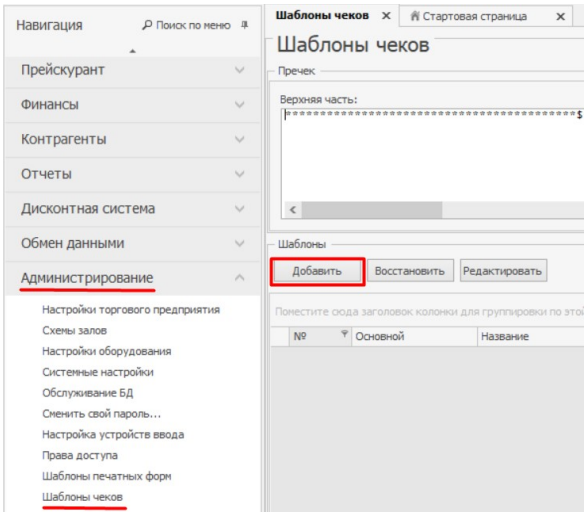
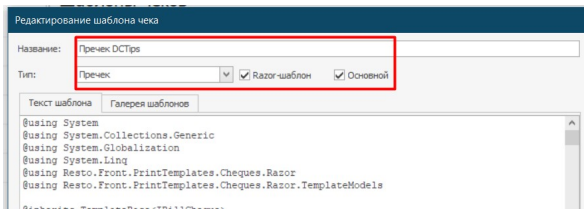
iiko. Инструкция по интеграции DCTips в Пречек

- После входа в iikoOffice (бэк офис) перейти в раздел Администрирование и выбрать подраздел Шаблоны чеков. Если этого пункта нет, значит не хватает полномочий пользователя iiko, надо запросить повышенные права у сотрудника, обслуживающего iiko.



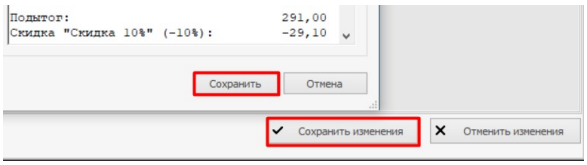
- Добавить новый шаблон с типом Пречек и галочкой Razor-шаблон и Основной. Если в разделе уже присутствует шаблон пречека, помеченный как "Основной", это значит, что клиент уже использует изменённый шаблон – необходимо скопировать его содержимое в новый шаблон и изменить код согласно инструкции.



- Далее – вставить и удалить, изменить код согласно примеру ниже (условные обозначения: добавить текст, удалить текст, изменить).

- В коде ниже измените значение с 123 в строчке var orgId = 123; на идентификатор организации из ЛК DCTips. А в строчке с var defaultWaiterId = 0; измените значение на идентификатор (id) официанта "по умолчанию". *Идентификатор (Id) организации и официанта передаются "настройщику" вместе с инструкцией.*

- Нажать на кнопку "Сохранить", а затем "Сохранить изменения" на общей вкладке раздела.




```
@using System
@using System.Collections.Generic
@using System.Globalization
@using System.Linq
@using Resto.Front.PrintTemplates.Cheques.Razor
@using Resto.Front.PrintTemplates.Cheques.Razor.TemplateModels
@using System.Text.RegularExpressions

@inherits TemplateBase<IBillCheque>
@{
    var order = Model.Order;
    var orgId = 123; // изменить на id настраиваемого ресторана
    var defaultWaiterId = 0; // изменить на код официанта "по умолчанию" из лк амиднистратора DCTips
```

```
    var urlParams = new Dictionary<string, string>
    {
        {"org", "?org="},
        {"src", "&src=3"},
        {"orderSum", "&orderSum="},
        {"orderId", "&orderId="}
    };
    var fullSum = order.GetFullSum() - order.DiscountItems.Where(di => !di.Type.PrintProductItemInPrecheque).Sum(di => di.GetDiscountS
    var categorizedDiscountItems = new List<IDiscountItem>();
    var nonCategorizedDiscountItems = new List<IDiscountItem>();
    foreach (var discountItem in order.DiscountItems.Where(di => di.Type.PrintProductItemInPrecheque && di.DiscountSums.Count > 0))
    {
        if (discountItem.IsCategorized)
        {
            categorizedDiscountItems.Add(discountItem);
        }
        else
        {
            nonCategorizedDiscountItems.Add(discountItem);
        }
    }
    var subTotal = fullSum - categorizedDiscountItems.Sum(di => di.GetDiscountSum());
    var totalWithoutDiscounts = subTotal - nonCategorizedDiscountItems.Sum(di => di.GetDiscountSum());
    var prepay = order.PrePayments.Sum(prepayItem => prepayItem.Sum);
    var total = (int)(Math.Max(totalWithoutDiscounts + order.GetVatSumExcludedFromPrice() - prepay, 0m));
    var regex = new Regex(@"(\d+)(?!.*\d)");
    var match = regex.Match(Model.Order.Waiter.Name);
}

<doc>
    @* Header (begin) *@
    @if (Model.AdditionalServiceChequeInfo == null)
    {
        <whitespace-preserve>@Raw(string.Join(Environment.NewLine, Model.Extensions.BeforeHeader))</whitespace-preserve>
    }
    <left><split><whitespace-preserve>@Model.CommonInfo.CafeSetup.BillHeader</whitespace-preserve></split></left>

    @if (Model.AdditionalServiceChequeInfo == null)
    {
        if (Model.RepeatBillNumber == 0)
        {
            <center>@Resources.BillHeaderTitle</center>
        }
        else
        {
            <center>@string.Format(Resources.RepeateBillHeaderTitleFormat, Model.RepeatBillNumber)</center>
        }
    }
    <pair left="@string.Format(Resources.BillHeaderSectionPattern, order.Table.Section.Name)" right="@string.Format(Resources.BillHead
    <pair left="@string.Format(Resources.BillHeaderOrderOpenPattern, FormatLongDateTime(order.OpenTime))" right="@string.Format(Resour

    @if (Model.AdditionalServiceChequeInfo != null)
    {
        <left>@string.Format(Resources.AdditionalServiceHeaderOrderItemsAddedPattern, FormatLongDateTime(Model.CommonInfo.CurrentTime)
    }
    <left>@string.Format(Resources.BillHeaderWaiterPattern, order.Waiter.GetNameOrEmpty())</left>
    <left>@string.Format(Resources.BillHeaderWaiterPattern,@Regex.Replace(order.Waiter.GetNameOrEmpty(),@"(\d+)(?!.*\d)",String.Empty)

    @foreach (var clientInfo in
        from discountItem in order.DiscountItems
        where discountItem.CardInfo != null
        select discountItem.CardInfo into cardInfo
        select string.IsNullOrWhiteSpace(cardInfo.MaskedCard) ? cardInfo.Owner : string.Format("{0} ({1})", cardInfo.Owner, cardInfo.M
        where !string.IsNullOrWhiteSpace(clientInfo)
        select clientInfo)
    {
        <left>@string.Format(Resources.ClientFormat, clientInfo)</left>
    }

    @if (Model.AdditionalServiceChequeInfo == null)
    {
        <whitespace-preserve>@Raw(string.Join(Environment.NewLine, Model.Extensions.AfterHeader))</whitespace-preserve>
    }

    @if (Model.AdditionalServiceChequeInfo != null)
    {
        if (order.ClientBinding != null && !string.IsNullOrWhiteSpace(order.ClientBinding.CardNumber))
        {
            <left>@string.Format(Resources.CardPattern, order.ClientBinding.CardNumber)</left>
        }
        <np />
        <center>@Resources.AdditionalServiceHeaderTitle</center>
    }
    @* Header (end) *@

    @* Body (begin) *@
    <table>
```

```
        <columns>
            <column />
            <column align="right" autowidth="" />
            <column width="2" />
            <column align="right" autowidth="" />
        </columns>
        <cells>
            @Guests()
            <linecell />
            @Summaries()
        </cells>
    </table>
    @* Body (end) *@

    @* Footer (begin) *@
    @if (@match.Success == true)
    {
        var waiterId = match.Groups[match.Groups.Count - 1].Value;
        var url = "https://tips.delivery-club.ru/pay/" + waiterId;
        <np />
        <f2>
        <center>Безналичные чаевые</center>
        <center>Delivery Club</center>
        </f2>
        <f0>
        <center>Чтобы оставить чаевые,</center>
        <center>наведите камеру на  QR-код </center>
        <np />
        <qrcode size="small" correction="low">@url@urlParams["org"]@orgId@urlParams["src"]@urlParams["orderSum"]@total@urlParams["orde
        </qrcode>
        </f0>
    }
    else
    {
        if (defaultWaiterId != 0)
        {
            var url = "https://tips.delivery-club.ru/pay/" + defaultWaiterId;
            <np/>
            <f2>
            <center>Безналичные чаевые</center>
            <center>Delivery Club</center>
            </f2>
            <f0>
            <center>Чтобы оставить чаевые,</center>
            <center>наведите камеру на QR-код </center>
            <np/>
            <qrcode size="small" correction="low">@url@urlParams["org"]@orgId@urlParams["src"]@urlParams["orderSum"]@total@urlParams["
            </qrcode>
            </f0>
        }
    }
    <np />
    @if (Model.AdditionalServiceChequeInfo == null)
    {
        <whitespace-preserve>@Raw(string.Join(Environment.NewLine, Model.Extensions.BeforeFooter))</whitespace-preserve>
    }
    <center><split><whitespace-preserve>@Model.CommonInfo.CafeSetup.BillFooter</whitespace-preserve></split></center>
    <np />
    <np />
    @if (Model.AdditionalServiceChequeInfo == null)
    {
        <whitespace-preserve>@Raw(string.Join(Environment.NewLine, Model.Extensions.AfterFooter))</whitespace-preserve>
    }
    <np />
    @* Footer (end) *@
</doc>

@helper Guests()
{
    var order = Model.Order;
    Func<IOrderItem, bool> orderItemsFilter;
    if (Model.AdditionalServiceChequeInfo != null)
    {
        orderItemsFilter = orderItem => Model.AdditionalServiceChequeInfo.AddedOrderItems.Contains(orderItem);
    }
    else
    {
        orderItemsFilter = orderItem => orderItem.DeletionInfo == null;
    }

    var guestsWithItems = Model.Order.Table.Section.DisplayGuests
        ? order.Guests.Select(guest => new
        {
            Guest = guest,
            Items = guest.Items.Where(item => orderItemsFilter(item) && OrderItemsToPrintFilter(item, order.DiscountItems))
        })
        .Where(guestWithItems => guestWithItems.Items.Any()).ToList()
```

```
        : EnumerableEx.Return(new
        {
            Guest = order.Guests.FirstOrDefault(),
            Items = order.Guests.SelectMany(g => g.Items.Where(item => orderItemsFilter(item) && OrderItemsToPrintFilter(item, order.D
        })
        .Where(guestWithItems => guestWithItems.Items.Any()).ToList();

    if (!guestsWithItems.Any())
    {
        return;
    }

    <linecell />
    <ct>@Resources.NameColumnHeader</ct>
    <ct>@Resources.ProductAmount</ct>
    <ct />
    <ct>@Resources.ResultSum</ct>
    <linecell />

    if (guestsWithItems.Count == 1)
    {
        @SingleGuest(guestsWithItems.Single().Items.ToList())
    }
    else
    {
        @OneOfMultipleGuests(guestsWithItems.First().Guest, guestsWithItems.First().Items.ToList())
        foreach (var guestWithItems in guestsWithItems.Skip(1))
        {
            <linecell symbols=" " />
            @OneOfMultipleGuests(guestWithItems.Guest, guestWithItems.Items.ToList())
        }
    }
}

@helper SingleGuest(IEnumerable<IOrderItem> items)
{
    foreach (var comboGroup in items.OrderBy(i => i.OrderRank).GroupBy(i => i.Combo))
    {
        var combo = comboGroup.Key;
        var isPartOfCombo = combo != null;
        var additionalSpace = isPartOfCombo ? " " : string.Empty;
        if (isPartOfCombo)
        {
            <ct>@combo.Name</ct>
            <ct>@FormatAmount(combo.Amount)</ct>
            <ct />
            <ct>@FormatMoney(combo.Price * combo.Amount)</ct>
        }
        foreach (var orderItemGroup in comboGroup.OrderBy(item => item.OrderRank).GroupBy(_ => _, CreateComparer<IOrderItem>(AreOrderI
        {
            var totalAmount = orderItemGroup.Sum(orderItem => orderItem.Amount);
            var totalCost = orderItemGroup.Sum(orderItem => orderItem.Cost);

            var productItem = orderItemGroup.Key as IProductItem;
            if (productItem != null && productItem.CompoundsInfo != null && productItem.CompoundsInfo.IsPrimaryComponent)
            {
                <ct><whitespace-preserve>@(additionalSpace + string.Format("{0} {1}", productItem.CompoundsInfo.ModifierSchemaName, pr
                <c colspan="3" />

                // для разделенной пиццы комменты печатаем под схемой
                if (Model.Order.Table.Section.PrintProductItemCommentInCheque && productItem.Comment != null && !productItem.Comment.D
                {
                    <c>
                        <table cellspacing="0">
                            <columns>
                                <column width="2" />
                                <column />
                            </columns>
                            <cells>
                                <c />
                                <c><split><whitespace-preserve>@(additionalSpace + productItem.Comment.Text)</whitespace-preserve></sp
                            </cells>
                        </table>
                    </c>
                    <c colspan="3" />
                }

                // у пиццы не может быть удаленных модификаторов, поэтому берем весь список
                foreach (var orderEntry in productItem.ModifierEntries.Where(orderEntry => ModifiersFilter(orderEntry, productItem, tr
                {
                    var productName = orderEntry.ProductCustomName ?? orderEntry.Product.Name;
                    <ct><whitespace-preserve>@(additionalSpace + "  " + productName)</whitespace-preserve></ct>

                    if (orderEntry.Amount != 1m)
                    {
                        <ct>@FormatAmount(orderEntry.Amount)</ct>
                    }
```

```
                    else
                    {
                        <ct />
                    }
                    <ct />

                    if (orderEntry.Cost != 0m)
                    {
                        <ct>@FormatMoney(orderEntry.Cost)</ct>
                    }
                    else
                    {
                        <ct />
                    }

                    @PrintOrderEntryAllergens(orderEntry)
                    @CategorizedDiscountsForOrderEntryGroup(new[] { orderEntry }, isPartOfCombo)
                }
            }

            if (productItem != null && productItem.CompoundsInfo != null)
            {
                <ct><whitespace-preserve>@(additionalSpace + " 1/2 " + GetOrderEntryNameWithProductSize(productItem))</whitespace-pres
            }
            else
            {
                <ct><whitespace-preserve>@(additionalSpace + GetOrderEntryNameWithProductSize(orderItemGroup.Key))</whitespace-preserv
            }

            <ct>@FormatAmount(totalAmount)</ct>
            if (!isPartOfCombo)
            {
                <ct />
                <ct>@FormatMoney(totalCost)</ct>
            }
            else
            {
                <c colspan="2" />
            }

            if (productItem != null)
            {
                @PrintOrderEntryAllergens(productItem)
            }

            @CategorizedDiscountsForOrderEntryGroup(orderItemGroup.ToList<IOrderEntry>(), isPartOfCombo)

            // здесь комменты для обычных блюд и целых пицц
            if (Model.Order.Table.Section.PrintProductItemCommentInCheque && productItem != null && productItem.Comment != null && !pr
            {
                <c>
                    <table cellspacing="0">
                        <columns>
                            <column width="2" />
                            <column />
                        </columns>
                        <cells>
                            <c />
                            <c><split><whitespace-preserve>@(additionalSpace + productItem.Comment.Text)</whitespace-preserve></split>
                        </cells>
                    </table>
                </c>
                <c colspan="3" />
            }

            foreach (var orderEntry in orderItemGroup.Key.GetNotDeletedChildren().Where(orderEntry => ModifiersFilter(orderEntry, orde
            {
                var productName = orderEntry.ProductCustomName ?? orderEntry.Product.Name;
                <ct><whitespace-preserve>@(additionalSpace + "  " + productName)</whitespace-preserve></ct>

                if (orderEntry.Amount != 1m)
                {
                    <ct>@FormatAmount(orderEntry.Amount)</ct>
                }
                else
                {
                    <ct />
                }
                <ct />

                if (orderEntry.Cost != 0m)
                {
                    <ct>@FormatMoney(orderEntry.Cost)</ct>
                }
                else
                {
                    <ct />
```

```
                }

                @PrintOrderEntryAllergens(orderEntry)
                @CategorizedDiscountsForOrderEntryGroup(new[] { orderEntry }, isPartOfCombo)
            }
        }
    }
}

@helper CategorizedDiscountsForOrderEntryGroup(ICollection<IOrderEntry> entries, bool isPartOfCombo)
{
    var orderEntry = entries.First();
    var additionalSpace = isPartOfCombo ? " " : string.Empty;
    if (orderEntry.Cost != 0m)
    {
        var categorizedDiscounts = from discountItem in Model.Order.DiscountItems
                                   where discountItem.IsCategorized && discountItem.PrintDetailedInPrecheque
                                   let discountSum = entries.Sum(entry => discountItem.GetDiscountSumFor(entry))
                                   where discountSum != 0m
                                   select new
                                   {
                                       IsDiscount = discountSum > 0m,
                                       Sum = Math.Abs(discountSum),
                                       Percent = Math.Abs(CalculatePercent(entries.Sum(entry => entry.Cost), discountSum)),
                                       Name = discountItem.Type.PrintableName,
                                       discountItem.Type.DiscountBySum
                                   } into discount
                                   orderby discount.IsDiscount descending
                                   select discount;

        foreach (var categorizedDiscount in categorizedDiscounts)
        {
            <c colspan="3">
                <whitespace-preserve>@(additionalSpace + GetFormattedDiscountDescriptionForOrderItem(categorizedDiscount.IsDiscount, c
            </c>
            <ct>@GetFormattedDiscountSum(categorizedDiscount.IsDiscount, categorizedDiscount.Sum)</ct>
        }
    }
}

@helper OneOfMultipleGuests(IGuest guest, ICollection<IOrderItem> items)
{
    <c colspan="0">@guest.Name</c>
    @SingleGuest(items)
    <c colspan="3" />
    <c><line /></c>

    var includedEntries = items.SelectMany(item => item.ExpandIncludedEntries()).ToList();
    var total = includedEntries.Sum(orderEntry => orderEntry.Cost);
    var totalWithoutCategorizedDiscounts = total - (from orderEntry in includedEntries
                                                    from discountItem in Model.Order.DiscountItems
                                                    where discountItem.IsCategorized
                                                    select discountItem.GetDiscountSumFor(orderEntry)).Sum();

    var totalWithoutDiscounts = totalWithoutCategorizedDiscounts - (from orderEntry in includedEntries
                                                                    from discountItem in Model.Order.DiscountItems
                                                                    where !discountItem.IsCategorized
                                                                    select discountItem.GetDiscountSumFor(orderEntry)).Sum();

    if (totalWithoutCategorizedDiscounts != totalWithoutDiscounts)
    {
        <c colspan="3">@Resources.BillFooterTotalPlain</c>
        <ct>@FormatMoney(totalWithoutCategorizedDiscounts)</ct>

        var nonCategorizedDiscounts = from discountItem in Model.Order.DiscountItems
                                      where !discountItem.IsCategorized
                                      let discountSum = includedEntries.Sum(orderEntry => discountItem.GetDiscountSumFor(orderEntry))
                                      select new
                                      {
                                          IsDiscount = discountSum > 0m,
                                          Sum = Math.Abs(discountSum),
                                          Percent = Math.Abs(CalculatePercent(includedEntries.Sum(entry => entry.Cost), discountSum)),
                                          Name = discountItem.Type.PrintableName,
                                          discountItem.Type.DiscountBySum
                                      } into discount
                                      orderby discount.IsDiscount descending
                                      select discount;

        foreach (var nonCategorizedDiscount in nonCategorizedDiscounts)
        {
            <c colspan="3">
                @(nonCategorizedDiscount.DiscountBySum
                ? GetFormattedDiscountDescriptionShort(nonCategorizedDiscount.IsDiscount, nonCategorizedDiscount.Name)
                : GetFormattedDiscountDescriptionDetailed(nonCategorizedDiscount.IsDiscount, nonCategorizedDiscount.Name, nonCategoriz
            </c>
            <ct>@GetFormattedDiscountSum(nonCategorizedDiscount.IsDiscount, nonCategorizedDiscount.Sum)</ct>
        }
```

```
    }

    <c colspan="3">@string.Format(Model.AdditionalServiceChequeInfo == null ? Resources.BillFooterTotalGuestPattern : Resources.Additi
    <ct>@FormatMoney(totalWithoutDiscounts)</ct>
}

@helper Summaries()
{
    var order = Model.Order;
    var fullSum = order.GetFullSum() - order.DiscountItems.Where(di => !di.Type.PrintProductItemInPrecheque).Sum(di => di.GetDiscountS
    var categorizedDiscountItems = new List<IDiscountItem>();
    var nonCategorizedDiscountItems = new List<IDiscountItem>();

    foreach (var discountItem in order.DiscountItems.Where(di => di.Type.PrintProductItemInPrecheque && di.DiscountSums.Count > 0))
    {
        if (discountItem.IsCategorized)
        {
            categorizedDiscountItems.Add(discountItem);
        }
        else
        {
            nonCategorizedDiscountItems.Add(discountItem);
        }
    }

    var subTotal = fullSum - categorizedDiscountItems.Sum(di => di.GetDiscountSum());
    var totalWithoutDiscounts = subTotal - nonCategorizedDiscountItems.Sum(di => di.GetDiscountSum());
    var prepay = order.PrePayments.Sum(prepayItem => prepayItem.Sum);
    var total = Math.Max(totalWithoutDiscounts + order.GetVatSumExcludedFromPrice() - prepay, 0m);

    if (Model.DiscountMarketingCampaigns != null)
    {
        total -= Model.DiscountMarketingCampaigns.TotalDiscount;
        totalWithoutDiscounts -= Model.DiscountMarketingCampaigns.TotalDiscount;
    }

    var vatSumsByVat = (Model.AdditionalServiceChequeInfo == null
            ? order.GetIncludedEntries()
            : Model.AdditionalServiceChequeInfo.AddedOrderItems.SelectMany(item => item.ExpandIncludedEntries()))
        .Where(orderEntry => !orderEntry.VatIncludedInPrice)
        .GroupBy(orderEntry => orderEntry.Vat)
        .Where(group => group.Key != 0m)
        .Select(group => new { Vat = group.Key, Sum = group.Sum(orderEntry => orderEntry.ExcludedVat) })
        .ToList();

    var vatSum = vatSumsByVat.Sum(vatWithSum => vatWithSum.Sum);

    if ((prepay != 0m || fullSum != total) && Model.AdditionalServiceChequeInfo == null)
    {
        <c colspan="3">@Resources.BillFooterFullSum</c>
        <ct>@FormatMoney(fullSum)</ct>
    }

    @PrintOrderDiscounts(categorizedDiscountItems, fullSum)

    if (categorizedDiscountItems.Any())
    {
        <c colspan="3">@Resources.BillFooterTotalPlain</c>
        <ct>@FormatMoney(subTotal)</ct>
    }

    @PrintOrderDiscounts(nonCategorizedDiscountItems, fullSum)

    if (Model.DiscountMarketingCampaigns != null)
    {
        foreach (var discountMarketingCampaign in Model.DiscountMarketingCampaigns.Campaigns)
        {
            <c colspan="3">@discountMarketingCampaign.Name</c>
            <ct>@("-" + FormatMoney(discountMarketingCampaign.TotalDiscount))</ct>
        }
    }

    if (prepay != 0m && (categorizedDiscountItems.Any() || nonCategorizedDiscountItems.Any()))
    {
        <c colspan="3">@Resources.BillFooterTotalWithoutDiscounts</c>
        <ct>@FormatMoney(totalWithoutDiscounts)</ct>
    }

    if (vatSum != 0m)
    {
        foreach (var vatWithSum in vatSumsByVat)
        {
            <c colspan="3">@string.Format(Resources.VatFormat, vatWithSum.Vat)</c>
            <ct>@string.Format(FormatMoney(vatWithSum.Sum))</ct>
        }
        if (vatSumsByVat.Count > 1)
        {
```

```
            <c colspan="3">@Resources.VatSum</c>
            <ct>@FormatMoney(vatSum)</ct>
        }
    }

    if (Model.AdditionalServiceChequeInfo != null)
    {
        <c colspan="3">@Resources.AdditionalServiceAddedFooterTotalUpper</c>
        <ct>@FormatMoney(Model.AdditionalServiceChequeInfo.AddedOrderItems.SelectMany(item => item.ExpandIncludedEntries()).Sum(orderE
    }

    if (prepay != 0m)
    {
        <c colspan="3">@Resources.Prepay</c>
        <ct>@FormatMoney(prepay)</ct>
    }

    <c colspan="3">@(Model.AdditionalServiceChequeInfo == null ? Resources.BillFooterTotal : Resources.AdditionalServiceFooterTotalUpp
    <ct>@FormatMoney(total)</ct>
    foreach (var rate in order.FixedCurrencyRates)
    {
        var currencyIsoName = rate.Key.IsoName;
        var defaultCurrencyIsoName = Model.CommonInfo.CafeSetup.CurrencyIsoName;
        <c colspan="2">
            <right>
                @string.Format(Resources.CurrencyRateFormat, currencyIsoName, rate.Value.ToString("f4", CultureInfo.CurrentCulture), d
            </right>
        </c>
        <ct />
        <ct>@string.Format(Resources.CurrencyFormat, currencyIsoName, FormatMoney(GetSumInAdditionalCurrency(rate.Key, rate.Value, tot
    }

    if (Model.AdditionalServiceChequeInfo != null && order.ClientBinding != null && order.ClientBinding.PaymentLimit.HasValue)
    {
        <c colspan="3">@Resources.AdditionalServiceLimit</c>
        <ct>@FormatMoney(order.ClientBinding.PaymentLimit.Value - total)</ct>
    }

    if (Model.DiscountMarketingCampaigns != null)
    {
        foreach (var discountMarketingCampaign in Model.DiscountMarketingCampaigns.Campaigns.Where(campaign => !string.IsNullOrWhiteSp
        {
            <c colspan="4">@discountMarketingCampaign.BillComment</c>
        }
    }
}

@helper PrintOrderDiscounts(IEnumerable<IDiscountItem> discountItems, decimal fullSum)
{
    foreach (var discountItem in discountItems.OrderByDescending(discountItem => discountItem.IsDiscount()))
    {
        <c colspan="3">
            @((!discountItem.IsCategorized || discountItem.PrintDetailedInPrecheque) && !discountItem.Type.DiscountBySum
                ? GetFormattedDiscountDescriptionDetailed(discountItem.IsDiscount(), discountItem.Type.PrintableName, Math.Abs(Calcula
                : GetFormattedDiscountDescriptionShort(discountItem.IsDiscount(), discountItem.Type.PrintableName))
        </c>
        <ct>@GetFormattedDiscountSum(discountItem.IsDiscount(), Math.Abs(discountItem.GetDiscountSum()))</ct>
    }
}

@helper PrintOrderEntryAllergens(IOrderEntry orderEntry)
{
    var totalAllergens = orderEntry.Allergens.ToArray();
    if (totalAllergens.Any())
    {
        <c colspan="0">@( string.Format(Resources.AllergenGroupsFormat, string.Join(", ", totalAllergens)) )</c>
    }
}

@functions
{
    /// <summary>
    /// Отфильтровывает флаерные блюда, для которых нет настройки "Печать в пречеке блюд по флаеру"
    /// </summary>
    private static bool OrderItemsToPrintFilter(IOrderItem orderItem, IEnumerable<IDiscountItem> discountItems)
    {
        return !(orderItem is IProductItem) || discountItems
            .Where(discountItem => discountItem.DiscountSums.ContainsKey(orderItem))
            .All(discountItem => discountItem.Type.PrintProductItemInPrecheque);
    }

    private bool AreOrderItemsEqual(IOrderItem x, IOrderItem y)
    {
        if (ReferenceEquals(x, y))
            return true;
        if (x == null)
            return y == null;
```

```
        if (y == null)
            return false;

        var xProductItem = x as IProductItem;
        var yProductItem = y as IProductItem;

        if (xProductItem == null || yProductItem == null || !ProductItemCanBeMerged(xProductItem) || !ProductItemCanBeMerged(yProductI
            return false;
        if (xProductItem.Product.Name != yProductItem.Product.Name)
            return false;
        if (xProductItem.ProductCustomName != yProductItem.ProductCustomName)
            return false;
        if (xProductItem.ProductSize == null ^ yProductItem.ProductSize == null)
            return false;
        if (xProductItem.ProductSize != null && yProductItem.ProductSize != null && xProductItem.ProductSize.Name != yProductItem.Prod
            return false;
        if (xProductItem.Price != yProductItem.Price)
            return false;
        if (xProductItem.Price == 0m)
            return true;

        var categorizedDiscounts = Model.Order.DiscountItems
            .Where(discountItem => discountItem.IsCategorized && discountItem.PrintDetailedInPrecheque && discountItem.DiscountSums.Co
            .ToList();

        var xCategorizedDiscountItems = categorizedDiscounts.Where(discountItem => discountItem.DiscountSums.ContainsKey(x));
        var yCategorizedDiscountItems = categorizedDiscounts.Where(discountItem => discountItem.DiscountSums.ContainsKey(y));

        return new HashSet<IDiscountItem>(xCategorizedDiscountItems).SetEquals(yCategorizedDiscountItems);
    }

    private bool ProductItemCanBeMerged(IProductItem productItem)
    {
        return productItem.CompoundsInfo == null &&
            productItem.Amount - Math.Truncate(productItem.Amount) == 0m &&
            productItem.GetNotDeletedChildren().Where(orderEntry => ModifiersFilter(orderEntry, productItem)).IsEmpty() &&
            (productItem.Comment == null || productItem.Comment.Deleted || !Model.Order.Table.Section.PrintProductItemCommentInCheque)
    }

    private static bool CommonModifiersFilter(bool isCommonModifier, IProductItem parent, bool onlyCommonModifiers)
    {
        if (parent.CompoundsInfo != null && parent.CompoundsInfo.IsPrimaryComponent)
        {
            if (onlyCommonModifiers && !isCommonModifier)
                return false;
            if (!onlyCommonModifiers && isCommonModifier)
                return false;
            return true;
        }
        return true;
    }

    private static bool ModifiersFilter(IOrderEntry orderEntry, IOrderItem parent, bool onlyCommonModifiers = false)
    {
        var parentProductItem = parent as IProductItem;
        if (parentProductItem != null && !CommonModifiersFilter(((IModifierEntry)orderEntry).IsCommonModifier, parentProductItem, only
            return false;

        if (orderEntry.Cost > 0m)
            return true;

        if (!orderEntry.Product.PrechequePrintable)
            return false;

        var modifierEntry = orderEntry as IModifierEntry;
        if (modifierEntry == null)
            return true;

        if (modifierEntry.ChildModifier == null)
            return true;

        if (!modifierEntry.ChildModifier.HideIfDefaultAmount)
            return true;

        var amountPerItem = modifierEntry.ChildModifier.AmountIndependentOfParentAmount
            ? modifierEntry.Amount
            : modifierEntry.Amount / GetParentAmount(parent, onlyCommonModifiers);

        return amountPerItem != modifierEntry.ChildModifier.DefaultAmount;
    }

    private static string GetFormattedDiscountDescriptionForOrderItem(bool isDiscount, string discountName, bool discountBySum, decima
    {
        return discountBySum
            ? string.Format("  {0}", discountName)
            : string.Format(isDiscount ? "  {0} (-{1})" : "  {0} (+{1})", discountName, FormatPercent(absolutePercent));
    }
```

```
    private static string GetFormattedDiscountDescriptionShort(bool isDiscount, string discountName)
    {
        return string.Format(isDiscount ? Resources.BillFooterDiscountNamePatternShort : Resources.BillFooterIncreaseNamePatternShort,
    }

    private static string GetFormattedDiscountDescriptionDetailed(bool isDiscount, string discountName, decimal absolutePercent)
    {
        return string.Format(isDiscount ? Resources.BillFooterDiscountNamePatternDetailed : Resources.BillFooterIncreaseNamePatternDet
            discountName, FormatPercent(absolutePercent));
    }

    private static string GetFormattedDiscountSum(bool isDiscount, decimal absoluteSum)
    {
        return (isDiscount ? "-" : "+") + FormatMoney(absoluteSum);
    }

    private static string GetOrderEntryNameWithProductSize(IOrderEntry orderEntry)
    {
        var productName = orderEntry.ProductCustomName ?? orderEntry.Product.Name;
        var productItem = orderEntry as IProductItem;
        return (productItem == null || productItem.ProductSize == null || productItem.CompoundsInfo != null)
            ? productName
            : string.Format(Resources.ProductNameWithSizeFormat, productName, productItem.ProductSize.Name);
    }

    private static decimal GetParentAmount(IOrderItem parent, bool onlyCommonModifiers)
    {
        return onlyCommonModifiers ? parent.Amount * 2 : parent.Amount;
    }
}
```