'auditor.html' con los datos obtenidos.

Consideraciones:
- Seguridad de Acceso: Asegura que solo los usuarios autenticados y autorizados puedan acceder a la página del auditor.
- Protección de Datos: Los datos de los artículos deben manejarse de manera segura y protegerse contra posibles modificaciones no autorizadas.

Función empleado()

Propósito: Esta función maneja las solicitudes en la ruta '/empleado' y muestra información relevante para los empleados, como nóminas y productos disponibles en la tienda.

Parámetros:
- se_filtra: Parámetro opcional que indica si se están aplicando filtros a los productos.

Operaciones:

- Consulta la base de datos para obtener información sobre los productos disponibles en la tienda.
- Obtiene el usuario actual.
- Recupera el ID del usuario.
- Obtiene el ID del empleado asociado al usuario.
- Recupera información sobre el empleado.
- Obtiene información de la nómina del empleado.
- Obtiene la cantidad de elementos en el carrito de compras del cliente.
- Si se aplican filtros a los productos, actualiza la lista de productos.

Consideraciones:

-Personalización de la Interfaz: Utiliza la información del usuario para personalizar la visualización de la página, mostrando diferentes opciones según el tipo de usuario. Por ejemplo, podrías mostrar funcionalidades adicionales específicas para empleados, como la capacidad de gestionar nóminas o ver información interna adicional.

Función nomina_generada()

Propósito:
Genera una nómina en formato PDF para un empleado y proporciona un enlace para descargarla.

Parámetros:
- Datos del formulario POST, incluyendo información relevante para generar la nómina.

Operaciones:

- Recibe los datos del formulario POST.
- Genera el contenido HTML de la nómina utilizando los datos proporcionados.
- Convierte el contenido HTML a formato PDF.
- Envía el PDF generado como una descarga al usuario.
- Establece un encabezado de actualización para redirigir al usuario a la página del empleado después de 5 segundos.

Consideraciones:

- Se debe tener cuidado al generar y enviar archivos PDF, ya que pueden consumir muchos recursos del servidor si son muy grandes o si hay muchas solicitudes simultáneas.

Función buscar_articulo_cliente() y buscar_articulo_empleado()

Propósito: Estas funciones manejan las solicitudes de búsqueda de artículos por parte de clientes y empleados, respectivamente.

Parámetros:
- entrada: Cadena de búsqueda introducida por el usuario.

Operaciones:
- Consulta la base de datos para obtener información sobre los productos disponibles en la tienda.
- Realiza una búsqueda aproximada de los productos que coinciden con la cadena de búsqueda introducida por el usuario.
- Actualiza la lista de productos con los resultados de la búsqueda.

Consideraciones:

- Se utiliza una búsqueda aproximada para encontrar productos similares, lo que puede producir resultados inesperados si la cadena de búsqueda es ambigua o si hay muchos productos similares en la base de datos.

- Personalización de Búsqueda: Aprovecha la información del usuario para personalizar los resultados de búsqueda, mostrando productos relevantes basados en el historial de compras o preferencias del usuario.

Función cliente()

Propósito: Esta función maneja las solicitudes en la ruta '/cliente' y muestra información relevante para los clientes, como facturas y productos disponibles en la tienda.

Parámetros:
- se_filtra: Parámetro opcional que indica si se están aplicando filtros a los productos.

Operaciones:

- Consulta la base de datos para obtener información sobre los productos disponibles en la tienda.
- Obtiene el usuario actual.
- Recupera el ID del usuario.
- Obtiene el ID del cliente asociado al usuario.
- Recupera información sobre las facturas del cliente.
- Obtiene la cantidad de elementos en el carrito de compras del cliente.
- Si se aplican filtros a los productos, actualiza la lista de productos.

Consideraciones:
- Al igual que con la función `empleado()`, esta función realiza múltiples consultas a la base de datos, lo que puede afectar el rendimiento si la base de datos es grande o hay muchas solicitudes simultáneas.

Función eliminar_usuario()

Propósito: Esta función se encarga de eliminar un usuario de la base de datos junto con su correspondiente cuenta de usuario en el sistema.

Parámetros:
- id: El ID del usuario que se desea eliminar.
- username: El nombre de usuario asociado al usuario que se desea eliminar.

Operaciones:

- Elimina el usuario de la tabla de usuarios en la base de datos.
- Elimina la cuenta de usuario asociada al nombre de usuario proporcionado.

Función agregar_usuario()

Propósito: Esta función se encarga de agregar un nuevo usuario a la base de datos junto con su cuenta de usuario en el sistema.

Parámetros:
- Datos del formulario POST para crear un nuevo usuario, incluyendo username, password, email, role, nombre, regimenFiscal, usoFiscal, area, salario, fechaNacimiento, domicilio, cp, telefono, sexo y rfc.

Operaciones:

- Inserta los datos del nuevo usuario en la tabla de usuarios en la base de datos.

- Crea una cuenta de usuario con el nombre de usuario y contraseña proporcionados.

Consideraciones:

- Validación de Entradas: Es esencial validar exhaustivamente los datos ingresados por el usuario para prevenir posibles vulnerabilidades.

- Seguridad de Contraseñas: Se deben aplicar políticas de seguridad de contraseñas para garantizar contraseñas robustas y seguras.

- Protección de Datos Sensibles: Asegurar que la información personal y sensible del usuario se almacene de forma segura y se proteja adecuadamente.

Función registro()

Propósito: Esta función se encarga de manejar las solicitudes POST en la ruta '/registrar_usuario' y renderiza la plantilla 'registrar_usuario.html' con los datos proporcionados.

Parámetros:
- Ninguno (solo recibe datos a través de la solicitud POST).

Operaciones:
- Renderiza la plantilla 'registrar_usuario.html' con los datos proporcionados a través de la solicitud POST.

Consideraciones:
- Seguridad de Acceso: Implementar medidas de seguridad adecuadas para proteger la página de registro contra ataques como CSRF y la inyección de código malicioso.

- Validación de Datos: Realizar una validación exhaustiva de los datos ingresados por el usuario para evitar posibles vulnerabilidades.

Función agregar_movimiento()

Propósito: Esta función se encarga de agregar un nuevo movimiento a la base de datos.

Parámetros:
- Datos del formulario POST para crear un nuevo movimiento, incluyendo date y description.

Operaciones:
1. Inserta los datos del nuevo movimiento en la tabla de movimientos en la base de datos.

Consideraciones:
- Validación de Entradas: Validar los datos ingresados por el usuario para garantizar que sean válidos y seguros antes de agregar el movimiento a la base de datos.

- Protección contra Inyección de SQL: Sanitizar y escapar los datos correctamente para prevenir ataques de inyección de SQL.

Función registrar_movimiento()

Propósito: Esta función se encarga de manejar las solicitudes GET en la ruta '/registrar_movimiento' y renderiza la plantilla 'registrar_movimiento.html'.

Parámetros:

- Ninguno (solo recibe solicitudes GET).

Operaciones:

- Renderiza la plantilla 'registrar_movimiento.html'.

Consideraciones:

- Seguridad de Acceso: Asegurar que solo los usuarios autenticados y autorizados puedan acceder a esta página.
- Implementar medidas de autenticación y autorización adecuadas para proteger la información sensible.

Función modificar_articulo()

Propósito: Esta función se encarga de manejar las solicitudes POST en la ruta '/modificar_articulo' y renderiza la plantilla 'modificar_articulo.html' con los datos proporcionados.

Parámetros:
- Datos del formulario POST para modificar un artículo, incluyendo nombre, cantidad, precio e id.

Operaciones:
- Renderiza la plantilla 'modificar_articulo.html' con los datos proporcionados a través de la solicitud POST.

Consideraciones:
- Validación de Entradas: Validar los datos ingresados por el usuario para garantizar que sean válidos y seguros antes de modificar el artículo.

- Seguridad de Acceso: Asegurar que solo los usuarios autorizados puedan realizar modificaciones en los artículos.

Función logout()

Propósito: Esta función se encarga de manejar las solicitudes GET y POST en la ruta '/logout' y redirige al usuario a la página de inicio de sesión.

Parámetros:
- Ninguno (solo recibe solicitudes GET y POST).

Operaciones:
- Redirige al usuario a la página de inicio de sesión.

Consideraciones:
- Seguridad de Sesión: Asegurarse de que el cierre de sesión se realice de manera segura y que todas las sesiones activas del usuario se invaliden correctamente.

- Implementar medidas para prevenir el secuestro de sesión y otras vulnerabilidades relacionadas con la gestión de sesiones.

Función agregar_a_carrito() y agregar_a_carrito_empleado()

Propósito: Esta función maneja la lógica para agregar un artículo al carrito de compras de un usuario.

Parámetros: No recibe parámetros directamente desde la URL. Sin embargo, espera recibir el ID del producto a agregar a través de los datos del formulario.

Operaciones:

- Obtiene el usuario actual de la base de datos.
- Verifica si el producto ya está en el carrito del usuario.
- Si el producto está en el carrito, incrementa la cantidad.
- Si el producto no está en el carrito, lo agrega con una cantidad de 1.
- Actualiza la base de datos con la nueva información del carrito.

Consideraciones:

-Maneja excepciones relacionadas con la base de datos, como la falta de stock, e imprime un mensaje adecuado en caso de error.

Función ver_carrito()

Propósito: Esta función muestra los artículos en el carrito de compras de un usuario.

Parámetros: Espera recibir datos del formulario que contienen información del usuario.

Operaciones:

- Obtiene el ID y tipo de usuario de la base de datos.
- Consulta los productos en el carrito del usuario.
- Calcula el total de la compra.
- Renderiza la plantilla compras.html con la información del carrito y el total.

Consideraciones:

- Utiliza la información del usuario para personalizar la visualización de la página, mostrando diferentes opciones según el tipo de usuario.

Función agregar_articulo()

Propósito: Agrega un nuevo artículo al inventario de productos.

Parámetros: Recibe los datos del formulario que incluyen el nombre, cantidad y precio del producto.

Operaciones:

- Inserta el nuevo producto en la base de datos.
- Maneja excepciones en caso de violaciones de integridad, como la duplicación de nombres de productos.

Consideraciones:

-Utiliza transacciones para garantizar la consistencia de la base de datos y manejar posibles errores de manera segura.

Función eliminar_articulo()

Propósito: Elimina un artículo del inventario de productos.

Parámetros: Recibe el ID del producto a eliminar a través de los datos del formulario.

Operaciones:

- Elimina el producto correspondiente de la base de datos.
- Maneja excepciones en caso de violaciones de integridad u otros errores.

Consideraciones:

-Utiliza transacciones para garantizar la integridad de la base de datos y manejar errores de manera adecuada.

Función buscar_articulo()

Propósito: Busca un artículo en el inventario de productos.

Parámetros: Recibe una cadena de búsqueda a través de los datos del formulario.

Operaciones:

- Obtiene todos los productos de la base de datos.
- Filtra los productos que coinciden con la cadena de búsqueda utilizando la biblioteca difflib.
- Renderiza la plantilla auditor.html con los productos filtrados.

Consideraciones:

-Utiliza técnicas de coincidencia de cadenas para mejorar la precisión de la búsqueda.

Función factura()

Propósito: Genera una factura para los artículos en el carrito de compras.

Parámetros: Recibe datos del formulario que incluyen los artículos en el carrito, el total de la compra, información del usuario y el tipo de usuario.

Operaciones:

- Renderiza la plantilla factura.html con los datos proporcionados.
- Consideraciones:
- Personaliza la factura según el tipo de usuario, mostrando diferentes opciones y detalles.

Función gracias()

Propósito: Muestra un mensaje de agradecimiento después de completar una transacción.

Parámetros: Recibe el tipo de cliente a través de los datos del formulario.

Operaciones:

- Renderiza la plantilla gracias_empleado.html si el tipo de cliente es "empleado", de lo contrario, renderiza gracias.html.

Función factura_generada()

Propósito: Genera y guarda una factura, además de crear un archivo PDF.

Parámetros: Recibe varios campos relacionados con la factura y los artículos a través de los datos del formulario.

Operaciones:

- Obtiene los datos del usuario y cliente.
- Guarda la factura en la base de datos si es necesario.
- Actualiza las cantidades de productos y limpia el carrito de compras.
- Renderiza la plantilla factura_generada.html y crea un PDF utilizando la biblioteca weasyprint.
- Envía el PDF como un archivo descargable y redirige al usuario a la página de cliente.

Consideraciones:

-Utiliza UUID para generar un identificador único para la factura.

-Utiliza transacciones para garantizar la integridad de la base de datos durante la generación de la factura.

Organización de archivos

```
├── app.py              # Código principal de la aplicación
├── templates/          # Plantillas HTML
│   ├── compras.html
│   ├── auditor.html
│   ├── factura.html
│   ├── gracias.html
│   ├── gracias_empleado.html
│   ├── factura_generada.html
└── static/             # Archivos estáticos (CSS, JS, imágenes)
```

Notas Adicionales

Asegúrate de que la base de datos Oracle esté correctamente configurada y que los esquemas necesarios existan.

Personaliza los nombres de usuario, contraseñas y DSN en la función get_db para adaptarlos a tu entorno.

La generación de PDF utiliza WeasyPrint, que debe estar correctamente instalado y configurado en tu entorno de desarrollo.

Documentation of FarmaFac Management Web Application

Introduction:

This web application is designed to manage purchases, handle product inventories, and generate invoices. It uses Flask as the web framework and cx_Oracle to interact with an Oracle database.

Requirements:

- Python 3.x
- Flask
- cx_Oracle
- num2words
- weasyprint
- Oracle Database

Initial Configuration:

- Install the necessary dependencies:

pip install Flask cx_Oracle num2words weasyprint

- Configure the connection to the Oracle database in the *get_db* function.

Application Routes:

Function get_db()

Purpose: This function is responsible for obtaining a connection to the database. If the connection does not exist in the global context (`g`), it attempts to retrieve the user's login credentials from the session and establishes the connection. Then, it returns the connection and a cursor to execute queries.

Parameters:

- None

Operations:

- Verifies if the connection already exists in the global context (`g`). If it doesn't exist, it proceeds to retrieve the user's login credentials from the session.
- Attempts to establish a connection to the database using the provided credentials.
- Creates a cursor associated with the connection.
- Returns the connection and the cursor.

Considerations:

- Session Management: Uses the user's session to store and retrieve login credentials.
- Data Security: Ensures that login credentials are transmitted securely and used to establish the connection securely.
- Input Validation: It's important to validate login credentials before using them to avoid potential SQL injection attacks or other vulnerabilities.

Initializing the Flask Application (app)

Purpose: This part of the code initializes the Flask application and configures the secret key for session management.

Parameters:

- None

Operations:

- Imports the `Flask` module.
- Initializes an instance of the Flask application.
- Configures the secret key for session management.

Considerations:
- Session Management: The secret key is crucial for session management security in Flask. It must be secure enough to prevent session hijacking attacks.

Function index()

Purpose: This function handles GET and POST requests on the '/login' route and authenticates users based on their provided username and password.

Parameters:

- None

Operations:

- If the request is POST, it verifies the login credentials provided by the user.
- If the credentials are valid, it sets the user's session and redirects based on the user's role.
- If the request is GET, it renders the 'login.html' template.

Considerations:

- User Authentication: Login credentials must be carefully verified to prevent potential security breaches.
- Password Security: Passwords should be securely stored using hashing and salting techniques to protect against brute-force and dictionary attacks.

Function admin()

Purpose: This function handles GET and POST requests on the '/admin' route and displays the admin page with the list of users and registered movements.

Parameters:

- None

Operations:

- Verifies if the user is authenticated.
- If authenticated, it retrieves the list of users and registered movements from the database.
- Renders the 'admin.html' template with the retrieved data.

Considerations:

- Access Security: Ensures that only authenticated and authorized users can access the admin page.
- Data Protection: User data and movements must be handled securely and protected against potential information leaks.

Function auditor()

Purpose: This function handles GET and POST requests on the '/auditor' route and displays the auditor page with the list of items and allows for modifications.

Parameters:

- None

Operations:

- Verifies if the user is authenticated.

- If authenticated, it retrieves the list of items from the database.
- If the request is POST, it modifies the item based on the provided data.
- Renders the 'auditor.html' template with the retrieved data.

Considerations:

- Access Security: Ensures that only authenticated and authorized users can access the auditor page.
- Data Protection: Item data must be handled securely and protected against unauthorized modifications.

Function empleado()

Purpose: This function handles requests on the '/employee' route and displays relevant information for employees, such as payrolls and products available in the store.

Parameters:
- filtrado: Optional parameter indicating if filters are being applied to the products.

Operations:

- Queries the database to retrieve information about products available in the store.
- Retrieves the current user.
- Retrieves the user ID.
- Obtains the ID of the employee associated with the user.
- Retrieves information about the employee.
- Obtains information about the employee's payroll.
- Retrieves the quantity of items in the customer's shopping cart.
- If filters are applied to the products, updates the product list.

Considerations:

- Interface Customization: Uses user information to personalize page display, showing different options based on the user type. For example, you might display additional functionalities specific to employees, such as the ability to manage payrolls or view additional internal information.

Function nomina_generada()

Purpose: Generates a payroll in PDF format for an employee and provides a download link.

Parameters:

- POST form data, including relevant information to generate the payroll.

Operations:

- Receives the POST form data.
- Generates the HTML content of the payroll using the provided data.
- Converts the HTML content to PDF format.
- Sends the generated PDF as a download to the user.
- Sets a refresh header to redirect the user to the employee page after 5 seconds.

Considerations:

- Care should be taken when generating and sending PDF files, as they can consume many server resources if they are very large or if there are many simultaneous requests.

Functions buscar_articulo_cliente() and buscar_articulo_empleado()

Purpose: These functions handle search requests for items by clients and employees, respectively.

Parameters:

- input: Search string entered by the user.

Operations:

- Queries the database to retrieve information about available products in the store.
- Performs an approximate search for products matching the search string entered by the user.
- Updates the list of products with the search results.

Considerations:

- Approximate Search: An approximate search is used to find similar products, which may produce unexpected results if the search string is ambiguous or if there are many similar products in the database.
- Search Customization: Leverages user information to personalize search results, showing relevant products based on user purchase history or preferences.

Function cliente()

Purpose: This function handles requests on the '/client' route and displays relevant information for clients, such as invoices and products available in the store.

Parameters:
- filtered: Optional parameter indicating if filters are being applied to the products.

Operations:

- Queries the database to retrieve information about products available in the store.
- Retrieves the current user.
- Retrieves the user ID.
- Obtains the ID of the client associated with the user.
- Retrieves information about the client's invoices.
- Retrieves the quantity of items in the client's shopping cart.
- If filters are applied to the products, updates the product list.

Considerations:

- Similar to the employee() function, this function performs multiple database queries, which can affect performance if the database is large or if there are many simultaneous requests.

Function eliminar_usuario()

Purpose: This function is responsible for deleting a user from the database along with their corresponding user account in the system.

Parameters:

- id: The ID of the user to be deleted.

- username: The username associated with the user to be deleted.

Operations:

- Deletes the user from the users table in the database.
- Removes the user account associated with the provided username.

Function agregar_usuario()

Purpose: This function is responsible for adding a new user to the database along with their user account in the system.

Parameters:

- POST form data to create a new user, including username, password, email, role, name, fiscalRegime, fiscalUsage, area, salary, dateOfBirth, address, postalCode, phone, gender, and rfc.

Operations:

- Inserts the data of the new user into the users table in the database.

- Creates a user account with the provided username and password.

Considerations:

- Input Validation: Thoroughly validate user-input data to prevent possible vulnerabilities.
- Password Security: Implement password security policies to ensure strong and secure passwords.
- Sensitive Data Protection: Ensure that the user's personal and sensitive information is securely stored and properly protected.

Function registro()

Purpose: This function handles POST requests on the '/register_user' route and renders the 'register_user.html' template with the provided data.

Parameters:
- None (only receives data through the POST request).

Operations:
- Renders the 'register_user.html' template with the provided data from the POST request.

Considerations:
- Access Security: Implement appropriate security measures to protect the registration page against attacks such as CSRF and malicious code injection.

- Data Validation: Perform thorough validation of user-input data to prevent potential vulnerabilities.

Function agregar_movimiento()

Purpose: This function is responsible for adding a new movement to the database.

Parameters:

- POST form data to create a new movement, including date and description.

Operations:

- Inserts the data of the new movement into the movements table in the database.

Considerations:

- Input Validation: Validate the user-input data to ensure it is valid and secure before adding the movement to the database.
- Protection against SQL Injection: Sanitize and escape data correctly to prevent SQL injection attacks.

Function registrar_movimiento()

Purpose: This function handles GET requests on the '/register_movement' route and renders the 'register_movement.html' template.

Parameters:
- None (only receives GET requests).

Operations:
- Renders the 'register_movement.html' template.

Considerations:
- Access Security: Ensure that only authenticated and authorized users can access this page.

- Implement appropriate authentication and authorization measures to protect sensitive information.

Function modificar_articulo()

Purpose: This function handles POST requests on the '/modify_item' route and renders the 'modify_item.html' template with the provided data.

Parameters:
- POST form data to modify an item, including name, quantity, price, and id.

Operations:
- Renders the 'modify_item.html' template with the provided data from the POST request.

Considerations:
- Input Validation: Validate the user-input data to ensure it is valid and secure before modifying the item.

- Access Security: Ensure that only authorized users can make modifications to the items.

Function logout()

Purpose: This function handles GET and POST requests on the '/logout' route and redirects the user to the login page.

Parameters:
- None (only receives GET and POST requests).

Operations:
- Redirects the user to the login page.

Considerations:

- Session Security: Ensure that the logout process is done securely and that all active user sessions are invalidated properly.
- Implement measures to prevent session hijacking and other session management-related vulnerabilities.

Function agregar_al_carrito() and agregar_al_carrito_empleado()

Purpose: These functions handle the logic for adding an item to a user's shopping cart.

Parameters: They do not directly receive parameters from the URL. However, they expect to receive the product ID to add through form data.

Operations:

- Retrieves the current user from the database.
- Checks if the product is already in the user's cart.
- If the product is in the cart, increments the quantity.
- If the product is not in the cart, adds it with a quantity of 1.
- Updates the database with the new cart information.

Considerations:
- Handles exceptions related to the database, such as lack of stock, and prints an appropriate message in case of error.

Function ver_carrito()

Purpose: This function displays the items in a user's shopping cart.

Parameters: Expects to receive form data containing user information.

Operations:

- Retrieves the ID and user type from the database.
- Queries the products in the user's cart.
- Calculates the total purchase.
- Renders the 'compras.html' template with the cart information and total.

Considerations:

- Utilizes user information to customize the page view, showing different options based on the user's type.

Function agregar_articulo()

Purpose: Adds a new item to the inventory of products.

Parameters: Receives form data including the name, quantity, and price of the product.

Operations:

- Inserts the new product into the database.
- Handles exceptions in case of integrity violations, such as duplicating product names.

Considerations:
- Uses transactions to ensure database consistency and handle errors safely.

Function eliminar_articulo()

Purpose: Deletes an item from the inventory of products.

Parameters: Receives the ID of the product to delete through form data.

Operations:
- Deletes the corresponding product from the database.

- Handles exceptions in case of integrity violations or other errors.

Considerations:
- Uses transactions to ensure database integrity and handle errors appropriately.

Function buscar_articulo()

Purpose: Searches for an item in the inventory of products.

Parameters: Receives a search string through form data.

Operations:
- Retrieves all products from the database.
- Filters the products matching the search string using the difflib library.
- Renders the 'auditor.html' template with the filtered products.

Considerations:
- Uses string matching techniques to improve search accuracy.

Function factura()

Purpose: Generates an invoice for the items in the shopping cart.

Parameters: Receives form data including the items in the cart, total purchase, user information, and user type.

Operations:

- Renders the 'invoice.html' template with the provided data.
- Considers:
- Personalizes the invoice according to the user type, showing different options and details.

Function gracias()

Purpose: Displays a thank you message after completing a transaction.

Parameters: Receives the client type through form data.

Operations:

- Renders the 'thank_you_employee.html' template if the client type is "employee"; otherwise, renders 'thank_you.html'.

Function factura_generada()

Purpose: Generates and saves an invoice and creates a PDF file.

Parameters: Receives various fields related to the invoice and items through form data.

Operations:

- Retrieves user and client data.
- Saves the invoice in the database if necessary.
- Updates product quantities and clears the shopping cart.
- Renders the 'generated_invoice.html' template and creates a PDF using the weasyprint library.
- Sends the PDF as a downloadable file and redirects the user to the client page.

Considerations:
- Uses UUID to generate a unique identifier for the invoice.
- Uses transactions to ensure database integrity during invoice generation.

File Organization

```
├── app.py            # Código principal de la aplicación
```

```
├── templates/          # Plantillas HTML
│   ├── compras.html
│   ├── auditor.html
│   ├── factura.html
│   ├── gracias.html
│   ├── gracias_empleado.html
│   ├── factura_generada.html
│   ├── other_html_files.html
└── static/             # Archivos estáticos (CSS, JS, imágenes)
```

Additional Notes:

Ensure that the Oracle database is properly configured, and the necessary schemas exist.

Customize usernames, passwords, and DSN in the get_db function to adapt to your environment.

PDF generation uses WeasyPrint, which must be correctly installed and configured in your development environment.