

Разработка

- Гайдлайны
 - Общая концепция разработки сервиса
 - Обработка ошибок
 - Опциональные значения в моделях
- Репозитории
 - Хранилища 1С
- Расширение описания API
 - Дерево условий скидок
 - Двухфакторная авторизация операций
- Система ссылок

Гайдлайны

Внутренние гайдлайны разработки

Общая концепция разработки сервиса

Общая архитектура

Сервис разрабатывается в слоеной модели. Код содержит три основных слоя в виде пакетов:

- **controller** - слой определяет взаимодействие с сервером fiber:
 - Разбирают запросы пользователей
 - Вызывает функции валидации объектов предметной области
 - Содержит документацию к предоставляемому API
 - Делегирует обработку данных слою **usecase**
- **usecase** - слой бизнес-логики.
 - Содержит типовые кейсы взаимодействия с системой
 - Определяет логику сервера приложений при взаимодействии с системой
 - Делегирует ввод/вывод слою **storage**
- **storage** - слой хранения данных
 - Содержит логику ввода/вывода в конкретные хранилища данных

Отдельно стоит пакет **domain**, который:

- Описывает структуры данных предметной области
- Логику их взаимодействия со всеми слоями системы
- Логику конвертации объектов предметной области в другие объекты предметной области
- Валидацию объектов предметной области

Этот пакет используется всеми компонентами системы.

Обработка ошибок

Общие требования

Для работы с ошибками требуется использовать вместо стандартного пакета **errors**, пакет **github.com/pkg/errors**, так как он позволяет учитывать стек вызова, при логировании ошибки.

В любых операциях на уровне выше слоя **storage**, нужно использовать внутренний пакет **apierrors**, для формирования ошибок выдаваемых пользователю. Т.е. на слоях **controller** и **usecase**, а так же в пакете **domain** нужно всегда возвращать соответствующую ошибку из пакета **apierrors**. Если из слоя ниже или другой библиотеки была получена ошибка другого типа, её следует обернуть в ошибку из пакета **apierrors**.

Если соответствующей ошибки нет, ее можно добавить, согласовав назначение нового класса ошибок. Не следует делать специализированные классы ошибок, класс должен отражать широкое множество ошибок схожих по области в которой они возникли. Например, **ValidationError** - класс, отражающий любые ошибки валидации, не следует создавать **MyStructFieldValidationError**, дабы не плодить сущности.

Централизованная обработка ошибок

Условно ошибки можно разделить на два вида:

- Обработанная - ошибка была завернута в **apierror**, значит разработчик предусмотрел её обработку и возврат пользователю.
- Внештатная - любая не обработанная разработчиком ошибка.

Сервис обрабатывает ошибки в едином обработчике, следующим образом:

- Если в обработчик пришла ошибка типа **apierror**, она выводится в лог, если обернутая в нее ошибка поддерживает получение стека-вызовов - стек также запишется в лог. Если не поддерживает, в логе будет только сообщение об ошибке. В логе ошибка помечается признаком **severity=handled**.

- Если пришла ошибка веб-сервера fiber - она записывается в лог с **severity=fiber**.
- В остальных случаях, обработчик считает, что ошибка была не обработана и она записывается в лог с **severity=unhandled**.

При любом варианте записи в лог вызывающему отдается ответ с классом ошибки и причиной её возникновения. Ответ выглядит следующим образом:

```
{
  "success": false,
  "message": "Внутренняя ошибка сервиса",
  "error": {
    "class": "internal",
    "type": "runtime.errorString",
    "reason": "runtime error: integer divide by zero"
  }
}
```

Класс ошибки определяет важность для клиентского приложения, сообщение описывает проблему, объект error показывает причину возникновения ошибки.

Соответствие внутренней и внешней классификации ошибок:

	Класс ошибки apierror
handled	Определяется разработчиком
unhandled	internal
fiber	internal

Опциональные значения в моделях

Цель

В некоторых случаях передача всех полей модели от клиента не имеет смысла. Например, клиент хочет обновить только наименование какой-то сущности. Необходимо предоставить простой программный интерфейс для объявления таких полей, получения значений только заполненных полей перед выполнением запроса к базе данных.

Пакет

gitlab.corp.rarus.cloud/rarus-lms/backend/pkg/optional

Пакет предоставляет обобщенный тип `optional.Optional[T any]`, который оборачивает другой тип и хранит указатель на значение этого типа. Таким образом можно легко различить наличие и отсутствие значения в опциональном поле.

Например, имеем модель клиента, у которого есть необязательное для передачи поле `age`:

```
type Client struct {  
    Id      uint64           `json: "id" `  
    Name    string          `json: "name" `  
    Age     optional.Optional[uint8] `json: "age" `  
}
```

При входном JSON отсутствующим полем `age`, поле не имеет значения:

```
jsonData := []byte(`{  
    "id": 1,  
    "name": "Василий",  
  }`)  
c := Client{}
```

```

json.Unmarshal(jsonData, &c) // Обертка использует парсер внутреннего типа

c.Age.IsSet() // если значение не установлено - false

v, err := c.Age.GetValue() //если значение == nil, возвращает error

c.Age.Value // указатель на внутренний тип, *uint8 в данном случае

v = c.Age.Default(30) // если значение не установлено, вернет значение по-умолчанию
// v == 30, но c.Age.IsSet() == false

c.Age.SetValue(30) // Теперь значение изменилось и изменения будут учтены при получении
map[string]any

```

Для записи в базу не фиксированного набора значений, обычно нужно получить ключи и значения из структуры. Для этого можно использовать функции:

```

jsonData := []byte(`{
  "id": 1,
  "name": "Василий",
}`)
c := Client{}
json.Unmarshal(jsonData, &c)

m := optional.StructToMap(c) // здесь будет map[string]any, но содержащая только обязательные
поля и Optional поля с заполненными значениями
// m = {"id":1, "name":"Василий"}

k, v := optional.GetKeysAndValues(m) // далее можно получить ключи и значения, для передачи в
аргументы драйвера к базе
// k = ["id", "name"]
// v = [1, "Василий"]

squirrel.Update("some_table").Columns(k).Values(v).Where(...)

```

Функция `optional.StructToMap` формирует ключи хэш-таблицы по следующим правилам:

- Приватные поля всегда игнорируются
- Если у поля не указаны тэги - оно игнорируется
- Если у поля указан тэг ``db: "some_name"``, ключ берется из него
- Если у поля указан тэг ``json: "some_name"``, но нет тэга `db`, ключ берется из него

Репозитории

Хранилища 1С

Хранилища расположены на серверах Севастопольского филиала 1С-Парус. Доступны только по протоколу tcp. Внутренний адрес серверов хранилищ **192.168.38.5**, внешний **178.34.183.66**. По внутреннему адресу хранилища доступны внутри сети 1С-Парус(в т. ч. через VPN). По внешнему - доступны в публичной сети.

Система развертывания клиентов

Версия платформы 1С-Предприятие	8.3.25.1286
Внутренний адрес	tcp://192.168.38.5:1286/LMSManagement
Внешний адрес	tcp://178.34.183.66:1286/LMSManagement

Расширение описания API

Расширение описание документации сервиса.

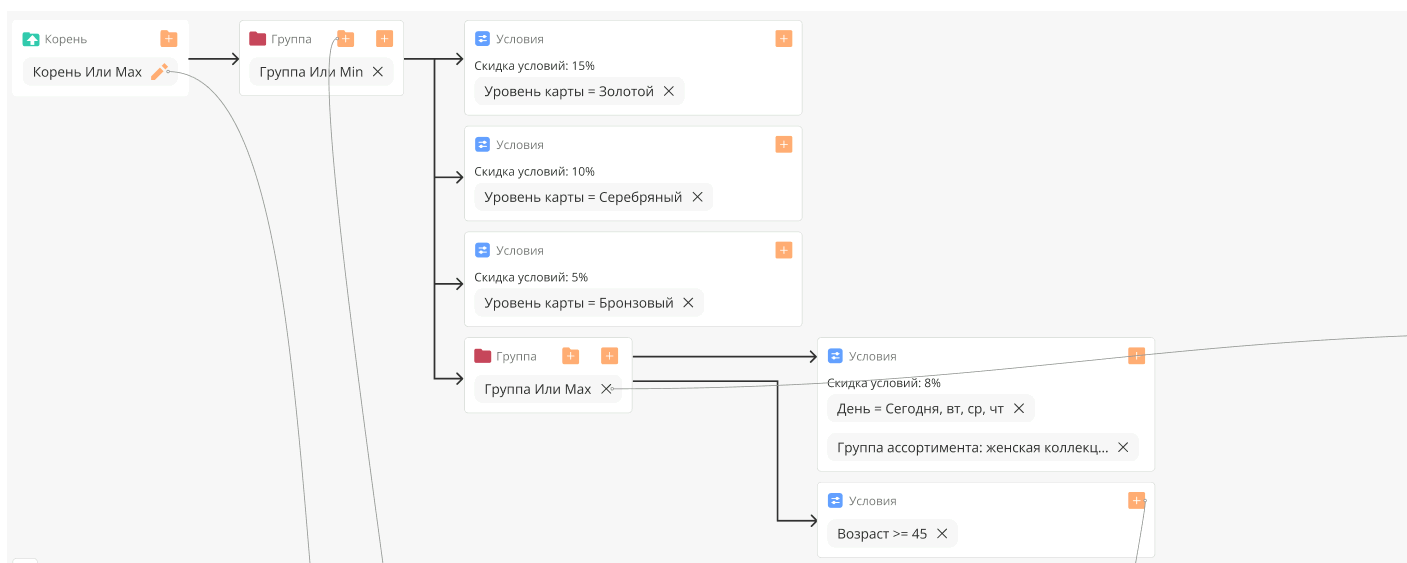
Дерево условий скидок

Передача и хранение

Дерево условий скидок хранится отдельно для каждой скидки. Дерево представляет собой рекуррентную структуру, состоящую из узлов трех типов:

- Группа условий
- Контейнер условий скидки
- Условие скидки

Начальным корневым узлом структуры может быть только группа условий. Внутри группы могут находиться только контейнеры условий, а уже в контейнерах условий содержатся массивы условий.



Дерево описывается вложенными узлами, с указанием соответствующего типа объекта:

```
"type": "group", // Определяет тип объекта
"container": {
  // Состав полей зависит от типа объекта (см. ниже)
}
```

Каждый объект имеет свою структуру полей. При разборе объектов сервис выполняет отложенный разбор поля `container`, в зависимости от типа указанного в поле `type`

Группа условий

Определяет взаимодействие вложенных результатов вычисленных условий.

```
{
  "type": "group",
  "container": {
    "rule": "max", // Правило взаимодействия значений скидки подчиненных элементов
    "operator": "or", // Правило взаимодействия срабатывания подчиненных элементов
    "items": [] // Подчиненные элементы - группы или контейнеры условий
  }
}
```

Контейнер условий

Определяет множество условий, значение скидки при срабатывании контейнера и правило взаимодействия сработавших скидок внутри контейнера.

```
{
  "type": "conditions",
  "container": {
    "rule": "and", // Правило взаимодействия условий
    "value": 1.5, // Значение скидки, переопределяющее основное значение
    "conditions": [] // Набор условий контейнера
  }
}
```

Условие

Определяет утверждение, при срабатывании которого условие считается выполненным.

```
{
  "type": "some-type", // Тип условия, от типа зависит состав полей объекта `container`
  "container": {} // Объект описывающий состав условия, зависит от типа условия(см. ниже)
}
```

Типы условий

Булево

Описывает простое булево условие, если значение в поле `operand` == true, условие считается всегда выполненным, в противном случае не выполненным.

```
{
  "type": "boolean",
  "container": {
    "operand": true
  }
}
```

Уровень карты

Выполняется, если уровень карты равен или не равен установленному

```
{
  "type": "card-level",
  "container": {
    // Оператор
    "operator": "==",
    // Идентификатор уровня карт
    "operand": 1
  }
}
```

Номер позиции

Выполняется, если номера позиций чека соответствуют условию

```
{
  "type": "row-number",
  "container": {
    // Оператор
    "operator": "==",
    // Номер строки
  }
}
```

```
    "operand": 1
  }
}
```

Дни недели

Выполняется, если дата документа выпадает на указанные в маске дни недели

```
{
  "type": "week-day",
  "container": {
    // Маска дней недели, каждый бит отвечает за день недели пн-вс слева направо
    "operand": "0010101"
  }
}
```

День рождения

Выполняется, если день рождения клиента попадает в диапазон дат от даты документа

```
{
  "type": "birthday",
  "container": {
    // Период перед днем рождения в днях, когда скидка может сработать
    "days_before": 2,
    // Период после дня рождения в днях, когда скидка может сработать
    "days_after": 2
  }
}
```

Сумма без учета скидки

Выполняется, если сумма без учета скидки соответствует условию. Возможно уточнение сегмента товаров, для которого условие будет проверяться

```
{
  "type": "sum-without-discounts",
  "container": {
    "area": "document|position",
    "operator": ">=",
    "operand": 100.50,
  }
}
```

```
{
  "segments": [1, 2, 3, 4]
}
```

Сумма с учетом скидки

Выполняется, если сумма с учетом скидок соответствует условию. Возможно уточнение сегмента товаров, для которого условие будет проверяться

```
{
  "type": "sum- with- discounts",
  "container": {
    "area": "document| position",
    "operator": ">=",
    "operand": 100.50,
    "segments": [1, 2, 3, 4]
  }
}
```

Количество товара

Выполняется, если количество товара соответствует условию. Возможно уточнение сегмента товаров, для которого условие будет проверяться

```
{
  "type": "quantity",
  "container": {
    "area": "document| position",
    "operator": ">=",
    "operand": 100.50,
    "segments": [1, 2, 3, 4]
  }
}
```

Первая покупка по карте

Выполняется, текущий документ первый по карте

```
{
  "type": "first- purchase",
  "container": {
```

```
{
  "operand": true
}
```

Сумма покупок по карте

Выполняется, если сумма покупок по карте соответствует условию

```
{
  "type": "card-purchase-sum",
  "container": {
    "operator": ">="
  },
  "operand": 100500.99
}
```

Группа ассортимента

Выполняется, если товар входит в группу ассортимента

```
{
  "type": "catalog-group",
  "container": {
    "operator": "in"
  },
  "operand": 42
}
```

Сумма покупок по карте за период

Выполняется, сумма покупок по карте за указанный период соответствует условию

```
{
  "type": "card-purchase-sum-over",
  "container": {
    "operator": ">",
    "operand": 50000,
    "step": "days|month|month-aligned|year|year-aligned|quarter-aligned",
    "steps_before": 1,
    "steps_after": 1
  }
}
```



```
}  
}
```

Комплект

Выполняется, если в документе есть комплекты

```
{  
  "type": "product-set",  
  "container": {  
    "price_sorting": "asc| desc",  
    "no_intersection": false,  
    "segments": [  
      {  
        "id": 1,  
        "calculate": true,  
        "distribute": true,  
        "quantity": 10  
      },  
    ],  
  }  
}
```

Группа промокодов

Выполняется, если промокод был передан и входит в группу

```
{  
  "type": "promo-group",  
  "container": {  
    "operand": 1 // Идентификатор группы промокодов  
  }  
}
```

Маска промокода

Выполняется, если промокод подходит под переданную маску. Возможно использование регулярных выражений.

```
{  
  "type": "promo-mask",
```

```
{
  "container": {
    "operand": "%PROMO-202%" // Маска, символы % будут интерпретироваться как 0 или более любых
    СИМВОЛОВ
  }
}
```

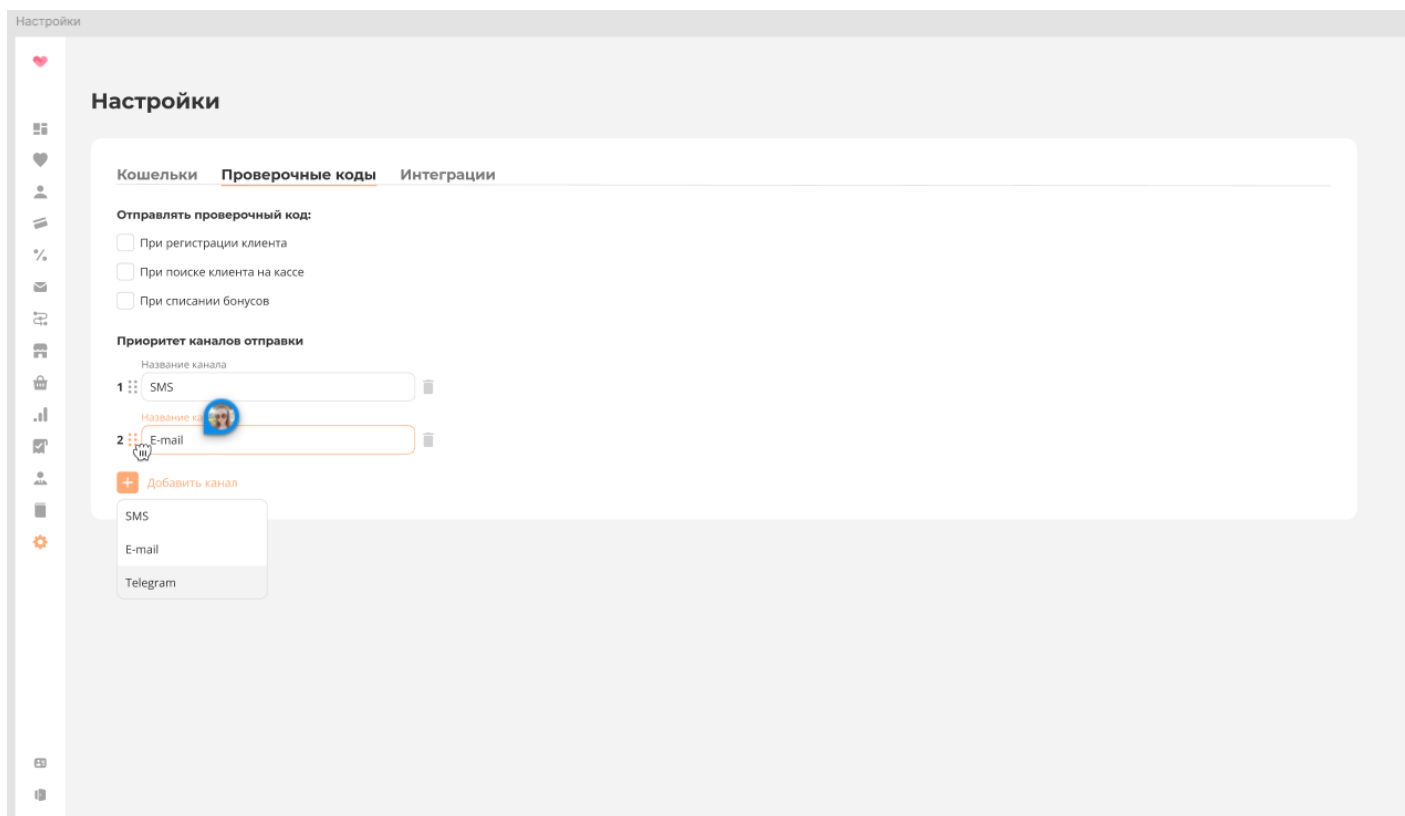
Двухфакторная авторизация операций

Описание механизма

Методы использующие двухфакторное подтверждение операций отмечены в документации красным тэгом **2FA**.

В качестве второго фактора выступает фактор владения покупателем частного ресурса(телефон, почта). Настройка 2FA выполняется в личном кабинете и состоит из двух частей:

- Галка для событий, при которых необходимо выполнить проверку второго фактора
- Список каналов которыми будет доставлен код подтверждения фактора
- В карточке магазина признак отключения проверки фактора, который совсем отключает механизм в конкретных магазинах



Каждое событие проверяется отдельно. Попытка отправки кода выполняется через канал в

порядке указания его в списке каналов, в зависимости от наличия канала в конкретном запросе или в профиле клиента.

Пример: Установлена настройка проверки фактора владения при регистрации клиента. Порядок каналов отправки: Телефон -> Почта. В запросе регистрации телефон не передан, но есть адрес электронной почты. При таких вводных отправка кода будет выполняться на почту, т.к. определить телефон не представляется возможным. При тех же настройках и выполнении запроса идентификации клиента, если в карточке клиента есть телефон, отправка произойдет на телефон, как более приоритетный канал отправки.

Описание протокола

Протокол построен на сессионной модели. Любой защищенный запрос(запрос требующий 2FA) при установленной настройке для соответствующей операции будет начинать сессию работы с пользователем в определенном магазине.

Если для защищенного метода установлена настройка и список каналов отправки не пустой, выполняется проверки:

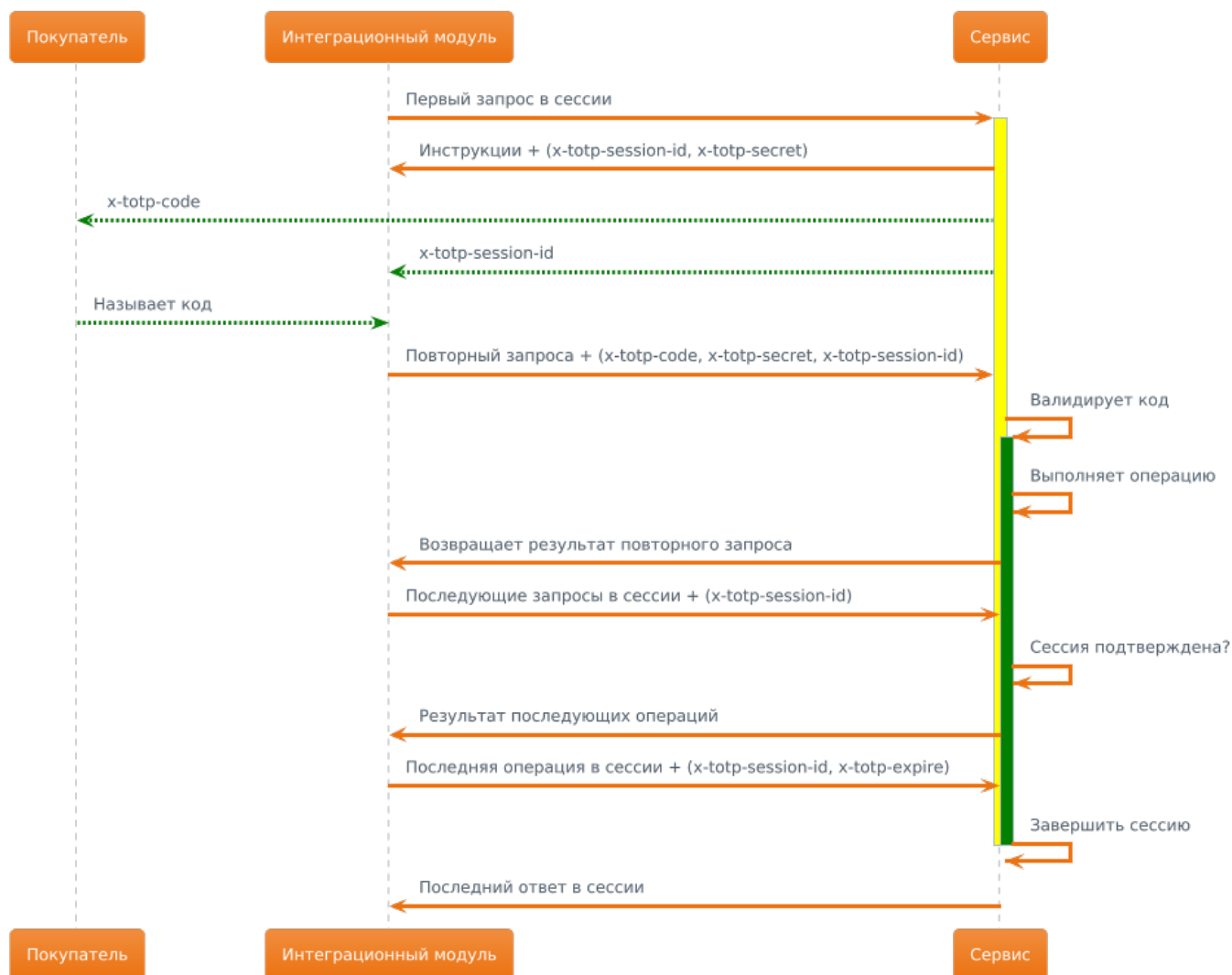
- Этот вызов является началом или продолжением сессии? Проверка выполняется по наличию специального заголовка `x-totp-session-id`.
- Если передан заголовок `x-totp-channel`, попытка отправки кода будет выполняться на этот канал
- Если заголовок `x-totp-session-id` не передан, начинается новая сессия, сервис генерирует и отправляет на первый доступный канал отправки временный код подтверждения фактора владения и отвечает специальными инструкциями для клиентского приложения:

```
{
  "success": true,
  "message": "OK",
  "data": {
    "session": {
      "id": "081ff88e5d1f925d33926fb0f580e49485fd231b", // Идентификатор сессии
      "issuer": "790300000001",
      "issuer_location": "",
      "confirmed": false,
      "created_at": null,
      "updated_at": null
    },
    "instruction": {
```

```
"channel": "phone", // Канал, на который выполнялась отправка кода подтверждения
"reciever": "79030000001", // Идентификатор получателя(зависит от канала)
"secret": "dummy-secret", // Секрет клиентского приложения, используется для проверки
введенного кода подтверждения
"duration": 120, // Время жизни кода подтверждения в секундах
"available_channels": [ // Доступные в текущем контексте каналы отправки кодов
подтверждения
    "phone"
]
}
}
```

- Если заголовок `x-totp-session-id` передан, считается, что это очередной вызов в рамках начатой ранее сессии
 - В таком случае выполняется проверка подтверждения фактора владения(было ли подтверждение) в рамках сессии.
 - Если сессия не была подтверждена ранее, выполняется проверка заголовков `x-totp-code` и `x-totp-secret`, если проверка прошла успешно - код верный, сессия помечается как подтвержденная и защищенный запрос возвращает свое тело ответа(зависит от конкретного запроса)
 - Если сессия уже была подтверждена - запрос просто возвращает свое тело ответа
- Клиентское приложение может завершить сессию любым запросом в сессии с передачей заголовка `x-totp-expire`.

Таким образом начинаться сессия может любым методом, подтверждаться любым методом, любой метод в цепочке вызовов после подтверждения больше не требует подтверждения.



Описание развертывания

Сервис использует переменные окружения связанные с отправкой SMS4B. Также используются две переменные окружения:

TOTP_SESSION_TTL_MIN

Время жизни сессии подтвержденной сессии в минутах. Не менее 10 минут, если указано значение меньше - сессия будет жить 10 минут. Значение по-умолчанию: 10 минут.

TOTP_SESSION_VACUUM_INTERVAL_MIN

Интервал, с которым сервис выполняет очистку просроченных сессий. Не менее 10 минут, если указано значение меньше - сессия будет жить 10 минут. Значение по-умолчанию: 10 минут.

Система ссылок

pro.bonus-rarus.ru - ЛК маркетолога про версии

mini.bonus-rarus.ru - ЛК маркетолога мини версии

{id}.cl.bonus-rarus.ru - личный кабинет покупателя про версии, где id - это id клиента

cl-m.bonus-rarus.ru/{id} - личный кабинет покупателя мини версии, где id - это id клиента

{id}.op.bonus-rarus.ru - ссылка на сайт-опросник про версии