**Сервис “Авторизация/Аутентификация”**

**Название:**   
Авторизация/Аутентификация (Auth Service)

**Запросы:**

* Выход из кабинета.  
  GET /api/v1/logout
* Проверка, есть-ли сессия (для сервисного взаимодействия).  
  GET /api/v1/auth
* Сообщение с призывом к регистрации/авторизации.  
  GET /api/v1/signin

**Команды:**

* Регистрация  
  POST /api/v1/register {email, password, username, firstname, lastname, phone}
* Авторизация  
  POST /api/v1/login {login, password}

**События:**

* Пользователь зарегистрирован  
  UserCreated (id, email, username, firstname, lastname, roles, phone)

**Зависимости:**

* -

**Комментарии:**

* Реализовано на базе сессий, на стороне пользователя хранятся в cookie, на стороне сервера в Redis.
* Поддержка двух ролей (клиенты, администраторы).

**Сервис “Пользователь”**

**Название:**   
Пользователь (User Service)

**Запросы:**

* Получить информацию о себе (о текущем пользователи).   
  GET /api/v1/user/{id}
* Получить список пользователей (только администраторы)  
  GET /api/v1/admin/user
* Получить информацию о пользователе (только администраторы)

GET /api/v1/admin/user/{id}

* Получить информацию о пользователе (для взаимодействия сервисов)

RPC: GetUserInfo

**Команды:**

* Изменить информацию о себе  
  PUT /api/v1/user/{id} {email, password, username, firstname, lastname, phone}
* Изменить информацию о пользователе (только администраторы)  
  PUT /api/v1/admin/user/{id} {email, password, username, firstname, lastname, phone, roles}

**События:**

* Обновление данных пользователя
* UserUpdated (id, email, username, firstname, lastname, roles, phone)

**Зависимости:**

* -

**Комментарии:**

* -

**Сервис “Счет”**

**Название:**   
Счет (Account Service)

**Запросы:**

* Получить информацию о счете  
  GET /api/v1/account/{id}
* Получить список счетов пользователей (только администраторы)  
  GET /api/v1/admin/account
* Получить информацию о счете пользователя (только администраторы)

GET /api/v1/admin/account/{id}

* Получить информацию о счете пользователя (для взаимодействия сервисов)

RPC: GetAccountInfo

**Команды:**

* Пополнение счета пользователем  
  PUT /api/v1/account {sum}
* Обновление данных по счету (только администраторы)  
  PUT /api/v1/admin/account/{id} {sum, statusId, limitCredit}
* Снятие денег со счета (для взаимодействия сервисов)  
  RPC AccountWithdraw
* Информация по счету

RPC AccountInfo

**События:**

* Деньги зачислены на счет

AccountReplenished {userId, sum}

**Зависимости:**

* Зависит от события создания пользователя (UserCreated), по данному событию создаем счет пользователю.

**Комментарии:**

* -

**Сервис “Автомобиль”**

**Название:**   
Автомобиль (Car Service)

**Запросы:**

* Список автомобилей   
  GET /api/v1/cars
* Информация об автомобиле  
  GET /api/v1/cars/{id}
* Получить информацию о классах автомобиля (только администраторы)

GET /api/v1/admin/cars/types

* Получить информацию о тарифах на автомобиль (только администраторы)

GET /api/v1/admin/cars/rates

* Получить список автомобилей (только администраторы)

GET /api/v1/admin/cars

* Получить информацию по автомобилю (только администраторы)

GET /api/v1/admin/cars/{id}

**Команды:**

* Добавление автомобиля (только администраторы)  
  POST /api/v1/admin/cars
* Обновление данных по автомобилю (только администраторы)  
  PUT /api/v1/admin/cars/{id}
* Удаление автомобиля (только администраторы)

DELETE /api/v1/admin/cars/{id}

**События:**

* CarStatusSaga (carId, status)

**Зависимости:**

* Слушаем событие от сервиса заказов, когда заказ создается, необходимо проверить и поставить в резерв автомобиль - OrderCreatedSaga
* Если заказ был отменен по причине сломанного автомобиля, меняем статус автомобилю – требуется ремонт/проверка, событие OrderCanceled
* Если заказ был успешно выполнен и аренда закончилась, требуется вернуть автомобиль на линию, меняем статус с “используется” на “активный” - OrderFinished

**Комментарии:**

* Возможно события и зависимости дополняются/изменятся с появлением выделенного сервиса поиска.

**Сервис “Заказ”**

**Название:**   
Сервис заказа (Order Service)

**Запросы:**

* Получить информацию о заказе  
  GET /api/v1/order/{id}
* Завершить аренду автомобиля пользователем  
  GET /api/v1/order/{id}/completed
* Получить информацию по заказам (только администраторы)

GET /api/v1/admin/order

**Команды:**

* Создание заказа / использования автомобиля

POST /api/v1/order {carId}

* Отмена заказа пользователем  
  PUT /api/v1/order/{id}/cancel {reason}

**События:**

* Создание заказа, генерируем событие на смену статуса автомобилю  
  OrderCreatedSaga (orderId, carId)
* Событие отмены заказа   
  OrderCanceled (orderId, userId, carId)
* Событие завершения аренды/заказа   
  OrderFinished (orderId, userId, carId)
* Предупреждение о том, что через N минут аренда прекратится связи с отрицательным балансом.   
  OrderCritical (orderId, userId, carId)

**Зависимости:**

* Слушаем событие CarReservedSaga, для смены успешности статуса заказа и начала использования автомобиля или сообщить клиенту о невозможности аренды данного авто.
* Зависим от сервиса “Счет”, есть RPC вызов на получения информации по счету пользователя GetAccountInfo, а также списание средств AccountWithdraw.

**Комментарии:**

* -

**Сервис “Уведомлений”**

**Название:**   
Сервис уведомлений (Notification Service)

**Запросы:**

* Получить список отправленных событий (только администраторы)  
  GET /api/v1/admin/notifications

**Команды:**

* -

**События:**

* -

**Зависимости:**

* Отправляем приветственное сообщение пользователю и кэшируем его данные

UserCreated

* Данные по пользователю обновлены, обновляем кэш

UserUpdated

* Отправляем сообщение о пополнении баланса

AccountReplenished

* Отправляем сообщение об отрицательном балансе

AccountNegative

* Отправляем сообщение об успешном завершении заказа/аренды.

OrderFinished

* Отправляем уведомление администратору о сломанном автомобили.

OrderCanceled

* Отправляем сообщение, что аренда скоро прекратится связи с окончанием денег на счете

OrderCritical

**Комментарии:**

* Кэш по пользователю будут хранится в Redis, в случае, если кэша нет, запрашиваем через RCP: GetUserInfo.