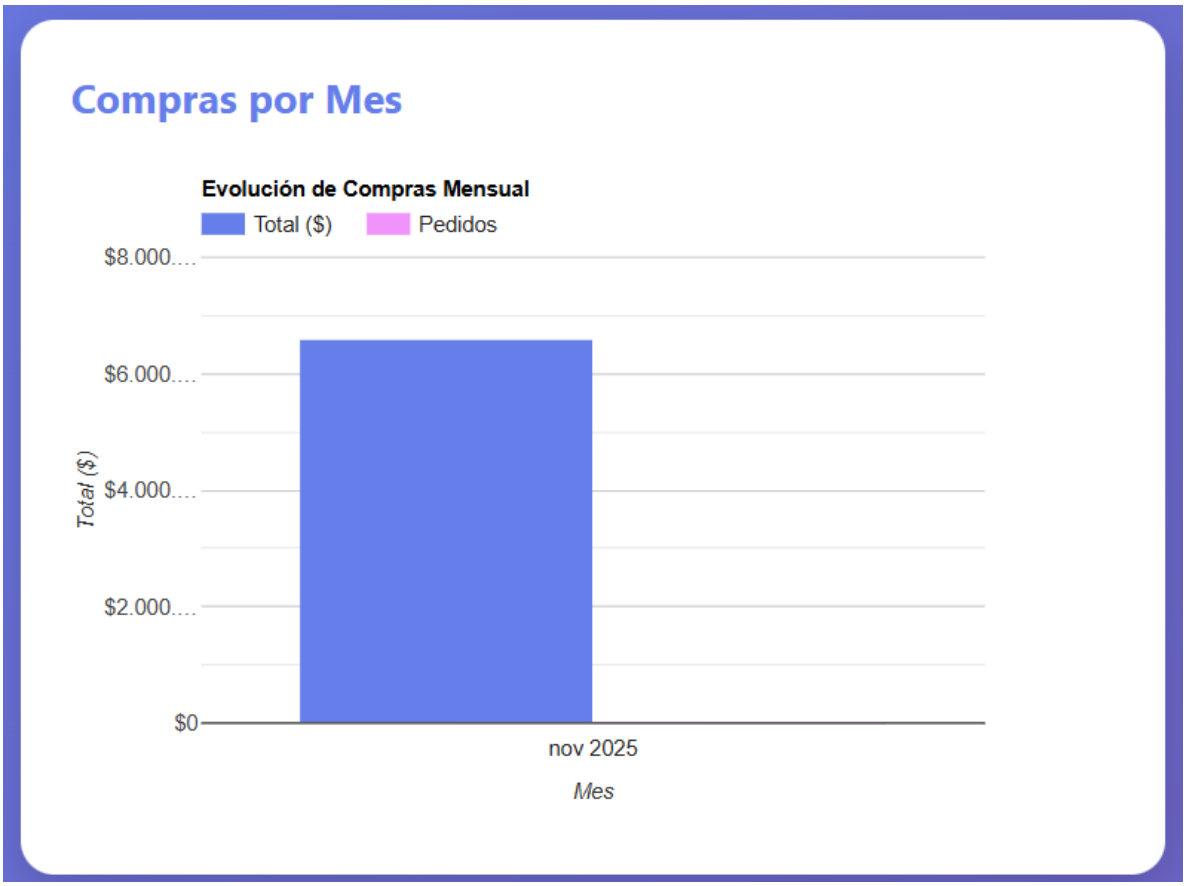
Contenido

Gráfica de ventas por mes

Código

```
// Calcular promedio mensual
Map<String, BigDecimal> ventasPorMes = pedidosRecientes.stream()
        .collect(Collectors.groupingBy(
                Pedido p -> p.getFechaPedido().format(DateTimeFormatter.ofPattern("yyyy-MM")),
                Collectors.mapping(
                        Pedido::getTotal,
                        Collectors.reducing(BigDecimal.ZERO, BigDecimal::add)
                )
        ));

BigDecimal promedioMensual = ventasPorMes.values().stream()
        .reduce(BigDecimal.ZERO, BigDecimal::add)
        .divide(BigDecimal.valueOf(ventasPorMes.size()), scale: 2, RoundingMode.HALF_UP);

// Calcular tendencia (crecimiento)
List<BigDecimal> valores = new ArrayList<>(ventasPorMes.values());
BigDecimal tendencia = BigDecimal.ZERO;
```

Vistas

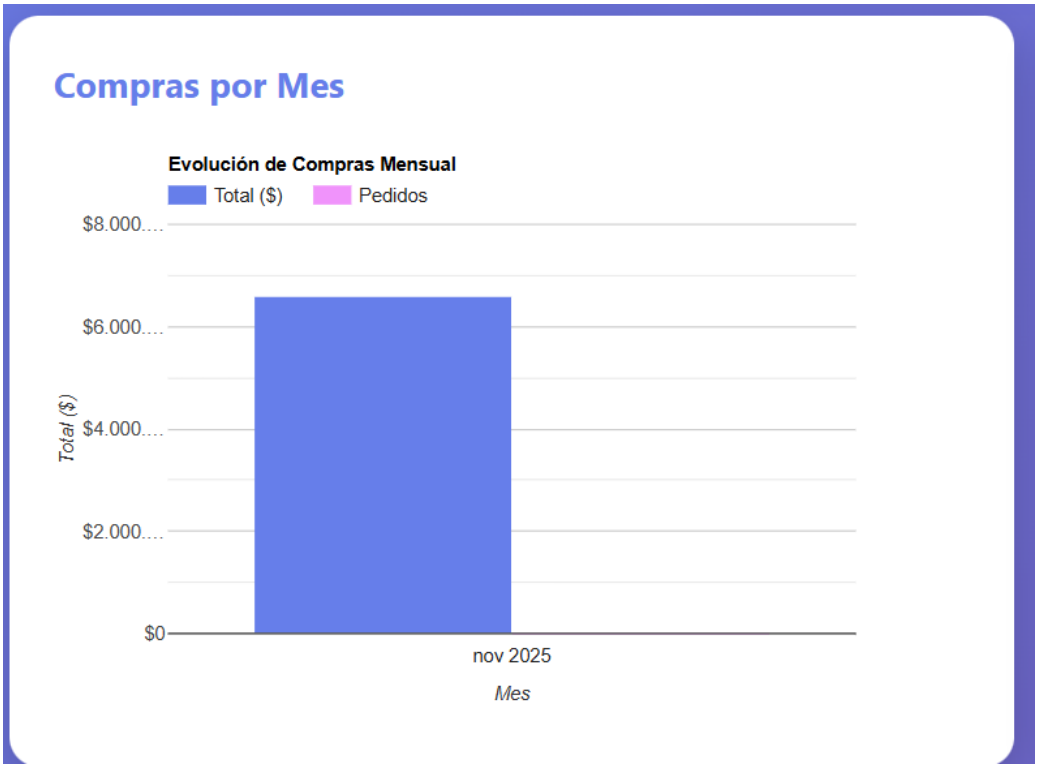


Gráfica de productos vendidos por mes

Código

```java
/**
 * Obtener compras por mes GLOBALES
 */
public List<Object[]> obtenerComprasPorMesGlobal() {  2 usages  JoseAlbertoPT
    List<Pedido> todosPedidos = pedidoRepository.findAll();

    Map<String, List<Pedido>> pedidosPorMes = todosPedidos.stream()
            .collect(Collectors.groupingBy( Pedido p -> {
                return p.getFechaPedido().format(DateTimeFormatter.ofPattern( pattern: "MMM yyyy", new Loca
            }));

    List<Object[]> resultado = new ArrayList<>();
    pedidosPorMes.forEach(( String mes,  List<Pedido> listaPedidos) -> {
        BigDecimal totalMes = listaPedidos.stream()
                .map(Pedido::getTotal)
                .reduce(BigDecimal.ZERO, BigDecimal::add);
        resultado.add(new Object[]{mes, totalMes, listaPedidos.size()});
    });

    return resultado;
}
```

Vistas

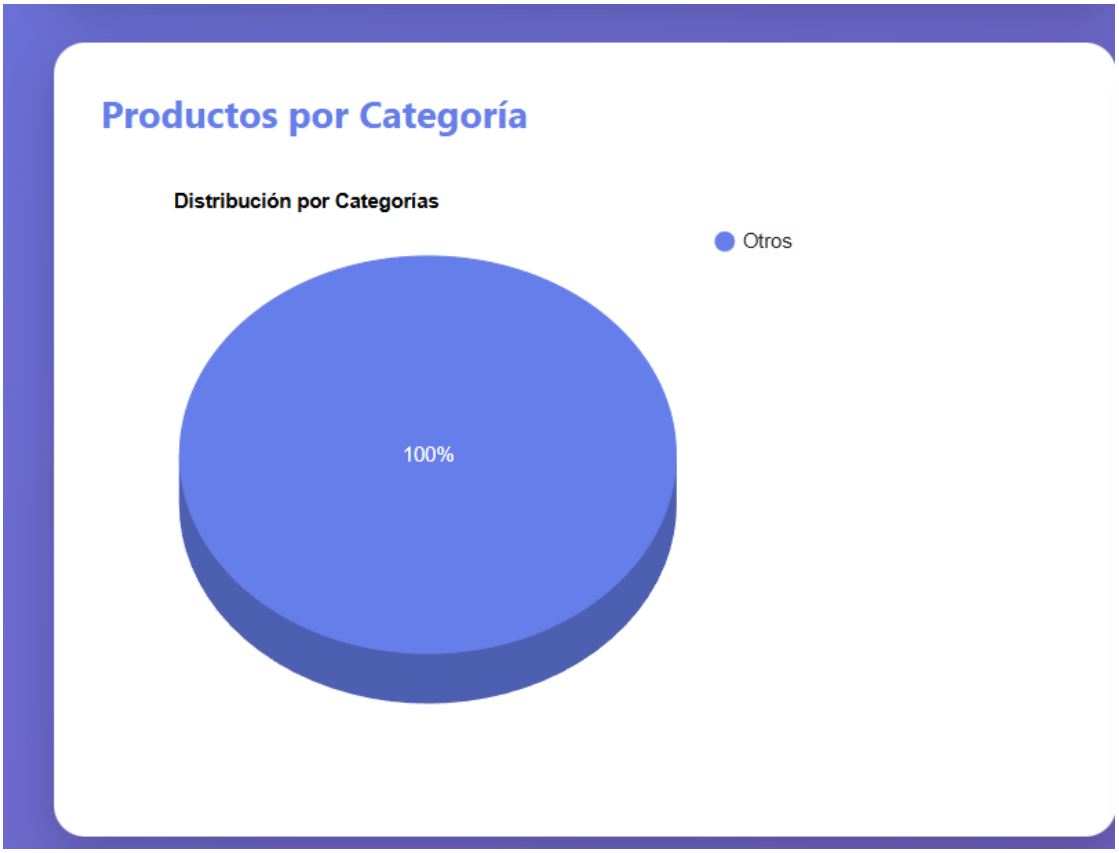


Gráfica de productos por categoría

Código

```java
/**
 * Obtener productos por categoría
 */
public List<Object[]> obtenerProductosPorCategoriaGlobal() {  2 usages  new *
    List<Pedido> todosPedidos = pedidoRepository.findAll();

    Map<String, Long> categoriaContador = new HashMap<>();

    for (Pedido pedido : todosPedidos) {
        List<PedidoDetalle> detalles = pedidoDetalleRepository.findByPedidoId(pedido.getId());

        for (PedidoDetalle detalle : detalles) {
            String categoria = extraerCategoria(detalle.getProducto());
            categoriaContador.put(
                    categoria,
                    categoriaContador.getOrDefault(categoria,  defaultValue: 0L) + detalle.getCantidad()
            );
        }
    }
```

Vistas

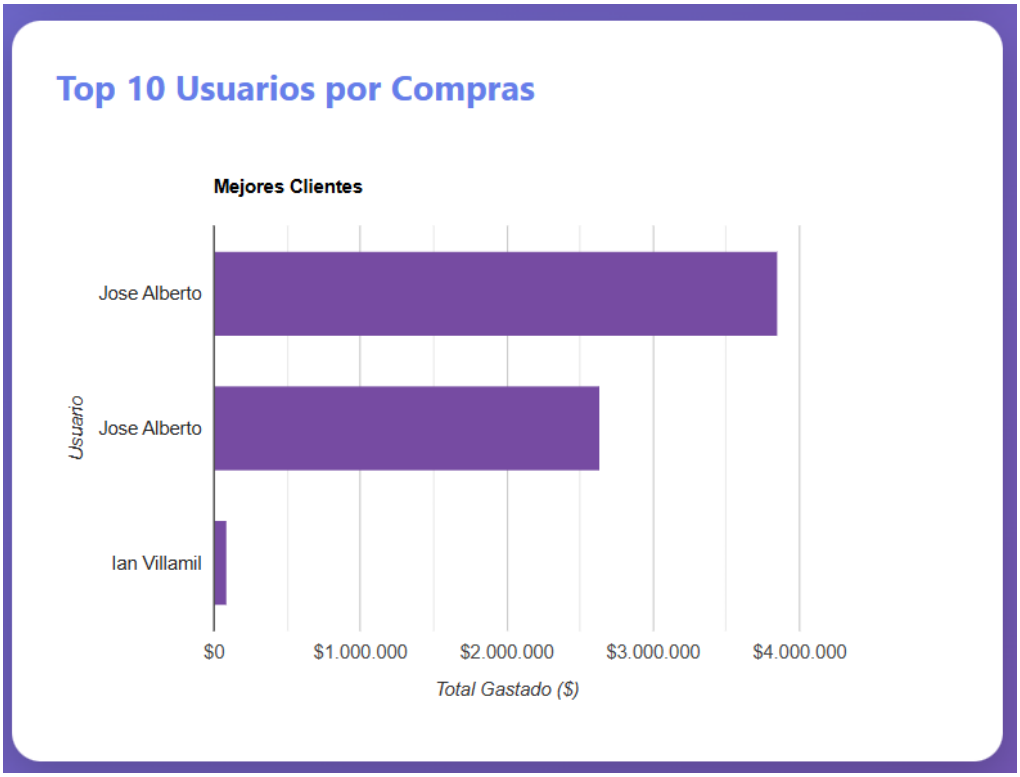


Gráfica de ventas por usuario

Código

```java
/**
 * Obtener ventas por usuario
 */
public List<Object[]> obtenerVentasPorUsuarioGlobal(int limite) {  2 usages  new *
    List<Pedido> todosPedidos = pedidoRepository.findAll();

    Map<Long, BigDecimal> ventasPorUsuario = new HashMap<>();
    Map<Long, String> nombreUsuarios = new HashMap<>();

    for (Pedido pedido : todosPedidos) {
        Long usuarioId = pedido.getUsuarioId();
        BigDecimal total = pedido.getTotal();

        ventasPorUsuario.put(
                usuarioId,
                ventasPorUsuario.getOrDefault(usuarioId, BigDecimal.ZERO).add(total)
        );

        // Guardar nombre del usuario (tomamos el nombre de dirección)
        if (!nombreUsuarios.containsKey(usuarioId)) {
            nombreUsuarios.put(usuarioId, pedido.getDireccionNombre());
        }
    }
```

Vistas



Proyección de venta

Código

```
// Agregar proyecciones futuras
BigDecimal factorCrecimiento = BigDecimal.ONE.add(
        tendencia.divide(new BigDecimal( val: "100"), scale: 4, RoundingMode.HALF_UP)
);

BigDecimal valorProyectado = promedioMensual;
for (int i = 1; i <= mesesProyeccion; i++) {
    LocalDateTime mes = ahora.plusMonths(i);
    String mesFormato = mes.format(formatter);

    valorProyectado = valorProyectado.multiply(factorCrecimiento)
            .setScale( newScale: 2, RoundingMode.HALF_UP);

    proyeccion.add(new Object[]{mesFormato, valorProyectado, "Proyección"});
}

return Map.of(
        k1: "success", v1: true,
        k2: "promedioMensual", promedioMensual,
        k3: "tendencia", tendencia,
        k4: "proyeccion", proyeccion
```

Vistas