Начальная работа с OTAPI

Использование методов OTAPI в документации описано на примере Коробки ОТ и мобильного приложения для нее.

Методы OTAPI можно использовать и варьировать в работе любых сайтов на различных CMS.

Общая работа с OTAPI

Все методы можно увидеть на странице с документацией: http://docs.otapi.net/

Для получения ответа от api в формате xml нужно отправить GET/POST запрос на адрес http://otapi.net/service/

Для получения ответа от api в формате json нужно отправить GET/POST запрос на адрес http://otapi.net/service-json/

К адресу нужно добавить имя метода, и параметры в GET/POST виде, в зависимости от запроса и его размера. Некоторые параметры могут не поместиться в GET-запрос, поэтому проще сделать всегда POST.

В ответе от api обязательно приходит узел ErrorCode, если он не равен 'Ok' и не равен 'BatchError' - необходимо обработать ошибку. Ошибки необходимо разделять по значению в узлах ErrorCode и/или SubErrorCode.

Некоторые ErrorCode которые можно обрабатывать глобально на уровне приложения:

- SessionExpired - сессия покупателя или администратора истекла, необходимо предложить пользователю авторизоваться и повторить свои действия.
- AccessDenied - для данного пользователя доступ к этому методу запрещен.
- InstanceKeyBan - ключ приложения заблокирован, обратитесь к менеджерам в ваш скайп чат за подробностями. Пользователю приложения, в таком случае, желательно показать заглушку, например "На сайте ведутся технические работы".

В некоторых случаях, необходимо показать пользователю само сообщение об ошибке. Оно хранится в узле ErrorDescription и приходит уже с переводом для языка, который указан при запросе в параметре 'language'.

Старт приложения

При старте приложения, необходимо получить настройки приложения: http://docs.otapi.net/ru/Documentations/Method?name=GetCommonInstanceOptionsInfo с передачей параметра applicationType=MobileApplication/Android или applicationType=MobileApplication/iOS в зависимости от системы.

Из настроек приложение должно получить список доступных языков, и предоставить пользователю интерфейс по выбору языка приложения (если язык не один в списке). Далее все запросы к api вызываются с этим языком. При старте, приложение должно взять первый язык из списка в настройках, проверить с этим языком сессию пользователя (анонимного или авторизованного) и из предпочтений пользователя получить язык приложения (подробнее смотрите блок по работе с пользователем).

На старте так же нужно запросить список переводов для интерфейса приложения: http://docs.otapi.net/ru/Documentations/Method?name=GetApplicationUITranslations с передачей языка и параметра applicationType=MobileApplication/Android или applicationType=MobileApplication/iOS в зависимости от системы.

Список успешно полученных настроек необходимо кэшировать. Обновлять кэш настроек стоит в фоновом режиме, не замедляя этим запросом работу приложения. Обновлять кэш в фоне стоит или раз в N часов или при каждом старте приложения. Настройки одни на всё приложение и не различаются для разных пользователей.

Мобильное приложение при первом запуске должно запросить у пользователя разрешение на получение push-уведомлений. Если пользователь согласился получать push-уведомления, необходимо отправить push-токен приложения в сервисы http://docs.otapi.net/ru/Documentations/Method?name=SetUserPushNotificationToken. В будущем, приложение должно следить за актуальностью push-токена и при необходимости обновлять его.

Получение сессии (авторизация пользователя)

Работа с пользователем - основные моменты

Часть методов otapi требует параметр сессии покупателя. Перед тем как работать с логикой для пользователей, необходимо получить сессию. Изначально нужно получить анонимную сессию методом http://docs.dev.otapi.net/ru/Documentations/Method?name=GetAnonymousSession, чтобы например по ней могли уже сохраняться корзина, предпочтения, и прочее. Если при авторизации была передана предыдущая анонимная сессия, все данные автоматом перенесутся.

Полученную любым путем сессию нужно сохранить в памяти приложения, и использовать далее для всех запросов.

Любой метод, использующий сессию пользователя, может вернуть ошибку SessionExpired, это означает что сессия истекла. В этом случае нужно перекинуть пользователя снова на авторизацию с соответствующим сообщением, а когда новая сессия будет успешно получена, вернуть пользователя в то место, где возникла ошибка.

Принудительно проверить сессию проще всего методом http://docs.otapi.net/ru/Documentations/Method?name=GetUserStatusInfo. Он наименее затратен по времени.

Регистрация

Регистрация пользователей осуществляется методом http://docs.otapi.net/ru/Documentations/Method?name=RegisterUser
Перед вызовом этого метода нужно выполнить проверку входных данных:

- Email, Login, Password - обязательные параметры
- Длина строки Password не менее 6 символов
- Строка с Email действительно является email адресом

На форме регистрации должно быть дополнительное обязательное поле-флаг "Я принимаю пользовательское соглашение", без этого флага не давать выполнить регистрацию.

После успешной регистрации методом RegisterUser проверить в ответе поле Result->IsEmailVerificationUsed. Если IsEmailVerificationUsed = "true" то показать пользователю сообщение "Для активации учетной записи Вам на почту было выслано письмо со ссылкой на активацию". Если IsEmailVerificationUsed = "false" нужно сразу авторизовать пользователя (установить для него новый SessionId полученный в ответ от сервисов).

Активация после регистрации

Вместе с сообщением про то что было отправлено письмо, нужно вывести поле ввода кода активации. Когда пользователь проверит почту, введет код в приложении, нужно вызвать http://docs.otapi.net/ru/Documentations/Method?name=ConfirmEmail. В ответе ConfirmEmail есть поле Result->SessionId->Value которое содержит сессию авторизованного пользователя. На основе этой сессии нужно сразу авторизовать пользователя в приложении.

Авторизация

Авторизация пользователей осуществляется методом http://docs.otapi.net/ru/Documentations/Method?name=Authenticate
На форме авторизации должно быть поля логин(userLogin), поле пароль(userPassword) и флаг запомнить(rememberMe). Перед вызовом метода Authenticate нужно выполнить проверку входных данных:

- userLogin, userPassword - обязательные параметры

Параметр sessionId в методе Authenticate - идентификатор сессии не авторизованного покупателя.

В качестве userLogin может быть передан не только логин, но и email пользователя, возможно стоит как-то подсказать это в интерфейсе.

Авторизация через социальные сети

Авторизация через соцсети (ОТ API)

Список социальных сетей надо брать в настройках от GetCommonInstanceOptionsInfo в зависимости от текущего языка, так как например админ может оставить ВКонтакт для русского, и убрать для английского. Смотреть в TranslatableOptions->выбор по языку->ExternalAuthentications.

Восстановление пароля

Восстановление пароля осуществляется методом http://docs.otapi.net/ru/Documentations/Method?name=RequestPasswordRecovery Поле userIdentifier может содержать логин или email пользователя. После вызова RequestPasswordRecovery показать пользователю сообщение "На Ваш email высланы дальнейшие инструкции" и отобразить форму с кодом подтверждения и двумя текстовыми полями для ввода пароля. Пользователь смотрит код в почте, возвращается в приложение, вводит код и новый пароль. Если оба поля с паролем совпадают, далее нужно вызвать http://docs.otapi.net/ru/Documentations/Method?name=ConfirmNewPassword. В ответе ConfirmNewPassword есть поле Result->SessionId->Value которое содержит сессию авторизованного пользователя. На основе этой сессии нужно сразу авторизовать пользователя в приложении.

Главная страница приложения

Получение подборок

Для получения подборок, используем метод http://docs.otapi.net/ru/Documentations/Method?name=BatchSearchRatingLists. При первом запросе передаем параметры:

```
<BatchRatingListSearchParameters><UseDefaultParameters>true</UseDefaultParameters><
/BatchRatingListSearchParameters>
```

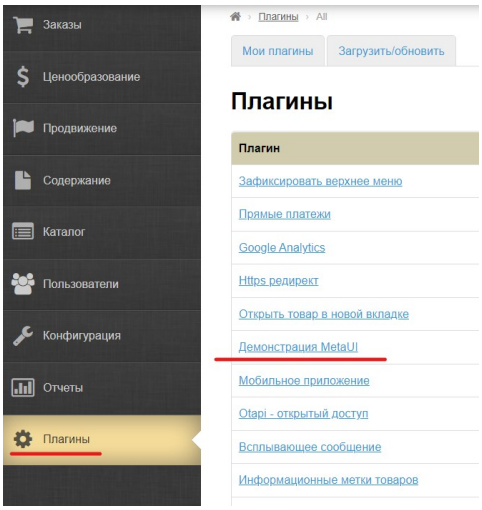
и параметр applicationType=MobileApplication/Android или applicationType=MobileApplication/iOS в зависимости от системы. В полученных данных для каждой подборки имеется порядковый номер для сортировки <Order>0</Order> , нужно отобразить на экране приложения подборки согласно этой сортировке.

Рекомендуется показывать подборки покупателю только из кэша. В кэше подборки необходимо обновлять в фоновом режиме: раз в 60 минут запрашивать метод BatchSearchRatingLists и сохранять результат в кэш. При получении ответа, нужно проверить узел HasTranslateErrors (http://docs.dev.otapi.net/ru/Documentations/Type?name=BatchRatingListsSearchResultAnswer), если узел равен "true", то кэшировать ответ не нужно. Для каждого элемента RatingList в ответе необходимо проверить узел HasError (http://docs.dev.otapi.net/ru/Documentations/Type?name=RatingListSearchResult), если узел равен "true", то этот элемент подборки кэшировать не нужно.

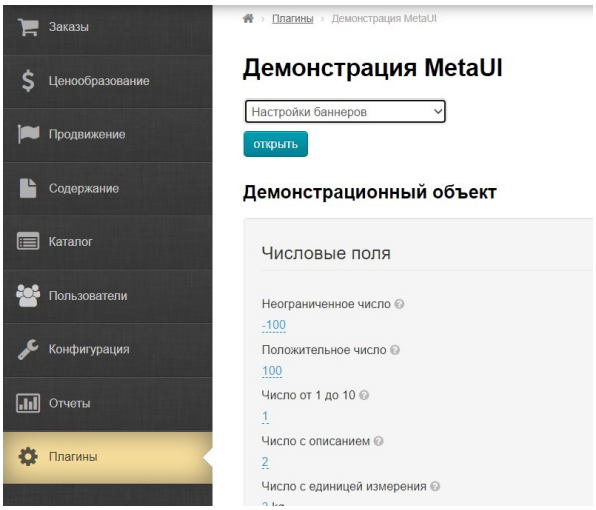
Таким образом получаем ситуацию, когда покупатель видит подборки, которые находятся в кэше. При этом подборки в кэше не имеют ошибок. Возможна ситуация когда мы показываем пользователю устаревшую информацию, при этом мы уверены что эта информация без ошибок.

Получение баннеров

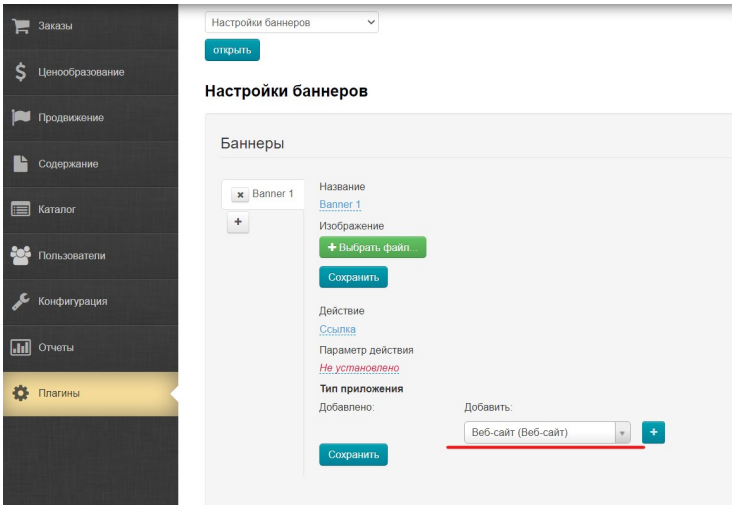
Сперва откройте плагин "Демонстрация MetaUI":



В выпадающем списке выберите "Настройки баннеров" и нажмите кнопку "Открыть":

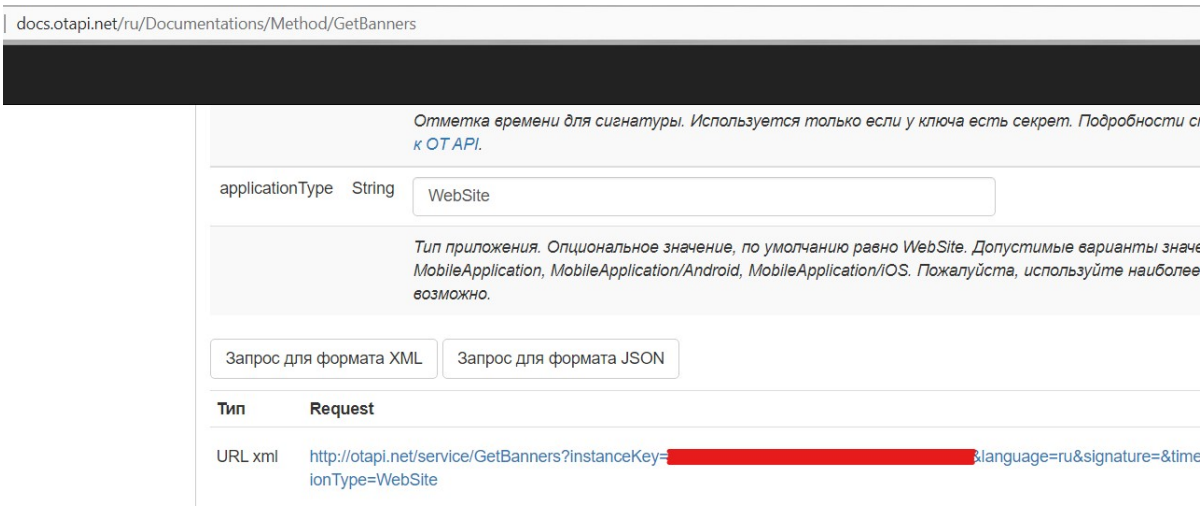


Затем заполните необходимые данные (обязательно добавьте веб-сайт):



После этого, выполните следующие шаги:

Полный список баннеров, настроенных для приложения, можно получить методом http://docs.otapi.net/ru/Documentations/Method/GetBanners , передав параметр applicationType=WebSite (applicationType=MobileApplication/Android
или applicationType=MobileApplication/iOS в зависимости от системы), а также ключ и язык. Выполните запрос и получите ссылку:



На каждый баннер будет его название, адрес картинки, тип действия, и параметр действия. Ниже перечислены текущие имеющиеся типы действий и смысл их параметров.

Тип действия	Параметр
Url	Абсолютный адрес ссылки, которую нужно открыть при нажатии на баннер
Category	Идентификатор категории, которую нужно открыть при нажатии на баннер

Короткое меню категорий

TODO:

Социальные сети

TODO:

Предпочтения пользователя

- Валюты. Список доступных валют необходимо показать из настроек приложения GetCommonInstanceOptionsInfo узел Currencies.
- Страна доставки. Список доступных стран необходимо показать из настроек приложения GetCommonInstanceOptionsInfo узел DeliveryCountries. Список стран необходимо отображать с учетом текущего языка приложения (все переводы можно найти в узле TranslatableOptions).
- Язык приложения. Список доступных языков приложения показать из настроек приложения GetCommonInstanceOptionsInfo узел Languages.

Покупатель может иметь ряд предпочтений-настроек, от которых зависят в будущем ответы otapi или поведение системы /приложения.

Получить текущие выбранные предпочтения можно с помощью метода http://docs.otapi.net/ru/Documentations/Method?name=GetUserPreferences

Изменить предпочтение пользователя можно с помощью метода http://docs.otapi.net/ru/Documentations/Method?name=UpdateUserPreferences

Интерфейс приложения должен предоставить пользователю возможность изменять свои предпочтения.

Список категорий

Для начала отправляем запрос: http://docs.otapi.net/ru/Documentations/Method?name=GetRootCategoryInfoList - получаем список корневых категорий сайта. Этот список можно кэшировать на 24 часа. Далее при попытке развернуть каждую категорию - делаем отдельный запрос http://docs.otapi.net/ru/Documentations/Method?name=GetCategorySubcategoryInfoList. Каждый список тоже можно кэшировать на 24 часа, используя в ключе идентификатор родительской категории.

Категория содержит другие категории (т.е. её можно развернуть), если для неё пришел флаг <IsParent>true</IsParent>.

Если категория скрыта <IsHidden>true</IsHidden> - её нужно не показывать в каталоге.

Флаг <IsVirtual>true</IsVirtual> означает что в категории нет товаров (только другие подкатегории).

В необязательном узле <IconImageUrl> http://open-demo.otcommerce.com/uploaded/category/zh_odezhda.jpg </IconImageUrl> может придти абсолютный адрес изображения для категории (изображение устанавливается администратором в админке).

В силу особенностей каталога, флаг <IsParent>true</IsParent> иногда возвращается ошибочно. Поэтому при открытии такой категории нужно реализовать дополнительную логику: если <IsVirtual>false</IsVirtual> и список подкатегорий пуст - открывать экран поиска товаров по данной категории.

Страница рекомендуемых продавцов

Метод http://docs.otapi.net/ru/Documentations/Method?name=BatchSearchRatingLists , xmlSearchParameters=

```
<BatchRatingListSearchParameters><RatingLists><RatingList><CategoryId>0</CategoryId><ItemRatingType>Best</ItemRatingType><IsRandomSearch>false</IsRandomSearch><ContentType>Vendor</ContentType><FramePosition>0</FramePosition><FrameSize>20</FrameSize></RatingList></RatingLists></BatchRatingListSearchParameters>
```

FrameSize - количество продавцов на странице, в ответе придет TotalCount - общее количество рекомендуемых продавцов. На основе TotalCount и FrameSize можно нарисовать пагинацию.

Страница рекомендуемых брендов

Метод http://docs.otapi.net/ru/Documentations/Method?name=BatchSearchRatingLists , xmlSearchParameters=

```
<BatchRatingListSearchParameters><RatingLists><RatingList><CategoryId>0</CategoryId><ItemRatingType>Best</ItemRatingType><IsRandomSearch>false</IsRandomSearch><ContentType>Brand</ContentType><FramePosition>0</FramePosition><FrameSize>20</FrameSize></RatingList></RatingLists></BatchRatingListSearchParameters>
```

FrameSize - количество брендов на странице, в ответе придет TotalCount - общее количество рекомендуемых брендов. На основе TotalCount и FrameSize можно нарисовать пагинацию.

Отображение продавцов

Как на странице рекомендуемых продавцов, так и в карточке товара в блоке продавца, так и при просмотре всех товаров продавца, для отображения информации о продавце нужно использовать следующие свойства:

Свойство	Описание	Где отображать	Комментарии
DisplayName	имя продавца	везде	Благодаря этому позже админ магазина сможет менять имена, а для МП ничего не поменяется в логике.
DisplayPictureUrl	картинка продавца	везде	Только если есть, иначе в рекомендуемых продавцах картинка-заглушка, в остальных местах не отображать. Благодаря этому позже админ магазина сможет менять картинки, а для МП ничего не поменяется в логике.
Credit/Level	рейтинг продавца от 1 до 20	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно.
Credit /TotalFeedbacks	число отзывов	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно.
Credit /PositiveFeedbacks	число положительных отзывов	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно. Если есть, отображать в виде % от числа общих отзывов.
Scores /DeliveryScore	оценка доставки от 1.0 до 5.0	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно.
Scores /ItemScore	оценка товаров от 1.0 до 5.0	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно.

Scores /ServiceScore	оценка услуг (или сервиса) от 1.0 до 5.0	в карточке товара на странице продавца	Только если есть и не равно 0, иначе отображать не нужно.

Поиск

Основная документация по поиску.

Для поиска товаров используем только один удобный метод поиска: http://docs.otapi.net/ru/Documentations/Method?name=BatchSearchItemsFrame

При первом открытии категории/продавца/бренда или поискового запроса, вызываем метод BatchSearchItemsFrame с параметром blockList: SubCategories,HintCategories,Vendor,Brand,Category,RootPath,SearchProperties,AvailableSearchMethods. В xmlParameters передаем параметры поиска (например ид категории <CategoryId>otc-3</CategoryId> или поисковый запрос <ItemTitle>dress</ItemTitle>). Мы рекомендуем с параметрами поиска передать: <UseOptimalFrameSize>true</UseOptimalFrameSize>, это позволит получить в ответе оптимальное количество товаров на странице для данного запроса.

Отображаем информацию полученную в ответе.

В узле AvailableSearchMethods приходят все способы поиска, доступные для текущего запроса. Если способов поиска больше чем один - нужно дать возможность покупателю изменять способ поиска.

В узле Items по значениям из узлов Provider и SearchMethod (например: <Provider>Taobao</Provider> <SearchMethod>Extended</SearchMethod>) в AvailableSearchMethods находим подробную информацию о текущем способе поиска, которым сейчас сделан запрос. По этой информации предоставляем пользователю фильтры доступные на текущей странице поиска. О соответствии полей из информации о способе поиска и параметрами поиска можно найти в основной документации.

Показываем пользователю SubCategories (Подкатегории), HintCategories (Возможно вас заинтересует). В узле Items - Categories приходят категории подходящие под текущий поисковый запрос (Уточните категорию).

Показываем товары из узла Items - Items . Проверяем у товара флаг IsSellAllowed=true и SellDisallowReason="IsFiltered" - то показываем заглушку для товара - товар запрещен к покупке. Если ErrorCode товара не равен "Ok" - вместо картинки товара, показываем ошибку: NotFound - Товар удален, NotAvailable - Товар недоступен, все остальные коды ошибок просто слово "Ошибка".

Хорошей практикой является менять отображение в зависимости от того, какие узлы вернулись в ответе на запрос BatchSearchItemsFrame. Если:

- пришел узел Vendor, показываем пользователю страницу продавца с описанием продавца;
- пришел узел Brand, показываем страницу бренда с описанием бренда;
- пришел узел Category, показываем страницу категории с описанием категории;
- иначе - обычную страницу поиска по фразе.

Такой подход позволит локализовать логику поиска и сервисы будут отвечать за логику отображения страницы.

Карточка товара

При первом открытии карточки, нужно использовать метод: http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemFullInfo , нужно передавать следующие blockList: RootPath,Vendor,MostPopularVendorItems16, Description

Отображаем карточку товара:

- Хлебные крошки категорий для данного товара (RootPath).
- Название (Item.Title).
- Видео (Item.Videos) и фото (фото представлены в трех размерах: маленький, средний, большой), в качестве главного фото всегда показываем первое изображение из списка.
- По желанию можно показать особенности товара (показываем все Item.Features у которых есть DisplayName) и вес товара (Item.Weight).
- По желанию можно показать информацию о продавце Vendor и его несколько товаров VendorItems.
- Все характеристики товара (Item.Properties) и описание товара (Description) занимают большую часть экрана, их обычно отображают в конце карточки.
- Отображаем список всех конфигураций товара (Item.Configurators, где Item.Configurators.Property - это свойство, а Item.Configurators.Property.Value - значение свойства).
  Если есть атрибут MultiInputPropertyId узла Item.Configurators.Property, данное свойство необходимо нарисовать в виде таблицы, с возможностью выбрать количество каждого значения этого свойства, пример: https://www.screencast.com/t/Ye5kaULlmWJ .

В карточке товара нужно учитывать настройки от метода GetCommonInstanceOptionsInfo, узел Showcase.ItemShowcase . Каждый узел является флагом (админ может управлять этими флагами) и сообщает какую информацию отображать покупателю, а какую скрыть:

- Title - название товара.
- Weight - вес товара.
- LinkToOriginal - ссылка на оригинал.
- SinglePrice - цена за 1 штуку.
- InternalDelivery - цена местной (внутренней) доставки.
- PriceRange - стоимость товара в зависимости от количества.

Отображаем цену товара:

используем метод: http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemConfigurationInfo , нужно передавать следующие blockList: AdditionalPrices (первый раз запрос можно сделать с пустым xmlRequest)

- Проверяем возможность купить данный товар и причину отказа от покупки (Configuration.Availability).
- Отмечаем какие свойства выбраны пользователем (Configuration.Configurators.Property.Value - Selected="true"), а какие не доступны для выбора (Configuration.Configurators.Property.Value - Disabled="true").
- Отображаем зависимость цены от количества (если пришел узел QuantityRanges).
- Если пришел узел MultiInputConfigurations, то показываем цену каждой конфигурации из MultiInputConfigurations, иначе показываем цену из узла Current:
    - Price - цена за 1шт. (если есть атрибуты Min и Max - то нужно показать не одну цену, а диапазон цен)
    - OldPrice - цена без скидки за 1шт. (если скидки нет, узел отсутствует)
    - DiscountPercent - процент скидки для OldPrice. (если скидки нет, узел отсутствует)
    - InternalDelivery - цена внутренней доставки (если узел отсутствует, показывать не надо)
    - AvailableQuantity - доступное количество для покупки.
- Выводим итоговую стоимость из узла TotalCost.
- Добавляем информацию о возможной добавочной стоимости (AdditionalPrices).

Предоставляем покупателю возможность выбирать конфигурацию и её количество (если MultiInputPropertyId нет - то поле для ввода количество одно, если MultiInputPropertyId есть - для каждого значения свойства MultiInputPropertyId можно ввести своё количество). После каждого "изменения свойства" и "изменения количества", отправляем новый запрос BatchGetSimplifiedItemConfigurationInfo с указанием выбранной пользователем конфигурации и количества, примеры xmlRequest:

```
<Request />

<Request>
    <Current />
</Request>

<Request>
    <Current Quantity="1" />
</Request>

<Request>
    <Current Quantity="5" />
</Request>
   :
<Request>
    <Current ConfigurationId="" />
</Request>

<Request>
    <Current ConfigurationId="" Quantity="5" />
</Request>

<Request>
    <Current>
        <Property Id="" ValueId=""
    </Current>
</Request>

<Request>
    <Current>
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
    </Current>
</Request>

<Request>
    <Current Quantity="5">
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
    </Current>
</Request>
```

```
<Request>
    <Selected ConfigurationId="" Quantity="1" />
    <Selected ConfigurationId="" Quantity="2" />
    ...
</Request>

<Request>
    <Selected Quantity="1">
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
    </Selected>
    <Selected Quantity="2">
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
    </Selected>
    ...
</Request>

          , :
<Request>
    <Current Quantity="5">
        <Property Id="" ValueId=""
    </Current>
    <Selected Quantity="2">
        <Property Id="" ValueId=""
        <Property Id="" ValueId=""
    </Selected>
</Request>
```

Что такое Current/Selected - и как правильно? Current или Selected?
Если нужен только 1 одновременно выбранный конфиг, то про Selected можно вообще не думать.
Если нужно несколько (как на 1688), то в Selected шлем всё где введено количество.

Избранные товары

Получить список товаров: Метод http://docs.otapi.net/ru/Documentations/Method?name=BatchGetUserData , blockList=Note.

Добавить товар в избранное http://docs.otapi.net/ru/Documentations/Method?name=AddItemToNote , fieldParameters=<Fields/>

Удалить товар из избранного http://docs.otapi.net/ru/Documentations/Method?name=RemoveItemFromNote , перед удалением требуется подтверждение удаления.

Переместить в корзину http://docs.otapi.net/ru/Documentations/Method?name=MoveItemFromNoteToBasket

Изменить кол-во товара в избранном http://docs.otapi.net/ru/Documentations/Method?name=EditNoteItemQuantity

Изменить комментарий: http://docs.otapi.net/ru/Documentations/Method?name=EditNoteItemFields , <Fields><FieldInfo Name="Comment" Value="Текст комментария"/></Fields>

Изменить конфигурацию товара:
1. доступные конфигурации получить методами http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemFullInfo и http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemConfigurationInfo - так же, как и в карточке товара.
2. сохранение новой конфигурации осуществляется добавлением новой и удалением старой, т.е. последовательными вызовами http://docs.otapi.net/ru/Documentations/Method?name=AddItemToNote (**важно** передать в метод все fieldParameters от старой записи, иначе можно потерять часть информации, например комментарий пользователя к товару) и http://docs.otapi.net/ru/Documentations/Method?name=RemoveItemFromNote

Избранные продавцы

Список продавцов: Метод http://docs.otapi.net/ru/Documentations/Method?name=SearchSimplifiedFavoriteVendors

Добавить продавца в избранное http://docs.otapi.net/ru/Documentations/Method?name=AddVendorToFavorites , fieldParameters=<Fields/>

Удалить продавца из избранного http://docs.otapi.net/ru/Documentations/Method?name=RemoveVendorFromFavorites Требуется подтверждение удаления.

Корзина

Получить список товаров: Метод http://docs.otapi.net/ru/Documentations/Method?name=BatchGetUserData , blockList=Basket.

Список товаров должен быть разделен по провайдерам, т.к. в один заказ можно оформить товары только одного провайдера. Пользователя нужно уведомить о минимальной сумме заказа (если она установлена в конфигурации), получить её можно с помощью метода: http://docs.otapi.net/ru/Documentations/Method?name=GetCommonInstanceOptionsInfo (Result->Order->MinOrderCost)

Удалить товар из корзины http://docs.otapi.net/ru/Documentations/Method?name=RemoveItemFromBasket , перед удалением требуется подтверждение удаления.

Полная очистка корзины http://docs.otapi.net/ru/Documentations/Method?name=ClearBasket , перед очисткой требуется подтверждение.

Перенести товар из корзины в избранное http://docs.otapi.net/ru/Documentations/Method?name=MoveItemFromCartToNote для переноса нескольких отмеченных товаров делается несколько вызовов MoveItemFromCartToNote

Изменение количества товара в корзине http://docs.otapi.net/ru/Documentations/Method?name=EditBasketItemQuantity

Добавление/изменение комментария к товару http://docs.otapi.net/ru/Documentations/Method?name=EditBasketItemFields <Fields><FieldInfo Name="Comment" Value="Текст комментария"/></Fields>

Редактирование веса товара http://docs.otapi.net/ru/Documentations/Method?name=EditBasketItemFields <Fields><FieldInfo Name="Weight" Value="Число нового веса товара"/></Fields>

Изменить конфигурацию товара:
1. доступные конфигурации получить методами http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemFullInfo и http://docs.otapi.net/ru/Documentations/Method?name=BatchGetSimplifiedItemConfigurationInfo - так же, как и в карточке товара.
2. сохранение новой конфигурации осуществляется добавлением новой и удалением старой, т.е. последовательными вызовами http://docs.otapi.net/ru/Documentations/Method?name=AddItemToBasket (**важно** передать в метод все fieldParameters от старой записи, иначе можно потерять часть информации, например комментарий пользователя к товару) и http://docs.otapi.net/ru/Documentations/Method?name=RemoveItemFromBasket

Группировка товаров в корзине по добавочной стоимости

Приложение должно сперва сгруппировать все строки в корзине по провайдеру.

Для каждого провайдера находим узел CollectionSummary, метод OTApi BatchGetUserData , blockList=Basket (или метод OTApi GetBasket) по узлу ProviderType
CollectionSummaries.CollectionSummary.ProviderType
В коллекции провайдера разделяем товары на группы. Проходимся по элементам AdditionalPriceInfo массива AdditionalPriceInfoList.Elements.
- Для каждого AdditionalPriceInfo указаны какие элементы корзины хранятся в данной группе (узел ElementIds), указано название группы DisplayName и цена Price
- Те строки корзины, которые не имеют добавочной стоимости - отображаются без группировки

Пример формирования данных для корзины

```
public function getBasketAction()
{
    $lang = Session::getActiveLang();
    $user = User::getObject();
    $sid = $user->getSid();
    $userAuthenticated = $user->isAuthenticated();
    $currencySign = InstanceProvider::getObject()->GetInternalCurrency()->GetCode();
    $currencyCode = $user->getCurrencyCode();

    $minOrderTotalCost = 0;
    $providers = array();
    $checked = array();
    $currentProviderExists = false;

    $currentProviderAlias = $this->request->getValue('activeProvider');
    $activeLines = $this->request->getValue('activeBasketLines');
    if ($this->request->valueExists('activeProvider')) {
        $activeLines = $activeLines ? explode(',', $activeLines) : array();
    }
    $instanceProvider = InstanceProvider::getObject();

    try {
        OTAPILib2::simpleRequest('GetBasket', [
            'language' => $lang,
            'sessionId' => $sid,
        ], $basket);
```

```
        OTAPILib2::makeRequests();
        $collectionInfo = $basket->GetRawData()->CollectionInfo;

        //
        if (isset($collectionInfo->Elements->ElementInfo)) {
            foreach ($basket->GetRawData()->CollectionInfo->Elements->ElementInfo as $item) {
                $item = new OtapiElementInfo($item);
                $basketLineId = $item->GetId();
                $itemId = $item->GetItemId();
                $provider = $item->GetProviderType();
                $product = Product::getObject($itemId, $item);

                //
                if (!key_exists($provider, $providers)) {
                    $alias = $instanceProvider->GetAliasByProviderName($lang, $provider);

                    $providerData = array();
                    $providerData['alias'] = $alias;
                    $providerData['itemsQuantity'] = 0;
                    $providerData['info'] = InstanceProvider::getObject()->GetProviderInfo($lang, $provider);

                    //     id=0.
                    $providerData['groups'][0] = array(
                        'name' => '',
                        'displayName' => '',
                        'convertedPriceList' => array(
                            'sign' => $currencySign,
                            'code' => $currencyCode,
                            'value' => 0
                        ),
                        'items' => array()
                    );

                    //
                    if (!$currentProviderAlias || $currentProviderAlias === $alias) {
                        $currentProviderExists = true;
                        $currentProviderAlias = $alias;
                        $providerData['isCurrent'] = true;
                    } else {
                        $providerData['isCurrent'] = false;
                    }

                    $providers[$provider] = $providerData;
                }

                //
                $providers[$provider]['groups'][0]['items'][$basketLineId] = $product;
                $providers[$provider]['itemsQuantity']++;
            }
        }

        //      ,   ,
        if (! $currentProviderExists) {
            $keys = array_keys($providers);
            if (! empty($keys)) {
                $firstProvider = $keys[0];
                $providers[$firstProvider]['isCurrent'] = true;
            }
        }

        //   ,
        if (isset($collectionInfo->CollectionSummaries->CollectionSummary)) {
            foreach ($basket->GetRawData()->CollectionInfo->CollectionSummaries->CollectionSummary as $summary)
{
                $summary = new OtapiCollectionSummary($summary);
                $provider = $summary->GetProviderType();

                //    ,
                foreach ($summary->GetAdditionalPriceInfoList()->GetElements()->GetAdditionalPriceInfo() as
$group) {
                    //
```

```
                $groupType = (string) $group->GetType();
                $groupName = (string) $group->GetName();
                $groupId = md5($groupType . $groupName);
                $groupDisplayName = (string) $group->GetDisplayName();

                //
                $convertedPriceList = array();
                $displayedMoneys = $group->GetPrice()->GetConvertedPriceList()->GetDisplayedMoneys();
                foreach ($displayedMoneys->GetMoney() as $displayMoney) {
                    if ($displayMoney->GetCodeAttribute() === $currencyCode) {
                        $convertedPriceList = array(
                            'sign' => (string) $displayMoney->GetSignAttribute(),
                            'code' => (string) $displayMoney->GetCodeAttribute(),
                            'value' => (float) $displayMoney->GetValue()
                        );

                        break;
                    }
                }

                //
                $providers[$provider]['groups'][$groupId] = array(
                    'name' => $groupName,
                    'displayName' => $groupDisplayName,
                    'convertedPriceList' => $convertedPriceList,
                    'items' => array(),
                );

                //
                foreach ($group->GetElementIds()->GetRawData()->children() as $basketLineId) {
                    $basketLineId = (string) $basketLineId;

                    //
                    $ungroupedItem = $providers[$provider]['groups'][0]['items'][$basketLineId];
                    //
                    $providers[$provider]['groups'][$groupId]['items'][$basketLineId] = $ungroupedItem;
                    //
                    unset($providers[$provider]['groups'][0]['items'][$basketLineId]);
                }
            }

            if (empty($providers[$provider]['groups'][0]['items'])) {
                //      ,
                unset($providers[$provider]['groups'][0]);
            }
        }
    }

    //
    $instanceSettings = InstanceProvider::getObject()->GetCommonInstanceOptionsInfo($lang);
    $minOrderTotalCost = $instanceSettings->GetOrder()->GetConvertedMinOrderCost();
    foreach ($minOrderTotalCost->GetDisplayedMoneys()->GetMoney() as $value) {
        if($value->GetSignAttribute() === User::getObject()->getCurrencySign()) {
            $currencySign = $value->GetSignAttribute();
            $minOrderTotalCost = $value->GetValue();
            break;
        }
    }

} catch (Exception $e) {
    $this->respondAjaxError($e);
}

$this->sendAjaxResponse([
    'content' => $this->renderPartial('controllers/basket/list', [
        'providers' => $providers,
        'userAuthenticated' => $userAuthenticated,
        'minOrderTotalCost' => $minOrderTotalCost,
        'currencySign' => $currencySign
    ])
]);
```

```
}
```

Профиль

Получение основной информации о пользователе (Имя, email и т.п.): http://docs.otapi.net/ru/Documentations/Method?name=GetUserInfo , редактирование этой информации возможно с помощью метода http://docs.otapi.net/ru/Documentations/Method?name=UpdateUser

Адрес доставки. Адрес доставки создается и редактируется отдельно от основной информации о пользователе. Всего возможно N адресов доставки, где N берется из настройки http://docs.otapi.net/ru/Documentations/Method?name=GetCommonInstanceOptionsInfo UserProfile->MaxProfilesCount . Методы для работы с адресом доставки:

- http://docs.otapi.net/ru/Documentations/Method?name=GetUserProfileInfoList
- http://docs.otapi.net/ru/Documentations/Method?name=ValidateUserProfile
- http://docs.otapi.net/ru/Documentations/Method?name=CreateUserProfile
- http://docs.otapi.net/ru/Documentations/Method?name=UpdateUserProfile
- http://docs.otapi.net/ru/Documentations/Method?name=DeleteUserProfile

Для пользователя, необходим интерфейс, который позволяет выбрать профиль используемый по умолчанию https://www.screencast.com/t/1fZ9HuauBjf . Выбранный профиль необходимо сохранить в предпочтения пользователя: http://docs.otapi.net/ru/Documentations/Method?name=UpdateUserPreferences , поле ProfileId.

Получение списка контентных страниц для экрана профиля: http://docs.otapi.net/ru/Documentations/Method?name=GetContentMenuItemTree , параметры:

applicationType=MobileApplication/Android или applicationType=MobileApplication/iOS в зависимости от системы
menuList=Profile
includeContent=true

Список стран, в которые возможна доставка, нужно получить с помощью метода: http://docs.otapi.net/ru/Documentations/Method?name=GetDeliveryCountryInfoList

Список городов нужно получать методом: http://docs.otapi.net/ru/Documentations/Method?name=SearchCities (пока пользователь вводит символы для поиска города, т.е. параметр QueryText пуст, нужно отправить IsMainCity=true для получения списка городов по умолчанию). В момент когда пользователь выбрал город из списка, сохраняем в профиле City и CityCode и автоматически заполняем поле Region.

Важно: CountryCode - обязательное поле для оформления заказа, CityCode - не обязательное поле, если пользователь выбрал город из списка, то система должна передать код в сервисы, если же пользователь ввёл город, которого нет в списке SearchCities, сохраняем только узел City.

Список заказов

http://docs.otapi.net/ru/Documentations/Method?name=SearchOrdersForUser

Активные: <OrderSearchParametersForUser><IsCancelled>false</IsCancelled><IsCompleted>false</IsCompleted></OrderSearchParametersForUser>

Отмененные: <OrderSearchParametersForUser><IsCancelled>true</IsCancelled></OrderSearchParametersForUser>

Закрытые: <OrderSearchParametersForUser><IsCompleted>true</IsCompleted></OrderSearchParametersForUser>

Ожидающие действий со стороны пользователя (оплата/доплата или подтверждение цены): <OrderSearchParametersForUser><IsAwaitingUser>true</IsAwaitingUser></OrderSearchParametersForUser>

Дизайном предусмотрено три группы заказов:

1) Ожидающие оплаты: <OrderSearchParametersForUser><IsAwaitingUser>true</IsAwaitingUser></OrderSearchParametersForUser>
2) Доставки: <OrderSearchParametersForUser><IsAwaitingUser>false</IsAwaitingUser><IsCancelled>false</IsCancelled><IsCompleted>false</IsCompleted></OrderSearchParametersForUser>
3) Покупки (иконка самолет), <OrderSearchParametersForUser><IsCompleted>true</IsCompleted><IsCancelled>true</IsCancelled></OrderSearchParametersForUser>

Список полей для отображения:

ID заказа, который передается в API во все возможные методы: Id
Номер заказа, отображаемый покупателю: DisplayId
Дата заказа: CreatedDateTime
Статус: StatusName
Сумма: TotalAmount
Уже оплачено: TotalAmount минус RemainAmount
К оплате: RemainAmount
Стоимость товаров: GoodsAmount
Позиций товаров: такой информации нет - не показываем это поле

Заказ. Подробная информация

Получить информацию о заказе: http://docs.otapi.net/ru/Documentations/Method?name=GetSalesOrderDetails

Отображаем для пользователя информацию о заказе (узел OrderInfo, подробнее здесь http://docs.otapi.net/ru/Documentations/Type?name=OtapiOrderInfo), особенности:

- DisplayId - идентификатор заказа.
- Id - идентификатор заказа, покупателю показывать не надо, этот Id передается в другие методы OTApi.
- CanCancel - если заказ можно отменить, показываем кнопку "Отменить заказ" - http://docs.otapi.net/ru/Documentations/Method?name=CancelSalesOrder
- Для отображения состояния заказа (выбор цветовой гаммы и т.п.), нужно использовать флаги: IsPaid, IsCancelled, IsCompleted

Отображаем список товаров в заказе (узел SalesLinesList, подробнее о каждой строке здесь http://docs.otapi.net/ru/Documentations/Type?name=SalesLine)

- Id - идентификатор строки заказа, этот Id передается в другие методы OTApi.
- CanBeClosed - возможность отметить что товар получен - http://docs.otapi.net/ru/Documentations/Method?name=CloseLinesOrder
- CanBeCancelled - возможность отменить строку - http://docs.otapi.net/ru/Documentations/Method?name=CancelLinesOrder
- CanBeConfirmed - возможность подтвердить строку - http://docs.otapi.net/ru/Documentations/Method?name=ConfirmPriceLinesOrder
- необходима возможность массового удаления/подтверждения строк заказа.

Список посылок - http://docs.otapi.net/ru/Documentations/Method?name=GetSalesOrderShippings

Заказ. Оформление заказа и выбор способа доставки

В один заказ можно оформить товары только одного провайдера.

При оформлении заказа, потребуется форма выбора адреса доставки или, если адрес ранее на создавался, форма создания адреса/ов доставки (якорь на документацию по профилям доставки).

При оформлении заказа, потребуется выбрать способ доставки из предоставленного списка: запрос http://docs.otapi.net/ru/Documentations/Method?name=SearchDeliveryModes , параметры xmlSearchParameters:

- ProviderType - ид провайдера;
- Weight - вес товаров, в случае оформления из корзины, потребуется получить список товаров методом http://docs.otapi.net/ru/Documentations/Method?name=GetPartialBasket , получить сумму веса каждой позиции (для каждого товара нужно умножить вес товара на заказываемое количество);
- CountryCode - код страны доставки из выбранного адреса доставки;
- CityCode - код города доставки из выбранного адреса доставки.

Пример параметров: <DeliveryModeSearchParameters><CountryCode>RU</CountryCode><CityCode>7700000000000</CityCode><Weight>75.000</Weight><ProviderType>Taobao</ProviderType></DeliveryModeSearchParameters>.

При отображении способов доставки, необходимо учесть флаг IsPickupPointMode.

- Если флаг = false, показываем пользователю поля Адрес доставки и Индекс (значения полей берем из выбранного адреса доставки).
- Если флаг = true, Адрес доставки и Индекс берем из выбранного пункта выдачи. Предоставляем пользователю интерфейс по выбору пункта выдачи: показываем все доступные пункты выдачи для текущего способа доставки http://docs.otapi.net/ru/Documentations/Method?name=SearchDeliveryPickupPoints (по умолчанию должен быть выбран пункт выдачи, который сохранен в выбранном адресе/профиле доставки).

После успешного оформления заказа, нужно сохранить ExternalDeliveryId в предпочтения пользователя (http://docs.otapi.net/ru/Documentations/Method?name=UpdateUserPreferences) , а так же обновить параметры адреса доставки, если интерфейс сразу этого не сделал (http://docs.otapi.net/ru/Documentations/Method?name=UpdateUserProfile).

Из корзины

Перед тем как оформить заказ из корзины, необходимо "проверить" актуальность цен и возможность оформления:

- после того как клиент выбрал галочками товары которые он хочет оформлять, вызываем метод http://docs.otapi.net/ru/Documentations/Method?name=RunBasketChecking с выбранными идентификаторами корзины, в ответ получаем и запоминаем "идентификатор активности"
- показываем клиенту оверлей, фразу "Пожалуйста, подождите. Товары в корзине проходят проверку на наличие. Время проверки зависит от количества товаров в корзине" и прогресс бар
- вызываем функцию http://docs.otapi.net/ru/Documentations/Method?name=GetBasketCheckingResult передав в нее идентификатор активности
    - если в ответе нет ошибки и IsFinished != true, показываем прогресс клиенту из узла ProgressPercent и через одну секунду еще раз вызываем метод GetBasketCheckingResult
    - из узла Messages для нужного элемента корзины показываем текст и статус (статус отображаем путем изменения цвета строки, Ok - зеленый, Warning - желтый, Error - красный)
    - если IsFinished == true, скрываем прогресс бар и
        - если в Messages есть хотя бы один Error - проматываем экран до первой ошибки и предоставляем пользователю возможность исправить ошибку (например выбрать другое количество товаров и по новой запустить проверку корзины)
        - если ошибок нет, но есть хотя бы один Warning - проматываем экран до первого предупреждения и предоставляем пользователю возможность повторно нажать на кнопку "оформить заказ", повторное нажатие должно сразу начать процедуру оформления заказа без повторной проверки корзины (важно: если пользователь изменил количество, конфигурацию или выбрал галочками другие товары - процесс проверки корзины должен начинаться заново)
        - если нет ни ошибок ни предупреждений, сразу открываем экран оформления заказа.

Важно при каждом изменении выбранных товаров запускать проверку корзины заново, а так же заново показывать сообщения об ошибке/успехе от сервисов.

Заказ можно оформить "из корзины", для этого нужно вызвать метод http://docs.otapi.net/ru/Documentations/Method?name=CreateOrder , передав в него:

- элементы корзины (якорь на документацию по корзине)
- способ доставки
- профиль пользователя
- не обязательно: можно предложить пользователю ввести комментарий к заказу.

Дозаказ

При оформлении заказа, нужно предложить пользователю возможность сделать дозаказ (если такая возможность у пользователя имеется). Получаем заказы, в которые можно сделать дозаказ, вызов http://docs.otapi.net/ru/Documentations/Method?name=SearchOrdersForUser с параметрами xmlSearchParameters=

<OrderSearchParametersForUser><ProviderType>ИдПровайдера(например: Taobao)</ProviderType><IsAvailableForRecreation>true</IsAvailableForRecreation></OrderSearchParametersForUser>. Если такие заказы есть, предлагаем покупателю выбрать: оформляет он новый заказ или делает дозаказ. Для дозаказа нужно вызвать метод http://docs.otapi.net/ru/Documentations/Method?name=RecreateOrder .

Быстрый заказ

Заказ можно оформить из карточки товара, кнопка "Быстрый заказ", для этого нужно вызвать метод http://docs.otapi.net/ru/Documentations/Method?name=AddOrder , передав в него:

- список оформляемых товаров (для быстрого заказа из карточки 1688, это несколько товаров разной конфигурации). Для каждого товара обязательно нужно передать ид товара, ид конфигурации и количество (остальные возможные параметры передавать не нужно)
- способ доставки
- профиль пользователя
- не обязательно: можно предложить пользователю ввести комментарий к заказу.

Оплата заказа и пополнение лицевого счета

Если заказ не оплачен, обязательно предлагаем пользователю его оплатить. Если на лицевом счете клиента есть средства - запрос http://docs.otapi.net/ru/Documentations/Method?name=GetAccountInfo , узел AvailableAmount (при отображении обязательно используем CurrencySign) , предлагаем пользователю кнопку, которая оплатит заказ с лицевого счета. На кнопке пишем сумму доступную для оплаты, это должна быть или RemainAmount из информации о заказе http://docs.otapi.net/ru/Documentations/Method?name=GetSalesOrderDetails (при условии что на лицевом счете больше чем RemainAmount) или AvailableAmount (при условии что на лицевом счете меньше чем RemainAmount).

При нажатии на кнопку вызываем метод http://docs.otapi.net/ru/Documentations/Method?name=PaymentPersonalAccount и обновляем страницу заказа.

Заказ можно оплатить и через платежные системы. Запрашиваем список доступных платежных систем: http://docs.otapi.net/ru/Documentations/Method?name=GetPaymentModes . Картинка ПС - AbsoluteImageUrl, название ПС - Name. При клике на платежную систему, отображаем кнопку "Оплатить" и если узел CustomField == "Email" - отображаем поле ввода почты, если узел CustomField == "Phone" - отображаем поле ввода номера телефона.

При клике на кнопку Оплатить, вызываем метод http://docs.otapi.net/ru/Documentations/Method?name=GetPaymentParameters . Передаем параметры Amount (RemainAmount из информации о заказе), CurrencyCode, PaymentSystemId, OrderId, в SuccessUrl и FailUrl адреса возврата, которые приложение сможет перехватить. По полученному ответу формируем http-форму, открываем её в браузере, и отправляем её.:

- RequestUrl - url на который нужно отправить форму
- RequestMethod , если не пришел отправляем POST форму (иногда приходится отправлять GET, согласно параметру)
- IsNewWindow - нужно открыть форму в новом окне или в текущем, для мобильного приложения параметр видимо не актуален - всегда открываем запрос в новом окне браузера
- IsIFrame - вместо form отображаем в браузере iframe src={RequestUrl}?{Parameters вида param1=value1&param2=value2}
- IsImmmediate - пока игнорируем в мобильном приложении
- Parameters - список параметров платежа, которые надо обработать в зависимости от свойства IsUserData7
    - Если IsUserData==false, параметр нужно вставить в форму, для передачи в платежную систему.
    - Если IsUserData==true, параметр нужно просто вывести на экран.

Если в метод GetPaymentParameters не передать OrderId , то лицевой счет клиента пополнится на сумму Amount.

После успешного или неуспешного платежа, браузер вернется на один из переданных адресов, соответственно приложение должно их перехватить, закрыть браузер, и показать соответствующий экран.

Счет пользователя

Информацию о лицевом счете можно получить с помощью метода - http://docs.otapi.net/ru/Documentations/Method?name=GetAccountInfo

Формирование выписки по счету можно получить с помощью метода - http://docs.otapi.net/ru/Documentations/Method?name=GetStatement