DOCUMENTACIÓN TÉCNICA

Sistema de Gestión ALS - Aplicación Web Flask

Autor: Diego Camarena Gutiérrez

DNI: 15970795N

Asignatura: ALS (Análisis, Lógica y Sistemas)

Fecha: Junio 2025

DIAGRAMAS DE SECUENCIA - SISTEMA ALS

1. Diagrama de Secuencia - Crear Pedido Completo

```
sequenceDiagramparticipant U as Usuarioparticipant R as
Routes/Controllerparticipant F as Formsparticipant M as Modelsparticipant S as
StorageServiceparticipant DB as Base de DatosNote over U,DB: Proceso completo de
creación de pedido%% Paso 1: Información básicaU->>R: GET
/pedidos/crear/paso1R->>S: get_by_criteria(Cliente, activos)S->>DB: Consultar
clientes activosDB-->>S: Lista de clientesS-->>R: clientesR->>F:
PedidoForm(choices=clientes)F-->>R: formulario_inicialR-->>U:
render_template(paso1, form)U->>R: POST /pedidos/crear/paso1
(datos_basicos)R->>F: form.validate_on_submit()F->>M:
Pedido.validar_cliente_existe()M->>S: get(Cliente, cliente_id)S->>DB: Buscar
clienteDB-->>S: cliente_dataS-->>M: cliente_validoM-->>F: validacion_okF-->>R:
validacion_exitosaR->>R: session['pedido_data'] = datosR-->>U: redirect(paso2)%%
Paso 2: Agregar productosU->>R: GET /pedidos/crear/paso2R->>R: pedido_data =
session['pedido_data']R-->>U: render_template(paso2)U->>R: POST
/pedidos/crear/paso2 (productos_data)R->>R: products =
json.loads(productos_data)loop Para cada productoR->>M:
validar_producto_existe()M->>S: get(Producto, producto_id)S->>DB: Consultar
productoDB-->>S: producto_dataS-->>M: producto_validoM-->>R: validacion_okend%%
Crear pedido y itemsR->>M: Pedido(cliente_id, descripcion, fecha_entrega)M->>M:
_generar_numero_pedido()M-->>R: pedido_instanciaR->>S: save(pedido)S->>DB: INSERT
pedidoDB-->>S: pedido_idS-->>R: pedido_guardadoloop Para cada producto en
productsR->>M: ItemPedido(producto_id, talla, color, cantidad, precio)M->>M:
subtotal = precio * cantidadM-->>R: item_instanciaR->>S: save(item)S->>DB: INSERT
item_pedidoDB-->>S: item_idS-->>R: item_guardadoend%% Calcular totalesR->>R:
calcular_totales_pedido(pedido, pedido_id)R->>S: get_by_criteria(ItemPedido,
pedido_items)S->>DB: Consultar items del pedidoDB-->>S: items_listS-->>R:
itemsR->>M: pedido.calcular_totales()M->>M: subtotal, iva, total, utilidadM-->>R:
totales_calculadosR->>S: save(pedido)S->>DB: UPDATE pedido SET totalesDB-->>S:
actualizadoS-->>R: pedido_actualizadoR-->>U: redirect(detalle_pedido)
```

2. Diagrama de Secuencia - Gestión de Productos (CRUD)

```
sequenceDiagramparticipant U as Usuarioparticipant R as
Routes/Controllerparticipant F as Formsparticipant M as Modelsparticipant S as
StorageServiceparticipant DB as Base de DatosNote over U,DB: Operaciones CRUD de
Productos%% CREATE - Crear Productorect rgb(200, 255, 200)Note over U,DB: CREATE
- Nuevo ProductoU->>R: GET /productos/nuevoR->>F: ProductoForm()F-->>R:
formulario_vacioR-->>U: render_template(nuevo, form)U->>R: POST /productos/nuevo
(datos_producto)R->>F: form.validate_on_submit()F->>F:
validate_tallas_disponibles()F->>F: validate_colores_disponibles()F-->>R:
validacion_exitosaR->>M: Producto(nombre, categoria, precio_base, descripcion,
tallas, colores)M->>M: inicializar_posiciones_personalizacion()M->>M:
inicializar_estadisticas()M-->>R: producto_instanciaR->>S: save(producto)S->>DB:
INSERT productoDB-->>S: producto_idS-->>R: producto_guardadoR-->>U:
redirect(ver_producto)end%% READ - Ver Productosrect rgb(200, 200, 255)Note over
U,DB: READ - Listar ProductosU->>R: GET /productos/R->>S:
get_by_criteria(Producto, activos)S->>DB: SELECT * FROM productos WHERE
is_active=trueDB-->>S: lista_productosS-->>R: productosR-->>U:
render_template(lista, productos)U->>R: GET /productos/{id}R->>S:
load(producto_id)S->>DB: SELECT * FROM productos WHERE id=?DB-->>S:
producto_dataS-->>R: productoR->>S: find_by_condition(ItemPedido,
producto_usado)S->>DB: SELECT items WHERE producto_id=?DB-->>S:
items_estadisticasS-->>R: estadisticas_usoR-->>U: render_template(detalle,
producto, estadisticas)end%% UPDATE - Actualizar Productorect rgb(255, 255,
200)Note over U,DB: UPDATE - Editar ProductoU->>R: GET
/productos/{id}/editarR->>S: load(producto_id)S->>DB: SELECT productoDB-->>S:
producto_dataS-->>R: productoR->>F: ProductoForm(data=producto_data)F-->>R:
formulario_prellenadoR-->>U: render_template(editar, form)U->>R: POST
/productos/{id}/editar (datos_actualizados)R->>F:
form.validate_on_submit()F-->>R: validacion_exitosaR->>M:
producto.actualizar(nuevos_datos)M->>M: update_timestamp()M-->>R:
producto_actualizadoR->>S: save(producto)S->>DB: UPDATE producto SET
camposDB-->>S: actualizadoS-->>R: guardado_exitosoR-->>U:
redirect(ver_producto)end%% DELETE - Eliminar Productorect rgb(255, 200, 200)Note
over U,DB: DELETE - Eliminar Producto (Soft Delete)U->>R: POST
```

```
/productos/{id}/eliminarR->>S: load(producto_id)S->>DB: SELECT productoDB-->>S:
producto_dataS-->>R: producto%% Verificar integridad referencialR->>S:
find_by_condition(ItemPedido, items_activos)S->>DB: SELECT items WHERE
producto_id=? AND is_active=trueDB-->>S: items_activosS-->>R:
items_encontradosalt items_activos existeR->>S: find_by_condition(Pedido,
pedidos_activos)S->>DB: SELECT pedidos activos con itemsDB-->>S:
pedidos_activosS-->>R: verificacion_pedidosR-->>U: error("Producto en uso en
pedidos activos")else sin_items_activosR->>M: producto.soft_delete()M->>M:
is_active = FalseM->>M: update_timestamp()M-->>R: producto_eliminadoR->>S:
save(producto)S->>DB: UPDATE producto SET is_active=falseDB-->>S:
actualizadoS-->>R: eliminacion_exitosaR-->>U: success("Producto eliminado")endend
```

3. Diagrama de Secuencia - Autenticación de Usuario

```
sequenceDiagramparticipant U as Usuarioparticipant R as
Routes/Controllerparticipant F as Formsparticipant A as Auth/Loginparticipant S
as StorageServiceparticipant DB as Base de Datosparticipant Sess as SessionNote
over U,DB: Proceso de Autenticación%% LoginU->>R: GET /auth/loginR->>F:
LoginForm()F-->>R: formulario_loginR-->>U: render_template(login, form)U->>R:
POST /auth/login (username, password)R->>F: form.validate_on_submit()F-->>R:
datos_validosR->>S: get_by_criteria(User, username)S->>DB: SELECT user WHERE
username=?DB-->>S: user_dataS-->>R: usuarioalt usuario_existeR->>A:
user.check_password(password)A->>A: verificar_hash_passwordA-->>R:
password_correctoalt password_correctoR->>A: login_user(user)A->>Sess:
crear_sesion_usuarioSess-->>A: sesion_creadaA-->>R: login_exitosoR-->>U:
redirect(dashboard)else password_incorrectoR-->>U: error("Credenciales
inválidas")endelse usuario_no_existeR-->>U: error("Usuario no encontrado")end%%
LogoutU->>R: GET /auth/logoutR->>A: logout_user()A->>Sess:
destruir_sesionSess-->>A: sesion_destruidaA-->>R: logout_exitosoR-->>U:
redirect(login)
```

4. Diagrama de Secuencia - Calculadora de Costes de Personalización

```
sequenceDiagramparticipant U as Usuarioparticipant R as
Routes/Controllerparticipant F as Formsparticipant M as Modelsparticipant C as
Calculadoraparticipant S as StorageServiceparticipant DB as Base de DatosNote
over U,DB: Cálculo de Costes de PersonalizaciónU->>R: GET
/pedidos/agregar_item/{pedido_id}R->>S: get(Producto, productos_activos)S->>DB:
SELECT productos WHERE is_active=trueDB-->>S: productos_listaS-->>R:
productosR->>F: ItemPedidoForm(choices=productos)F-->>R: formulario_itemR-->>U:
render_template(agregar_item, form)U->>R: POST /api/producto/{producto_id}
(AJAX)R->>S: get(Producto, producto_id)S->>DB: SELECT productoDB-->>S:
producto_dataS-->>R: productoR-->>U: JSON(tallas, colores, precio_base)U->>R:
POST /api/proceso/{proceso_id} (AJAX)R->>S: get(Proceso, proceso_id)S->>DB:
SELECT procesoDB-->>S: proceso_dataS-->>R: procesoR->>M:
determinar_precio_base_proceso()M-->>R: precio_base_procesoR-->>U:
JSON(proceso_info, precio_base)Note over U,R: Usuario configura personalización
(tipo, tamaño, cantidad)U->>R: calcular_costo_personalizacion (AJAX)R->>M:
proceso.tipoalt tipo == DTF || SUBLIMACIONR->>C:
calcular_precio_dtf_sublimacion(ancho, alto, cantidad)C->>C: metros =
calcular_metros_necesarios()C->>C: precio = metros * precio_por_metroC-->>R:
precio_calculadoelse tipo == BORDADOR->>C: calcular_precio_bordado(tamaño,
cantidad, incluir_setup)C->>C: precio_base = tamaños_bordado[tamaño]C->>C:
precio_total = (precio_base * cantidad) + setupC-->>R: precio_calculadoelse tipo
== VINIL_TEXTILR->>C: calcular_precio_vinil(tipo_vinil, area, cantidad)C->>C:
precio = area * precio_por_cm2 * cantidadC-->>R: precio_calculadoendR-->>U:
JSON(precio_personalización)U->>R: POST /pedidos/agregar_item/{pedido_id}
(item_completo)R->>F: form.validate_on_submit()F-->>R: validacion_exitosaR->>M:
ItemPedido(producto_id, talla, color, cantidad, precio_prenda)M->>M: subtotal =
precio_prenda * cantidadM-->>R: item_creadoR->>S: save(item)S->>DB: INSERT
item_pedidoDB-->>S: item_idS-->>R: item_guardado%% Procesar personalizacionesloop
Para cada diseño/personalizaciónR->>M: Personalizacion(proceso_id,
precio_proceso, cantidad)M->>M: subtotal = precio_proceso * cantidadM-->>R:
personalizacion_creadaR->>S: save(personalizacion)S->>DB: INSERT
personalizacionDB-->>S: personalizacion_idS-->>R: personalizacion_guardadaend%%
Recalcular totales del pedidoR->>R: calcular_totales_pedido(pedido,
pedido_id)R->>S: get_by_criteria(ItemPedido, items_del_pedido)S->>DB: SELECT
items WHERE pedido_id=?DB-->>S: items_listaS-->>R: itemsloop Para cada itemR->>S:
get_by_criteria(Personalizacion, personalizaciones_item)S->>DB: SELECT
personalizaciones WHERE item_id=?DB-->>S: personalizaciones_listaS-->>R:
```

```
personalizacionesR->>M: calcular_subtotal_personalizaciones()M-->>R:
subtotal_persR->>M: item.subtotal_personalizaciones = subtotal_persM-->>R:
item_actualizadoendR->>M: pedido.subtotal = sum(item.subtotal +
item.subtotal_personalizaciones)R->>M:
pedido.calcular_totales(iva_porcentaje=16.0)M->>M: iva = subtotal * 0.16M->>M:
utilidad = subtotal * (porcentaje_utilidad / 100)M->>M: total = subtotal +
ivaM-->>R: totales_calculadosR->>S: save(pedido)S->>DB: UPDATE pedido SET
subtotal, iva, total, utilidadDB-->>S: actualizadoS-->>R:
pedido_actualizadoR-->>U: redirect(detalle_pedido)
```

5. Diagrama de Secuencia - Gestión de Estados de Pedido

```
sequenceDiagramparticipant U as Usuarioparticipant R as
Routes/Controllerparticipant F as Formsparticipant M as Modelsparticipant S as
StorageServiceparticipant DB as Base de Datosparticipant N as NotificacionesNote
over U,DB: Cambio de Estado de PedidoU->>R: GET
/pedidos/{id}/cambiar_estadoR->>S: get(Pedido, pedido_id)S->>DB: SELECT
pedidoDB-->>S: pedido_dataS-->>R: pedidoR->>F:
CambiarEstadoPedidoForm(estado_actual)F-->>R: formulario_estadosR-->>U:
render_template(cambiar_estado, form)U->>R: POST /pedidos/{id}/cambiar_estado
(nuevo_estado)R->>F: form.validate_on_submit()F-->>R: validacion_exitosa%%
Validaciones de transición de estadoR->>M:
validar_transicion_estado(estado_actual, nuevo_estado)alt
transicion_validaM-->>R: transicion_permitidaR->>M:
pedido.cambiar_estado(nuevo_estado)M->>M: estado = nuevo_estadoM->>M:
update_timestamp()alt nuevo_estado == COMPLETADOM->>M: fecha_completado =
now()M-->>R: estado_actualizadoR->>N: notificar_cliente_pedido_listo()N-->>R:
notificacion_enviadaelse nuevo_estado == ENTREGADOM->>M: fecha_entrega_real =
now()M->>M: validar_pago_completo()M-->>R: estado_actualizadoR->>N:
notificar_pedido_entregado()N-->>R: notificacion_enviadaelse nuevo_estado ==
CANCELADOM->>M: fecha_cancelacion = now()M->>M: motivo_cancelacion =
form.motivoM-->>R: estado_actualizadoR->>N: notificar_cancelacion()N-->>R:
notificacion_enviadaelse otro_estadoM-->>R: estado_actualizadoendR->>S:
save(pedido)S->>DB: UPDATE pedido SET estado, timestampsDB-->>S:
actualizadoS-->>R: guardado_exitosoR-->>U: success("Estado actualizado")else
transicion_invalidaM-->>R: transicion_no_permitidaR-->>U: error("Transición de
estado no válida")end
```

Información Adicional

Patrones de Interacción:

1. **Request-Response**: Comunicación HTTP estándar entre usuario y servidor 2. **AJAX/JSON**: Para actualizaciones dinámicas sin recarga de página 3. **Session Management**: Manejo de datos temporales durante procesos multi-paso 4. **Validation Chain**: Validación en múltiples capas (cliente, servidor, base de datos)

Características de los Flujos:

1. **Integridad Referencial**: Validaciones antes de operaciones críticas 2. **Soft Delete**: Eliminación lógica para mantener historial 3. **Transaccionalidad**: Operaciones atómicas en procesos complejos 4. **Cálculos Automáticos**: Recálculo de totales tras cambios 5. **Notificaciones**: Comunicación de cambios de estado a usuarios

Manejo de Errores:

- Validación en cada capa del sistema - Rollback automático en caso de errores - Mensajes descriptivos para el usuario - Logging para auditoría y debugging