Web-сервис для сборки заказов СМ

Назначение сервиса

Назначение web-сервиса сборки заказов СМ - обработка запросов от приложений сборки заказов, изменение состояний заказа и отправка уведомлений (нотификаций) в СМ. В качестве приложения для сборки заказа может выступать как Домино, так и отдельное приложение (в том числе на мобильном устройстве), которое поддерживает протокол обмена с сервисом сборки заказов. Сервис реализуется в виде Домино web-сервера, при необходимости масштабирования может быть развернут пул из нескольких однородных web-серверов с балансировкой нагрузки через nginx proxy.

Web-сервис работает с данными БД Домино от имени специального пользователя SberMarketOrdersApi.

Методы web-сервиса сборки заказов СМ

Все методы web- сервиса сборки заказов вызываются http(s) запросом POST. Приложения Домино могут получить адрес сервиса из параметра «Сервис сборки заказов СМ» карточки компании.

Адрес точки доступа для сервиса сборки заказов:

> https://компания.ru/sbermarket/orders/api/

Для аутентификации при вызове необходимо передавать авторизационный токен в заголовке запроса Client-Token. Приложения Домино могут получить токен из параметра «Авторизационный токен сервиса сборки заказов» карточки компании. Каждый метод сервиса должен проверить полученный токен, и вернуть http-код 403 forbidden в случае ошибки.

Имя вызываемого метода передается в URL. Параметры всегда передаются как content в формате json (требуется установить параметр заголовка Content-type : application/json). Json – объект с параметрами запроса имеет следующую структуру:

```
{
  requestId : <уникальный идентификатор запроса, string, не обязательный>,
  requestData : {
```

```
  <параметры запроса в формате json, в зависимости от вызванного
метода>
 }
}
```

Рекомендуется в каждом запросе передавать уникальный идентификатор запроса requestId. Это упростит анализ протоколов при разборе ошибок. В качестве идентификатора запроса проще всего использовать GUID.

Если метод выполнился, то http-кодом ответа всегда будет 200 OK, вне зависимости от результата выполнения метода. Иные коды ответа (4xx или 5xx) означают ошибки в работе сервиса, но не метода.

Результат выполнения метода всегда возвращается в виде json- объекта с фиксированной на верхнем уровне структурой:

```
{
 requestId : <уникальный идентификатор запроса, string, не
обязательный>,
 errorCode : <код ошибки (0-успешно), целое, обязательный>,
 errorMsg : <описание кода ошибки, string, обязательный при ошибке>,
 responseData : {
  <данные ответа в формате json, в зависимости от вызванного метода>
 }
}
```

Объект «Заказ» возвращается большинством методов сервиса сборки. Его структура:

```
order : {
    orderId : <Идентификатор заказа СМ, string>
    storeId : <Идентификатор магазина СМ, string>
    state : <Статус заказа, enum string>
    created : <Время поступления заказа, datetime>,
    collectAt : <Планируемое время сборки заказа, datetime>,
    deliveryAt : <Планируемое время передачи заказа в доставку, datetime>,
    collector : <Сборщик, string>,
    customer : {
      name : <Имя покупателя, string>,
      phoneNumber : <Телефон для связи с покупателем, string>,
      auxNumber : <Добавочный номер, string>
    },
    positions : [
      {
        productId : <Идентификатор товара (SKU), string>,
        replacedById : <Товар, на который заменена позиция, string>,
        name : <Наименование товара, string>,
```

```
      picture : <URL картинки с изображением товара, string>,
      storage : <Место хранения товара, string>,
      isWeight : <Весовой товар, bool>,
      isMarked : <Маркированный товар, bool>,
      orderedQuantity : <Заказанное количество, number>,
      agreedQuantity : <Согласованное количество, number>,
      collectedQuantity : <Собранное количество, number>,
      markingCodes : [
        <Код маркировки, string>,
        ...
      ]
    },
    ...
  ],
  comment : <Комментарий к заказу, string>,
  replacementPolicy : <Политика замены, string>
}
```

Комментарии по реквизитам заказа:

1. Тип datetime - дата и время в ISO формате, обычно YYYY-MM-DDTHH:MI:SS (время местное).
2. state, статус заказа. Возможные значения «Новый», «В сборке», «Собран, «Передан

курьеру», «Доставлен», «Отменен».

Для синхронизации состояния заказа между СМ и Домино сервис сборки заказов отправляет в СМ специальные уведомления. Спецификация API уведомлений СМ https://docs.sbermarket.ru/api-products/other/orders/partners-notifications .

Перечень уведомлений, которые сервис для сборки заказов отправляет в СМ

- in_work - заказ взят в работу (сборку)
- ready_for_delivery - сборка заказа завершена, заказ готов к передаче курьеру
- canceled - заказ отменен магазином

Сборщик заказа - это Пользователь в терминах Домино, для которого установлена роль «Сборщик заказов СМ». Сборщик во всех методах web- сервиса сборки заказов идентифицируется именем пользователя, которое в Домино уникально. Если заказ собирается без указания сборщика (т.н. сборка магазина), то поле «Сборщик» документа заказа СМ остается пустым (NULL). При вызове методов web- сервиса сборки заказов можно указывать специальное имя сборщика «Сборка магазина», либо null, либо вообще не указывать - все эти способы эквивалентны, если иное не указано явно. В возвращаемом заказе в случае сборки магазина поле order.collector всегда будет иметь значение null.

Метод getOrdersList. Получить список заказов указанного магазина

Метод возвращает список заказов по указанному магазину, которые удовлетворяют заданным условиям. Список заказов возвращается в виде массива объектов типа order без строк (без order.positions). Таким образом вызывающее приложение получает все реквизиты заказа без дополнительных запросов. Список заказов возвращается отсортированным по убыванию даты создания (самые новые сначала). Размер выдачи метода ограничен 1000 записей.

По умолчанию, если в запросе не указано никаких ограничений, метод возвращает все заказы по магазину, которые находятся незавершённом состоянии (т.е. кроме «Доставлен» и «Отменен»).

Параметр collector ограничивает выдачу метода заказами указанного сборщика. По умолчанию возвращаются только заказы в статусе «В сборке». При указании предопределённого сборщика «Сборка магазина» возвращаются заказы, не назначенные сборщику (т.н. сборка магазина без конкретизации сборщика).

Параметр states (массив статусов) ограничивает выдачу метода заказами, которые находятся в одном из указанных статусов.

Параметр createdBefore ограничивает выдачу заказами, созданными до указанной даты (включительно).

Параметр createdAfter ограничивает выдачу заказами, созданными после указанной даты (включительно).

Параметры pageSize/pageNumber задают размер и номер страницы выдачи. pageSize не может быть больше 1000 записей. Страничной выдачей следует пользоваться осторожно,

так как метод не хранит состояние между вызовами (является stateless), поэтому при параллельном изменении заказов некоторые из них могут не попасть в выдачу или попасть в нее дважды. Кроме того, попытка получить данные с большим номером страницы может потребовать от сервера много времени и ресурсов на предварительное пролистывание набора данных.

Параметры

> requestData : {
>
> storeId : <Идентификатор магазина СМ, обязательный, string>,
>
> collector : <Сборщик, не обязательный, string>,
>
> states : [<Массив статусов, не обязательный>],
>
> createdAfter : <Созданные после даты, не обязательный, datetime>,
>
> pageSize : <Размер страницы выдачи, integer>,
>
> pageNumber : <Номер страницы выдачи, integer>
>
> }

Возвращаемый ответ

> responseData : {
>
> endOfData : <Признак, что возвращены все записи, bool>,
>
> orders : [
>
> <Массив объектов типа order, без строк (order.positions)>
>
>]

```
    }
```

Метод getOrder. Получить содержимое заказа

Метод возвращает заказ по указанному магазину и номеру. Заказ возвращается в виде объекта типа order включая строки (order.positions).

Примечание: хотя номер (идентификатор) заказа СМ и является уникальным, указание идентификатора магазина при вызове метода обязательно.

Параметры

```
requestData : {
    storeId : <Идентификатор магазина СМ, обязательный, string>,
    orderId : <Идентификатор заказа СМ, обязательный, string>
  }
```

Возвращаемый ответ

```
responseData : {
    order : {
        <Заказ в виде объекта типа order, включая строки (order.positions)>
    }
  }
```

Метод collectOrder. Начать сборку заказа

Метод переводит заказ, находящийся в статусе «Новый» в состояние «В сборке», назначает сборщика на заказ, отправляет в СМ уведомление order.in_work, создаёт телеграмм-уведомление для сборщика и магазина о назначении заказа в сборку. Возвращает обновлённое содержание заказа, включая строки (order.positions).

Алгоритм:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа
- находит сборщика, если указан
- проверяет, что для сборщика активна смена по указанному магазину
- отправляет уведомление order.in_work в СМ
- изменяет в заказе статус на «В сборке», записывает сборщика и дату-время начала сборки
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- записывает в очередь телеграмм-уведомлений сообщения «Начата сборка заказа NN, сборщик XX, завершить до HH:MM» для сборщика и магазина,
- возвращает обновлённое содержание заказа

Параметры

> requestData : {

```
  storeId : <Идентификатор магазина СМ, обязательный, string>,
  orderId : <Идентификатор заказа СМ, обязательный, string>,
  collector : <Сборщик, не обязательный, string>
}
```

Возвращаемый ответ

```
responseData : {
  order : {
    <Заказ в виде объекта типа order, включая строки (order.positions)>
  }
}
```

Метод completeOrder. Завершить сборку заказа

Метод проверяет полноту сборки заказа, после чего переводит заказ, находящийся в статусе «В сборке» в состояние «Собран», отправляет в СМ уведомление order.ready_for_delivery, создаёт телеграмм- уведомление для сборщика и магазина о завершении сборки заказа. Возвращает обновлённое содержание заказа, включая строки (order.positions).

Алгоритм:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа
- проверяет, что по всем позициям собранное количество совпадает с согласованным (по весовому товару допустимо отклонение 10% в обе стороны)
- отправляет уведомление order.ready_for_delivery в СМ вместе с обновлённым составом заказа,
- изменяет в заказе статус на «Собран», записывает дату-время завершения сборки
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- записывает в очередь телеграмм-уведомлений сообщения «Завершена сборка заказа NN, сборщик XX» для сборщика и магазина,
- возвращает обновлённое содержание заказа

Параметры

> “

```
requestData : {
 storeId : <Идентификатор магазина СМ, обязательный, string>,
 orderId : <Идентификатор заказа СМ, обязательный, string>
}
```

Возвращаемый ответ

```
responseData : {
 order : {
    <Заказ в виде объекта типа order, включая строки (order.positions)>
 }
}
```

Метод cancelOrder. Отменить заказ

Метод проверяет статус заказа, после чего переводит его в состояние «Отменен», отправляет в СМ уведомление order.cancelled с указанием причины, создаёт телеграмм-уведомление для сборщика и магазина об отмене заказа. Возвращает обновлённое содержание заказа, включая строки (order.positions).

Алгоритм:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа
- отправляет уведомление order.cancelled в СМ с указанием причины отмены,
- изменяет в заказе статус на «Отменен», записывает дату-время отмены заказа
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- записывает в очередь телеграмм-уведомлений сообщения «Отменен заказ NN, сборщик XX, причина …» для сборщика и магазина,
- возвращает обновлённое содержание заказа

Параметры

> requestData : {
>
> storeId : <Идентификатор магазина СМ, обязательный, string>,
>
> orderId : <Идентификатор заказа СМ, обязательный, string>

```
    cancelReason : <Причина отмены, string>

  }
```

Возвращаемый ответ

```
responseData : {

    order : {

        <Заказ в виде объекта типа order, включая строки (order.positions)>

    }

  }
```

Методы работы с позицией заказа xxxPosition

Группа методов работы с позицией заказа (collectPosition, changePosition, appendPosition, replacePosition, clearPosition) обеспечивает собственно процесс сборки товаров. Все эти методы используют одинаковую схему работы с заказом (алгоритм) и все они в качестве результата возвращают обновлённое содержание заказа, включая строки (order.positions).

Алгоритм (каркас) для всех методов:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа, должен быть «В сборке»,
- изменяет строку заказа в соответствии со сценарием конкретного метода,
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции,
- фиксирует транзакцию,
- возвращает обновлённое содержание заказа

Возвращаемый ответ для всех методов:

> responseData : {
>
> order : {
>
> <Заказ в виде объекта типа order, включая строки (order.positions)>

```
    }

  }
```

Метод collectPosition. Сборка позиции

Метод сборки товара, который есть в заказе, в рамках согласованного количества.

Передаются код товара (ШК, КМ или SKU) и (опционально) собранное количество. По коду товара находится товар в Домино и позиция (строка) в заказе. Если товар весовой, то код товара должен содержать вес, либо вес должен быть передан отдельно как собранное количество. Для штучного товара, если собранное количество не передано, предполагается 1 шт. Если код товара является кодом упаковки, то собранное количество умножается на мультипликатор (вложимость) упаковки. Если товар маркированный, то код товара должен быть кодом маркировки, при этом собранное количество не должно передаваться. Для весовых маркированных товаров (в заводской фасовке с маркировкой) вес должен передаваться тегом (AI) 310 кода маркировки. Весовые маркированные товары собственной фасовки должны идентифицироваться и обрабатываться как немаркированные. Увеличивает собранное количество в строке заказа, при этом проверяется, чтобы собранное количество не превысило согласованное (для весового товара допускается превышение в 10%). Если товар маркированный, то переданный код маркировки сохраняется в дочерней строке специального типа.

Параметры

> requestData : {
>
> storeId : <Идентификатор магазина СМ, обязательный, string>,
>
> orderId : <Идентификатор заказа СМ, обязательный, string>
>
> productCode : <Код товара (ШК, КМ или SKU), обязательный, string >

```
  collectedQuantity : <Собранное количество, не обязательный, number>,

}
```

Метод changePosition. Изменение согласованного количества по позиции

Метод позволяет изменить согласованное количество для позиции заказа. Передается идентификатор товара (SKU) и новое согласованное количество. По идентификатору товара находится позиция в заказе. Проверяется, чтобы новое значение для согласованного количества не было меньше уже собранного количества в строке заказа. При увеличении согласованного количества проверяется, чтобы общий вес и цена заказа не превысили установленных лимитов. Устанавливается новое значение для согласованного количества. Если установить согласованное количество равным 0, то это будет интерпретироваться как «вычерк», т.е. исключение позиции из заказа по согласованию с покупателем.

Параметры

```
requestData : {
    storeId : <Идентификатор магазина CM, обязательный, string>,
    orderId : <Идентификатор заказа CM, обязательный, string>
    productId : <Идентификатор товара (SKU), обязательный, string>
    agreedQuantity : <Согласованное количество, обязательный, number>,
  }
```

Метод appendPosition. Добавление новой позиции

Метод сборки товара, которого нет в заказе, с одновременным добавлением позиции в заказ. Передается код товара (ШК, КМ или SKU), согласованное количество и (опционально) собранное количество. По коду товара находится товар в Домино, проверяется его вхождение в ассортимент СМ. Проверяется, что такая позиция отсутствует в заказе. Аналогично методу changePosition проверяется, чтобы общий вес и цена заказа не превысили установленных лимитов. Создается строка заказа с указанным товаром, согласованным количеством и заказанным количеством равным 0. Далее логика такая же, как при сборке существующей позиции методом collectPosition.

Параметры

```
requestData : {
    storeId : <Идентификатор магазина СМ, обязательный, string>,
    orderId : <Идентификатор заказа СМ, обязательный, string>
    productCode : <Код товара (ШК, КМ или SKU), обязательный, string >
    agreedQuantity : <Согласованное количество, обязательный, number>,
    collectedQuantity : <Собранное количество, не обязательный, number>,
  }
```

Метод replacePosition. Замена позиции

Замена позиции в СМ является особым случаем, когда некоторый товар из заказа отсутствует и полностью заменяется при сборке на другой (аналогичный) товар. В заменяемой позиции согласованное количество устанавливается равным 0 (как при вычерке), и указывается идентификатор товара (SKU), на который произведена замена. Заменяющий товар может как присутствовать в заказе изначально, так и быть добавлен в процессе сборки. При этом покупатель будет информирован, на какой именно товар была заменена отсутствующая позиция.

В метод передаются идентификатор заменяемого товара, код заменяющего товара (ШК, КМ или SKU), согласованное количество и (опционально) собранное количество. По идентификатору заменяемого товара находится позиция заказа и проверяется, чтобы заказанное количество в ней было больше 0, а собранное равно 0. Далее по коду заменяющего товара находится товар в Домино, и проверяется, есть ли такая позиция в заказе. Если такая позиция есть, то согласованное количество в ней увеличивается на переданное в параметрах метода значение (с учётом ограничений по весу и стоимости заказа), после чего логика такая же, как при сборке существующей позиции методом collectPosition. Если же позиция в заказе отсутствует, то она добавляется аналогично методу appendPosition. После всех этих действий в строке с заменяемым товаром согласованное количество устанавливается равным 0, и сохраняется идентификатор заменяющего товара.

Параметры

> requestData : {
>
> storeId : <Идентификатор магазина СМ, обязательный, string>,

```
  orderId : <Идентификатор заказа СМ, обязательный, string>

  replacingProductId : <Идентификатор заменяемого товара (SKU),
обязательный, string>,

  productCode : <Код заменяющего товара (ШК, КМ или SKU),
обязательный, string >

  agreedQuantity : <Согласованное количество, не обязательный, number>,

  collectedQuantity : <Собранное количество, не обязательный, number>,

}
```

Метод clearPosition.Удалить информацию о сборке позиции

Метод удаляет из заказа информацию о сборке конкретной позиции: обнуляет собранное количество и удаляет информацию о кодах маркировки. Согласованное с покупателем количество не изменяется. Если эта позиция является заменой для других позиций, то эта информация так же не удаляется. Метод применяется для повторного пересчета собранных товаров по позиции - например в том случае, когда нужно заменить уже отобранный маркированный товар.

Параметры

```
requestData : {
    storeId : <Идентификатор магазина CM, обязательный, string>,
    orderId : <Идентификатор заказа CM, обязательный, string>
    productId : <Идентификатор товара (SKU), обязательный, string>
  }
```

Метод clearAllPositions. Удалить из заказа всю информацию по сборке

Метод удаляет из заказа всю информацию о сборке по всем позициям, включая данные по согласованным заменам и количествам. Заказ приводится в исходное состояние, как если бы он только что поступил из СМ и был переведён в состояние «В сборке». Возвращает обновлённое содержание заказа, включая строки (order.positions).

Алгоритм:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа, должен быть «В сборке»
- удаляет из заказа все дочерние строки с кодами маркировки,
- удаляет из заказа все товарные строки, где заказанное количество равно 0,
- прописывает во все оставшиеся товарные строки согласованное количество равное заказанному количеству, собранное количество равное 0, очищает ссылку на позицию-замену,
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- возвращает обновлённое содержание заказа

Параметры

> “

```
requestData : {
 storeId : <Идентификатор магазина СМ, обязательный, string>,
 orderId : <Идентификатор заказа СМ, обязательный, string>
}
```

Возвращаемый ответ

```
responseData : {
 order : {
    <Заказ в виде объекта типа order, включая строки (order.positions)>
 }
}
```

Метод assignCollector. Назначить сборщика на заказ

Метод назначает нового сборщика на заказ, уже находящийся в состоянии «В сборке», создаёт телеграмм- уведомление о передаче заказа для старого и нового сборщиков и магазина (применимо и для перевода заказа с магазина на конкретного сборщика, и со сборщика на магазин). Новый сборщик должен иметь открытую смену по магазину. Возвращает обновлённое содержание заказа, включая строки (order.positions).

Алгоритм:

- открывает транзакцию,
- находит заказ по идентификатору подразделения и заказа, читает его с блокировкой (for update),
- проверяет текущий статус заказа, должен быть «В сборке»
- находит нового сборщика (если это передача другому сборщику, а не магазину),
- проверяет, что для сборщика активна смена по указанному магазину, и что это не передача заказа самому себе,
- записывает нового сборщика в заказ,
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- записывает в очередь телеграмм-уведомлений сообщения «Заказ NN передан сборщику XX, завершить сборку до HH:MM» для обоих сборщиков и магазина,
- возвращает обновлённое содержание заказа

Параметры

```
requestData : {
    storeId : <Идентификатор магазина СМ, обязательный, string>,
    orderId : <Идентификатор заказа СМ, обязательный, string>
    collector : <Сборщик, не обязательный, string>,
  }
```

Возвращаемый ответ

```
responseData : {
    order : {
       <Заказ в виде объекта типа order, включая строки (order.positions)>
    }
  }
```

Метод beginSession. Начать смену сборщика

Метод начинает смену сборщика с указанием её планируемой продолжительности. Для начала новой смены у сборщика не должно быть открытых смен. Создаёт телеграмм-уведомление сборщику и магазину.

Алгоритм:

- находит магазин и сборщика, проверяет продолжительность (диапазон значений от 1 до 24 часов),
- создаёт пустой документ управления сменами за текущий день, если таковой отсутствует,
- открывает транзакцию,
- блокирует от изменения документ управления сменами за текущий день,
- по документам управления сменами за последние 3 дня проверяет, чтобы у сборщика не было открытой смены,
- добавляет строку смены в документ управления сменами за текущий день, в строке заполняются Магазин, Сборщик, текущая дата как Дата начала смены, Планируемая дата окончания смены,
- записывает в очередь телеграмм-уведомлений сообщения «Сборщик XX начал смену в магазине MM в hh:nn, планируемое время завершения смены dd-mm-yyyy hh:nn» для сборщика и магазина,
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- возвращает код завершения и сообщение об ошибке (если ошибка)

Параметры

```
requestData : {

 storeId : <Идентификатор магазина СМ, обязательный, string>,

 collector : <Сборщик, обязательный, string>,

 duration : <Продолжительность смены, часов, обязательный, [1..24],
integer>

}
```

Радел responceData в возвращаемом ответе отсутствует. Возвращается только стандартный заголовок с кодом и сообщением об ошибке.

Метод endSession. Завершить смену сборщика

Метод завершает смену сборщика, записывая в строку учёта фактическое время завершения смены. Для завершения смены сборщик не должен иметь назначенных заказов со статусом «В сборке». Создаёт телеграмм- уведомление сборщику и магазину.

Алгоритм:

- находит магазин и сборщика,
- открывает транзакцию,
- находит открытую смену по сборщику, анализируя документы управления сменами за последние 3 дня,
- блокирует от изменения соответствующий документ управления сменами,
- проверяет Сборщика по Заказам СМ по Магазину в статусе «В сборке» за последние 3 дня,
- в строке смены заполняет текущей датой поле Фактическая дата окончания смены,
- записывает в очередь телеграмм-уведомлений сообщения «Сборщик XX завершил смену в магазине MM в hh:nn, отработанное время hh:mm» для сборщика и магазина,
- если любой из перечисленных выше шагов завершился с ошибкой, то выполняет откат транзакции
- фиксирует транзакцию
- возвращает код завершения и сообщение об ошибке (если ошибка)

Параметры

> “

```
requestData : {
  storeId : <Идентификатор магазина СМ, обязательный, string>,
  collector : <Сборщик, обязательный, string>
}
```

Радел responceData в возвращаемом ответе отсутствует. Возвращается только стандартный заголовок с кодом и сообщением об ошибке.

Метод getSessions. Получить список активных смен по магазину

Метод возвращает список активных смен (сборщиков) по указанному магазину. Смены подбираются по строкам документов учета рабочих смен сборщиков за последние 3 дня. Так как плановая продолжительность смены ограничена 24 часами, то этого достаточно. Смена активна, если в строке не проставлена фактическая дата завершения смены. Если в запросе указать конкретного сборщика, то в ответе будет только смена для этого сборщика, если она активна, и пустой массив в противном случае.

Параметры

```
requestData : {
    storeId : <Идентификатор магазина СМ, обязательный, string>
    collector : <Сборщик, не обязательный, string>
  }
```

Возвращаемый ответ

```
responseData : {
    sessions : [
```

```
    {
      collector : <Сборщик, string>,
      startAt : <Дата и время начала смены, datetime>,
      finishAt : <Планируемые дата и время окончания смены, datetime>,
      orders : <Число назначенных заказов в сборке, integer>
    },
    ...
  ]
}
```