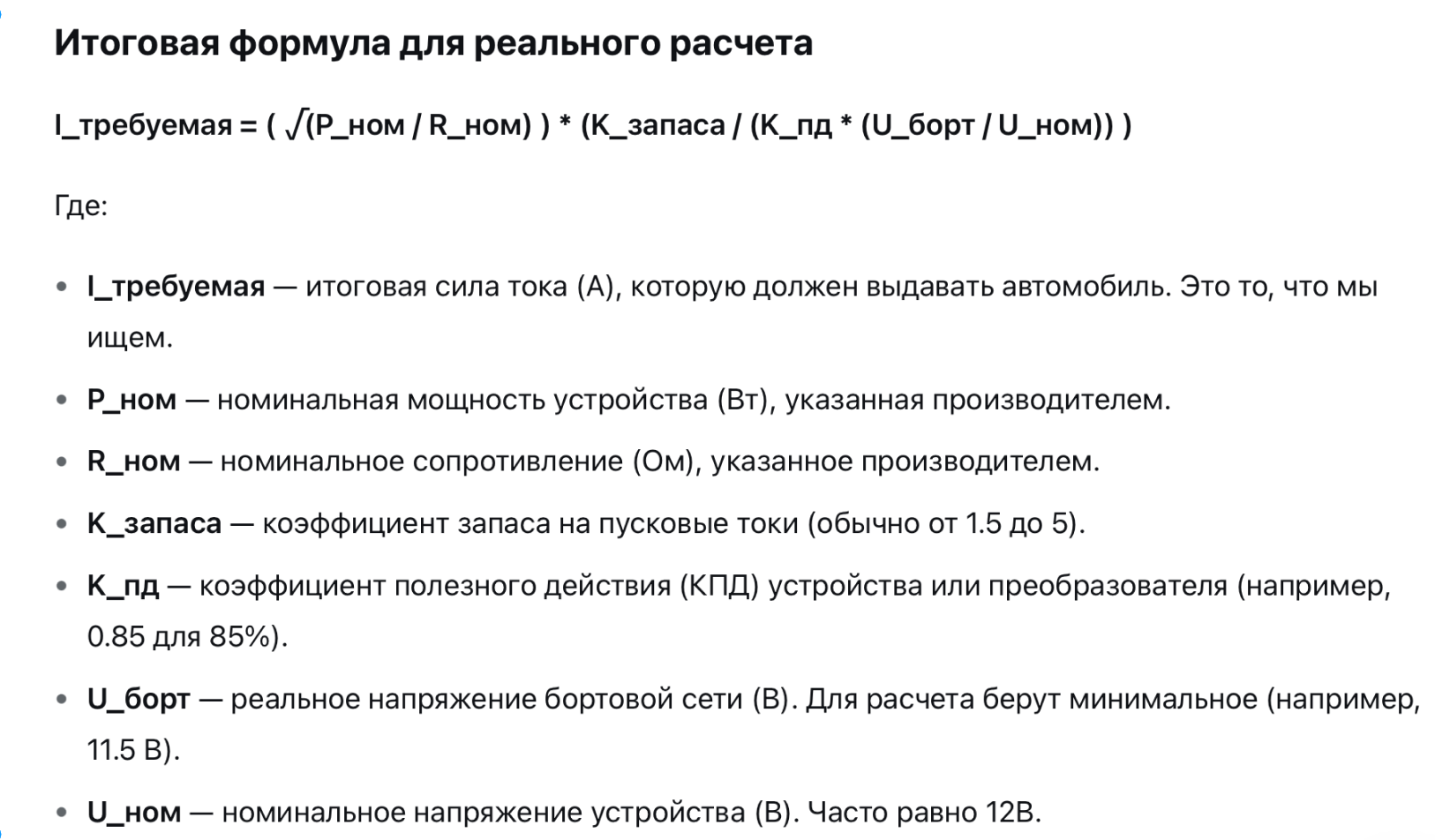
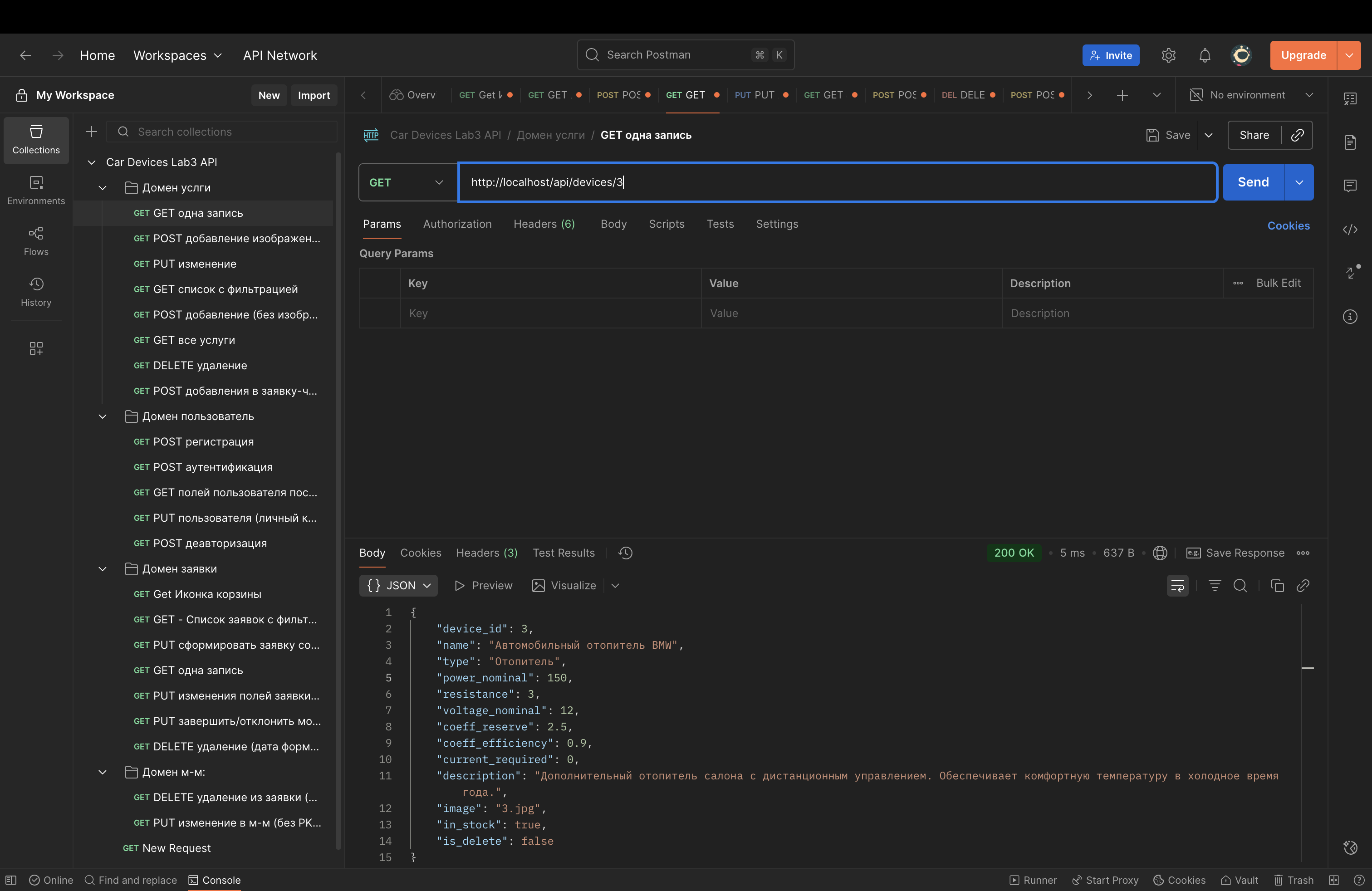
**РИП ЛР3 Порядок показа**

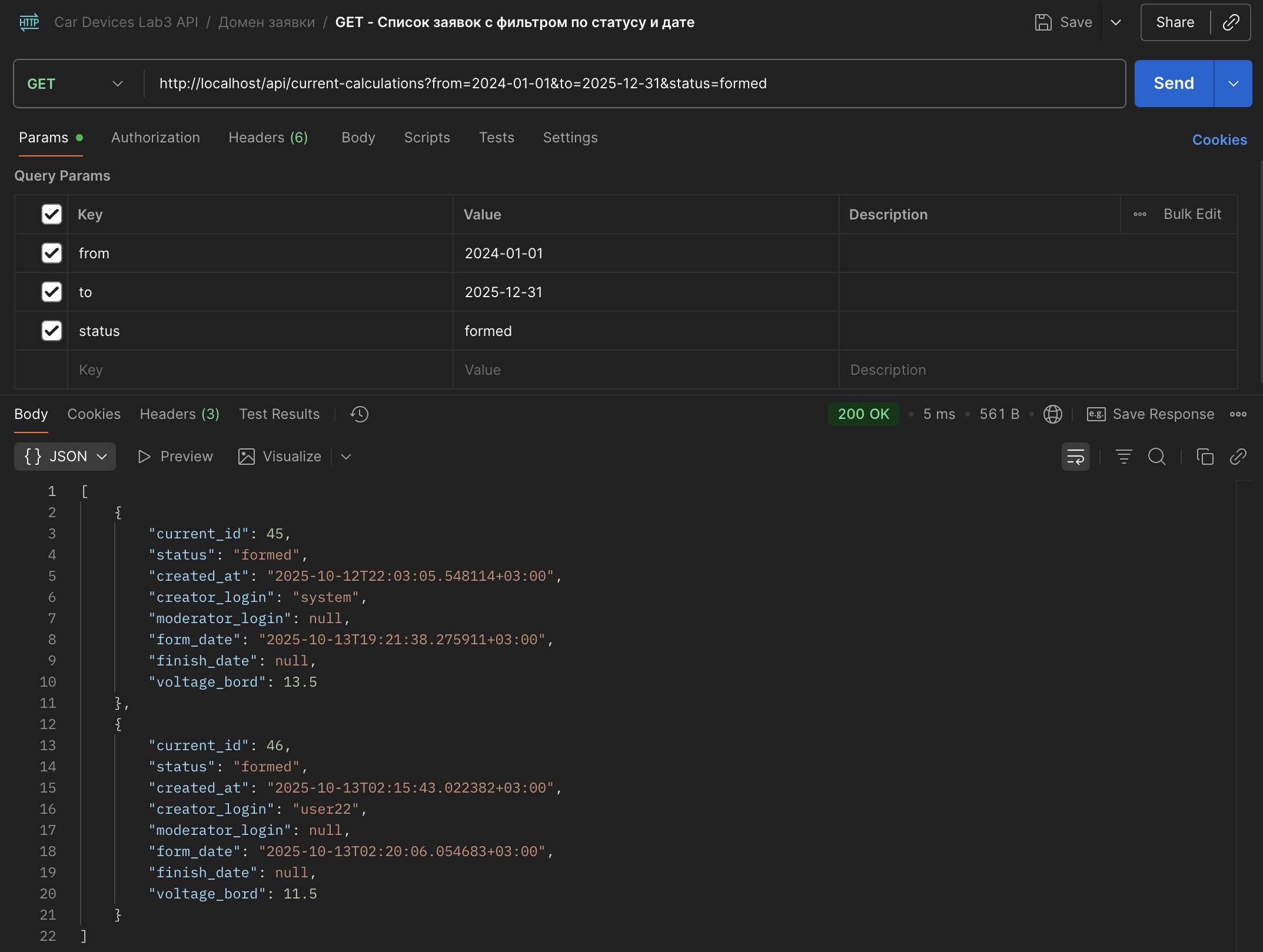
**ТЕМА**  
Электронные устройства и электричество  
**Определение необходимой силы тока для устройств автомобиля.**  
**Услуги** - устройства (колонки, компьютер, внешние устройства по USB), их мощность (Вт) и сопротивление (Ом).   
**Заявка** - расчет минимальной силы тока (А), которую должен выдавать автомобиль для каждого из устройств.



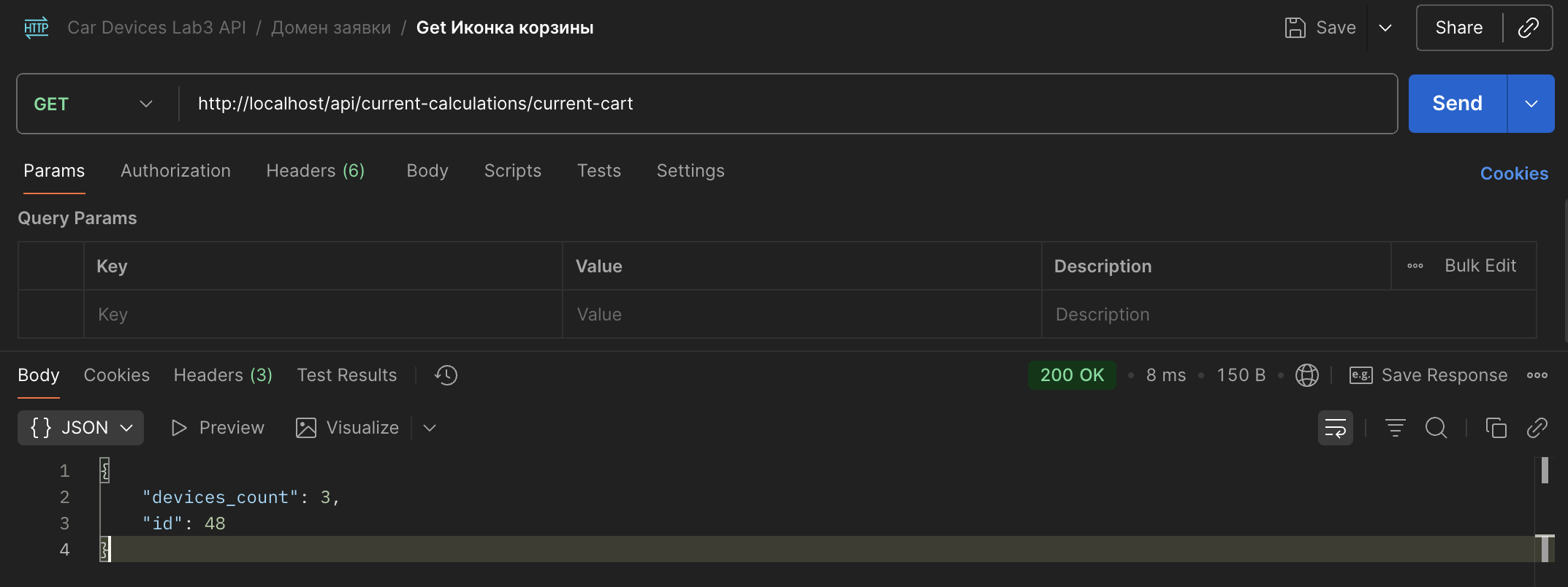
Postman. Коллекция 21 запроса



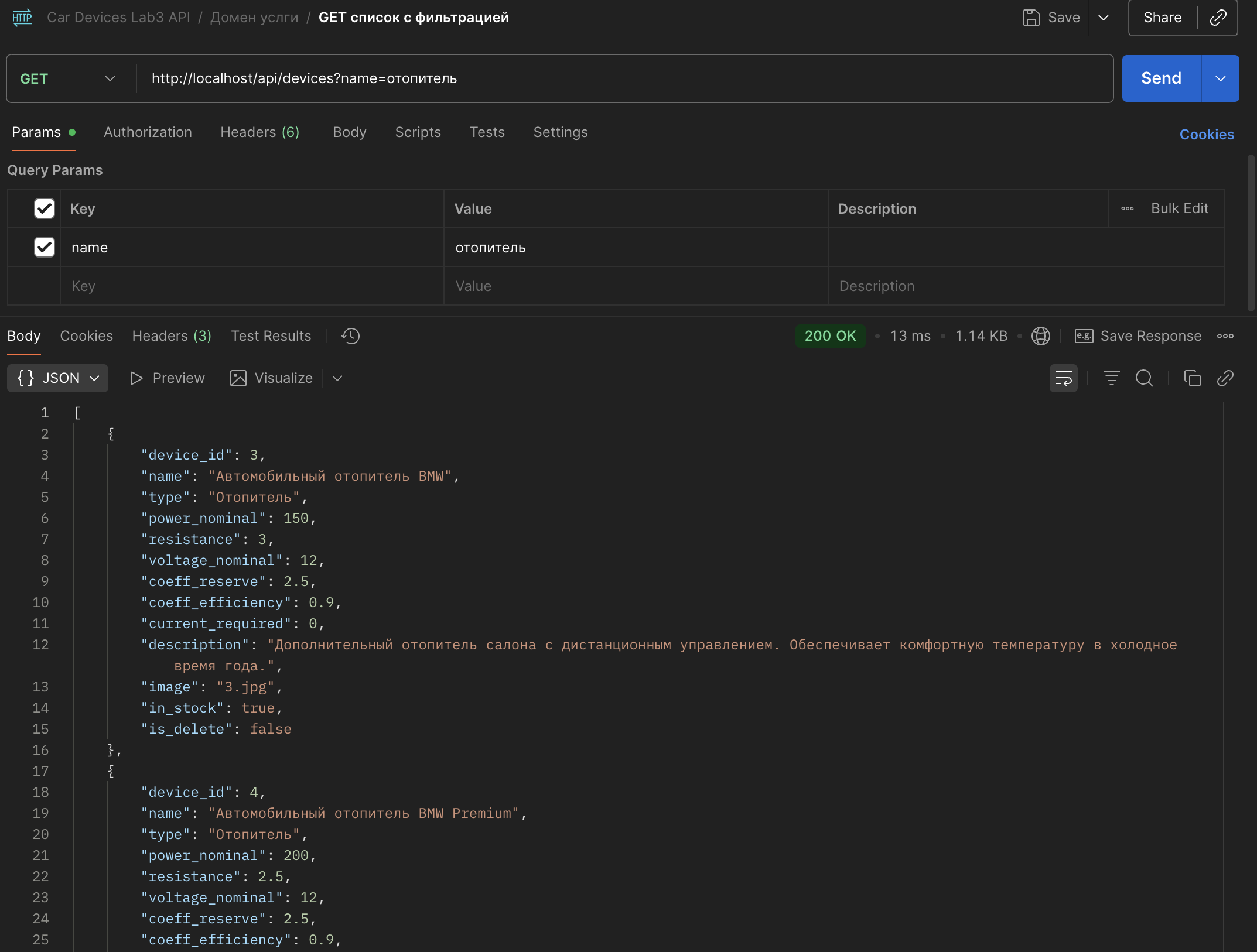
Выполнить GET списка заявок (отфильтровать по дате формирования и статусу)



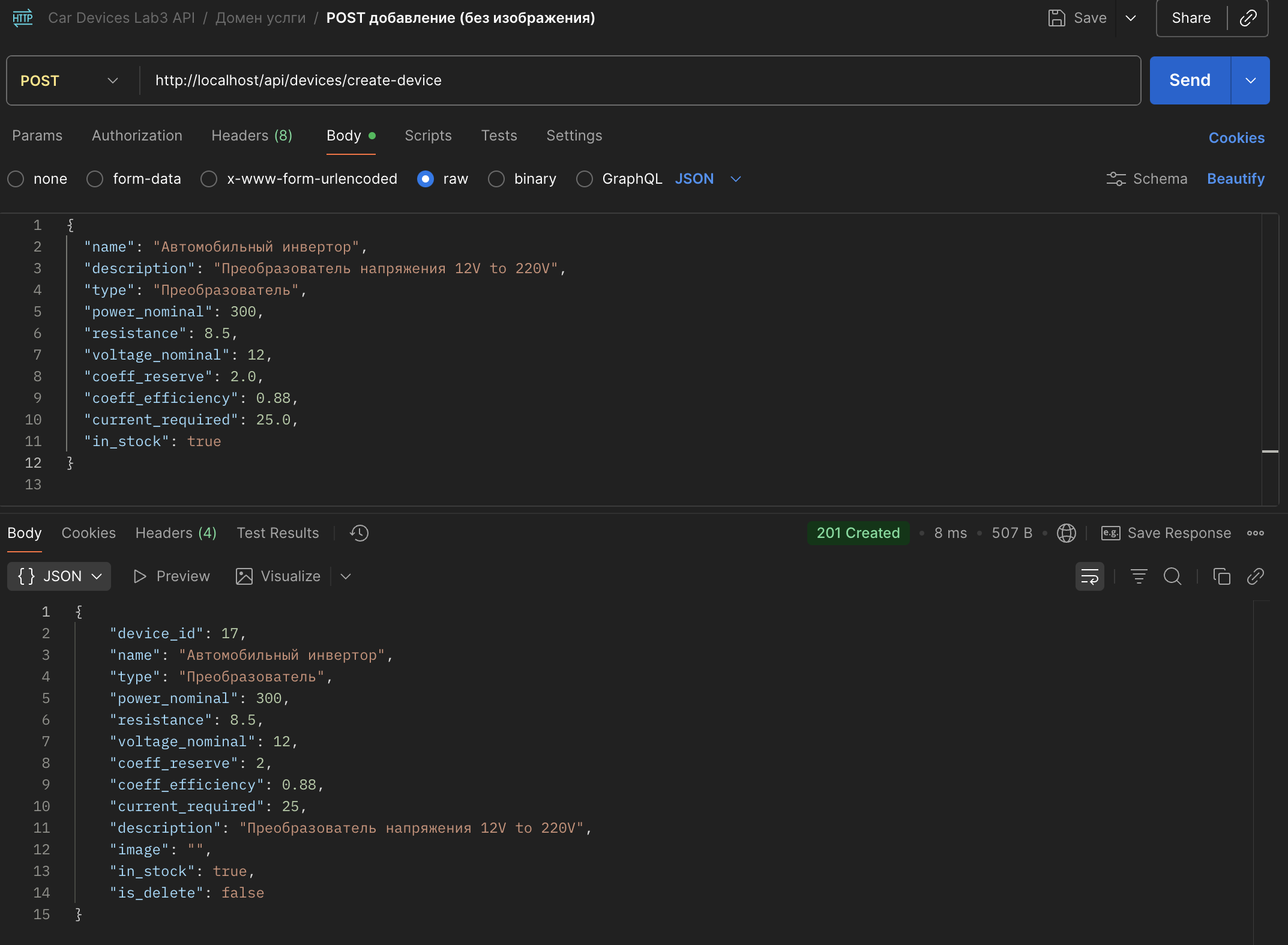
GET иконки корзины, удалить введенную заявку (если есть)

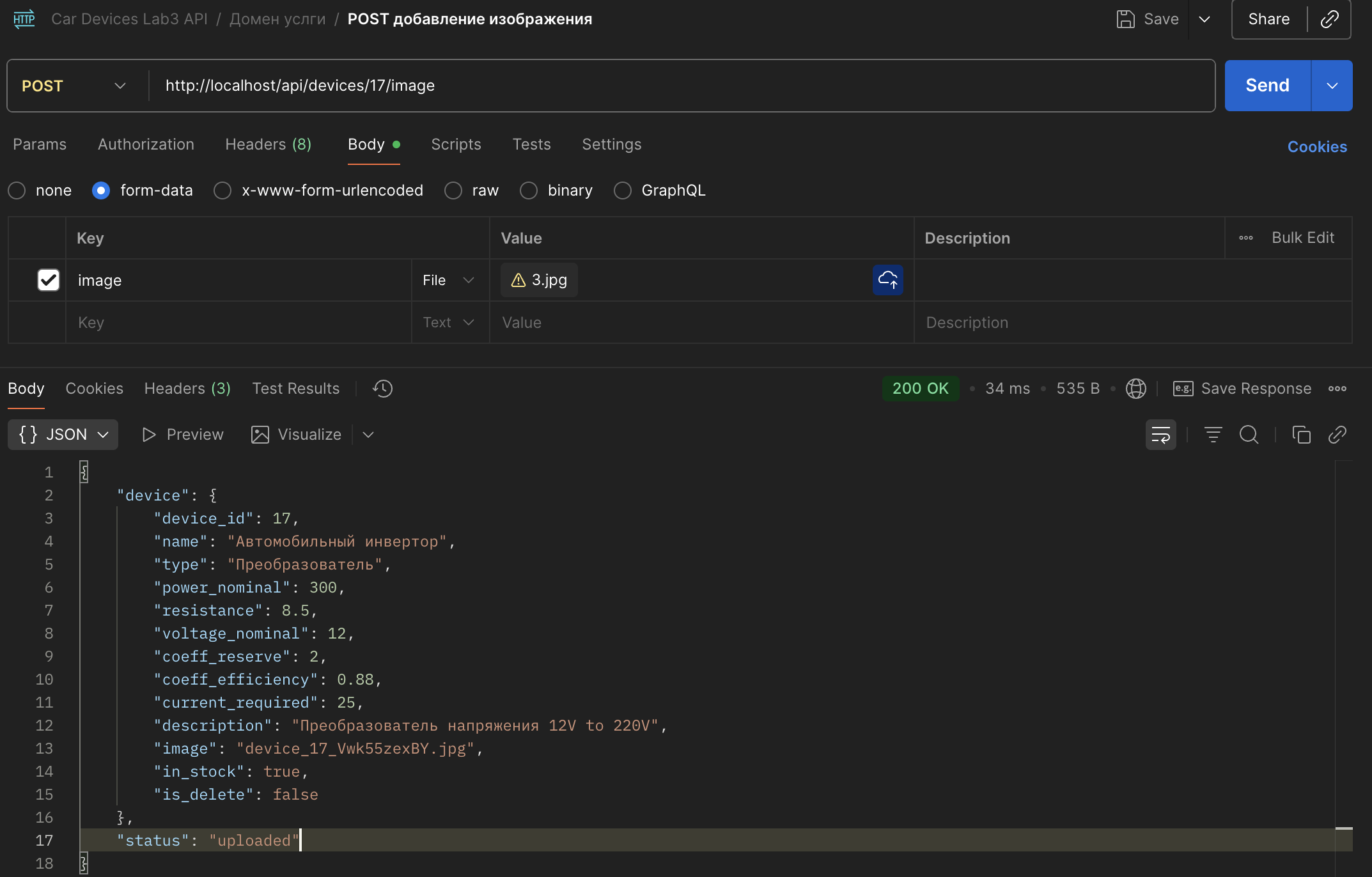


Выполнить GET списка услуг (с фильтром)

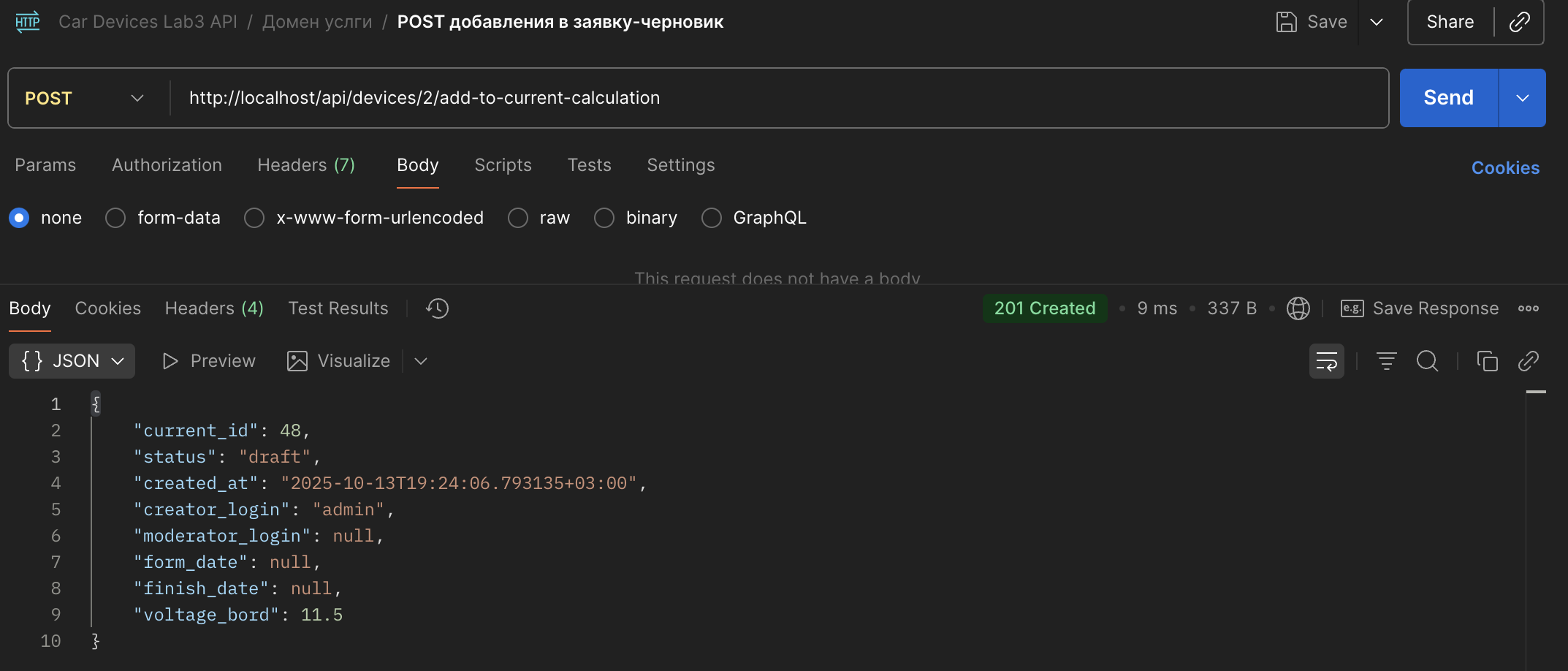


Добавить новую услугу и отдельным методом к ней картинку

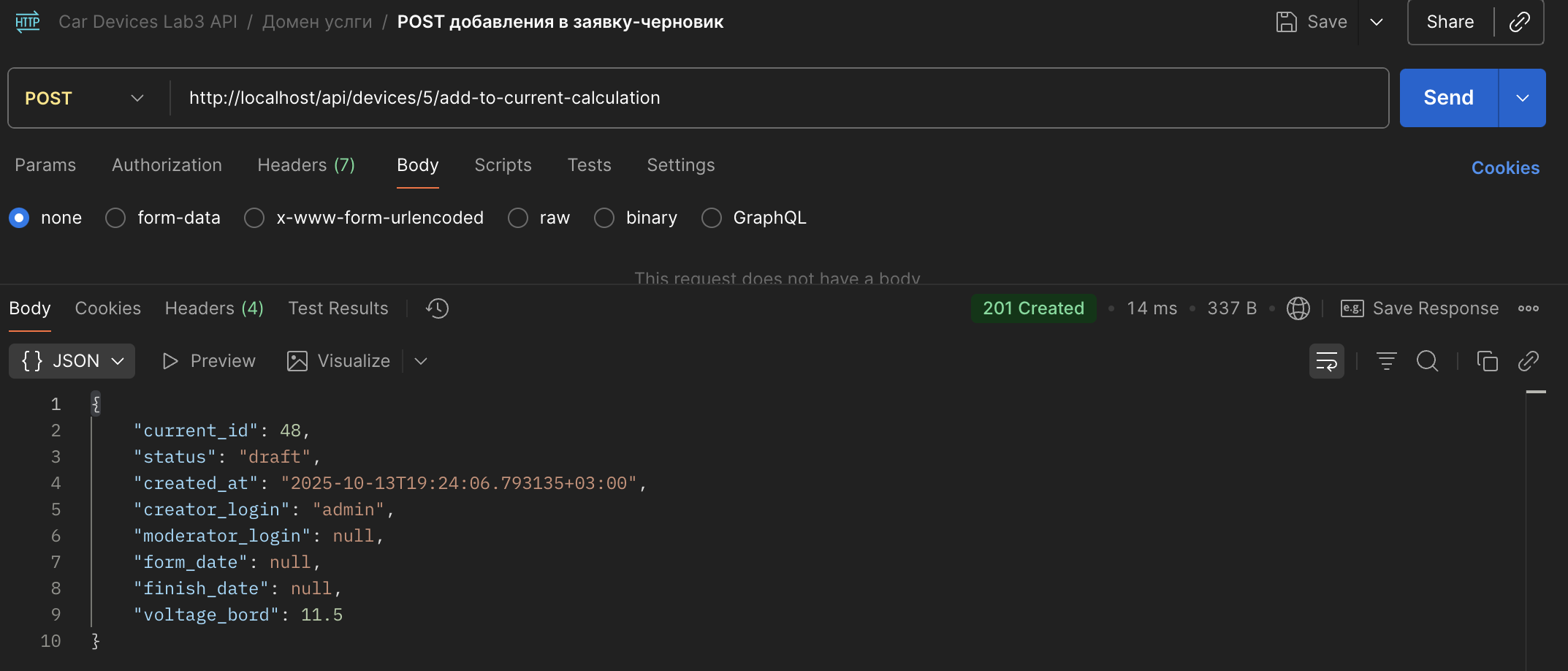




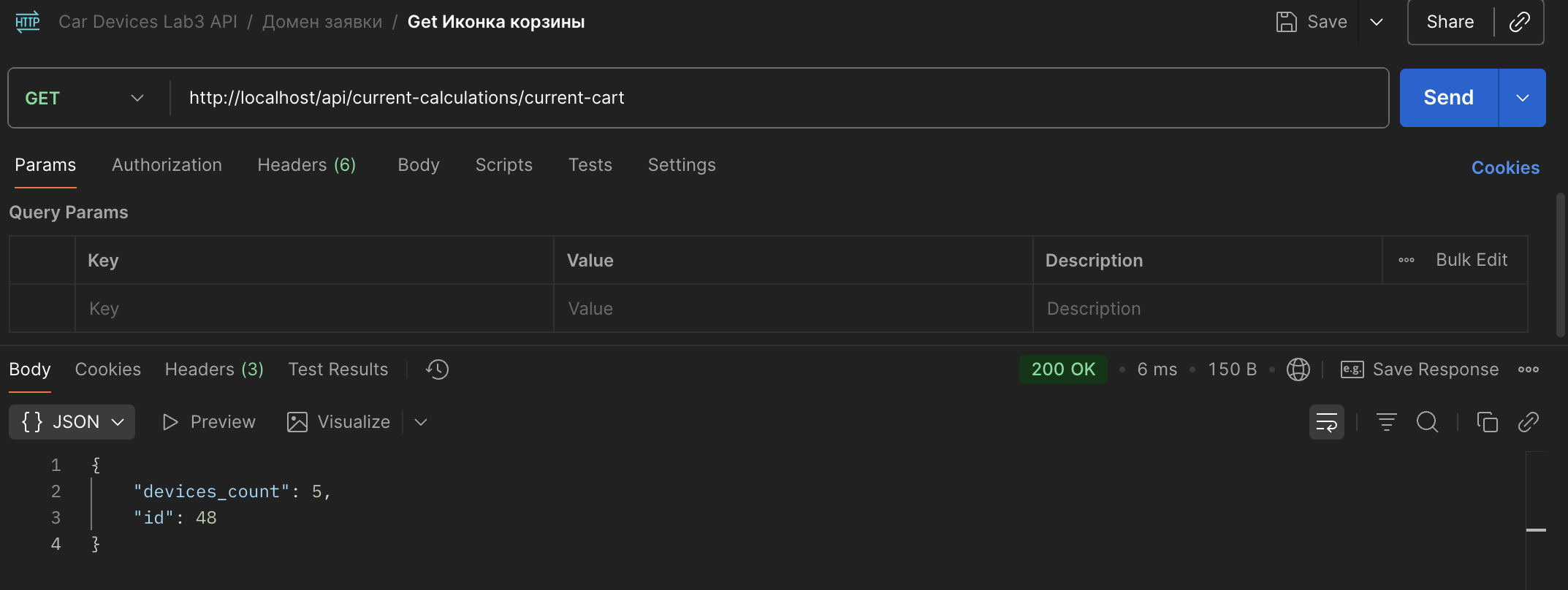
добавить услугу в заявку



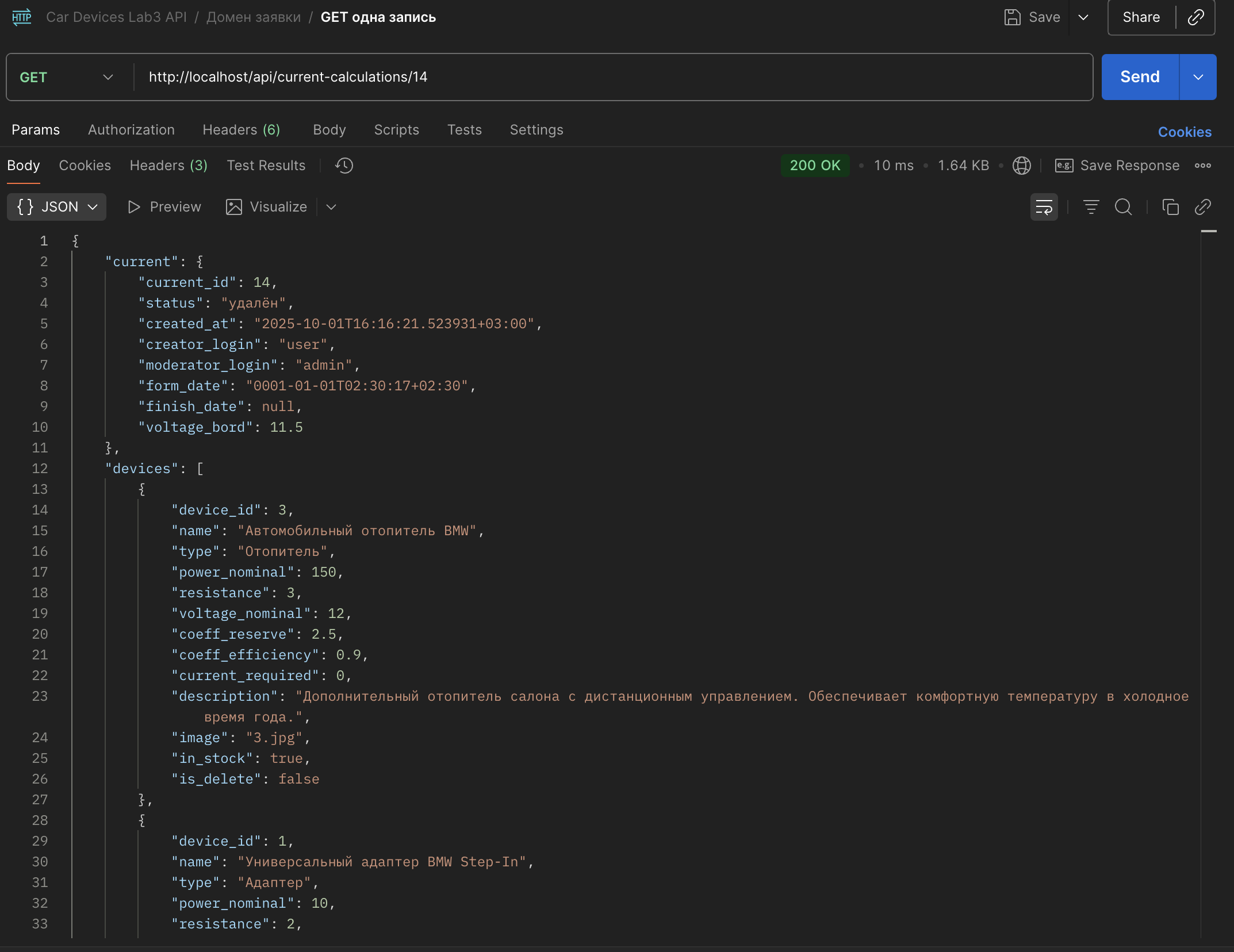
добавить другую услугу в заявку



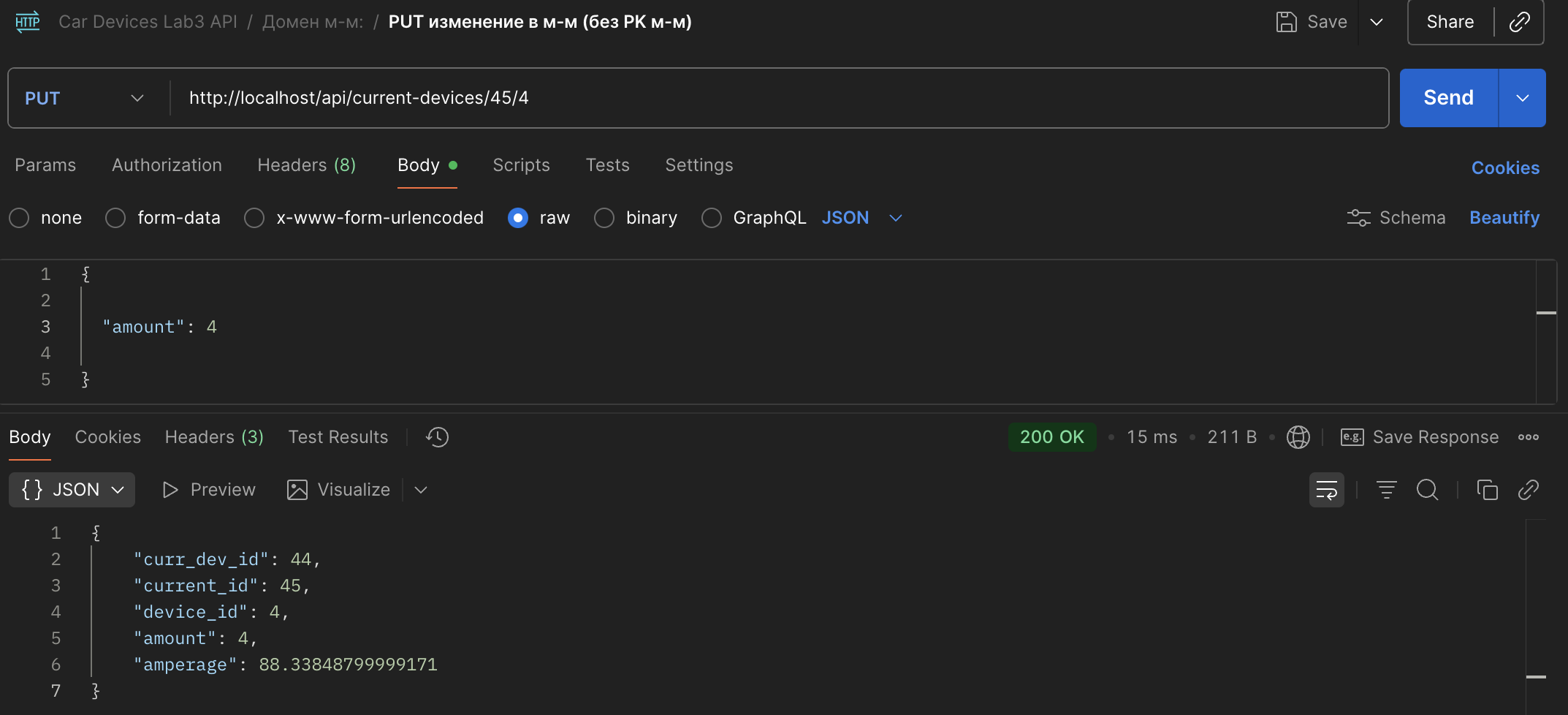
GET иконки корзины



Посмотреть заявку (из 2 услуг)

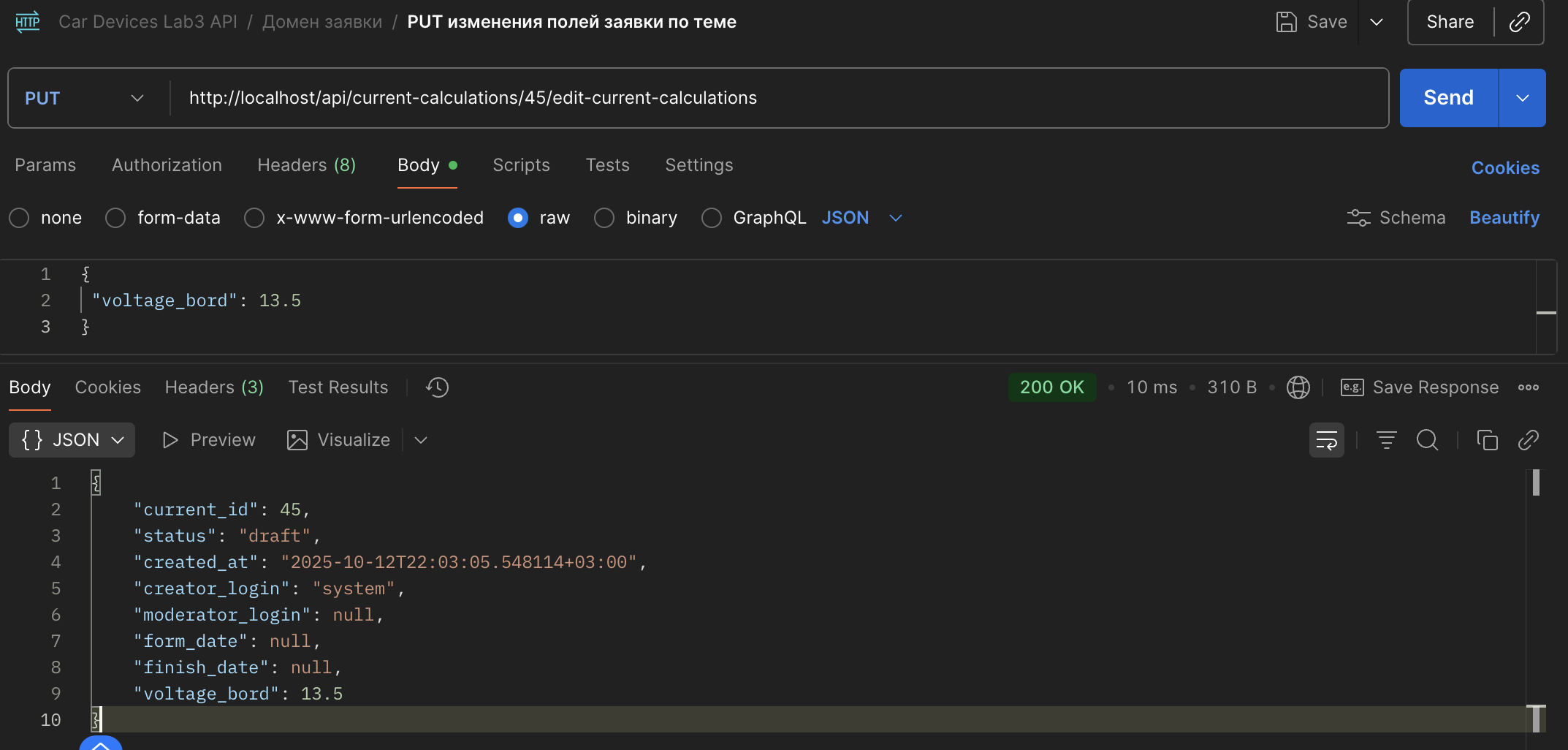


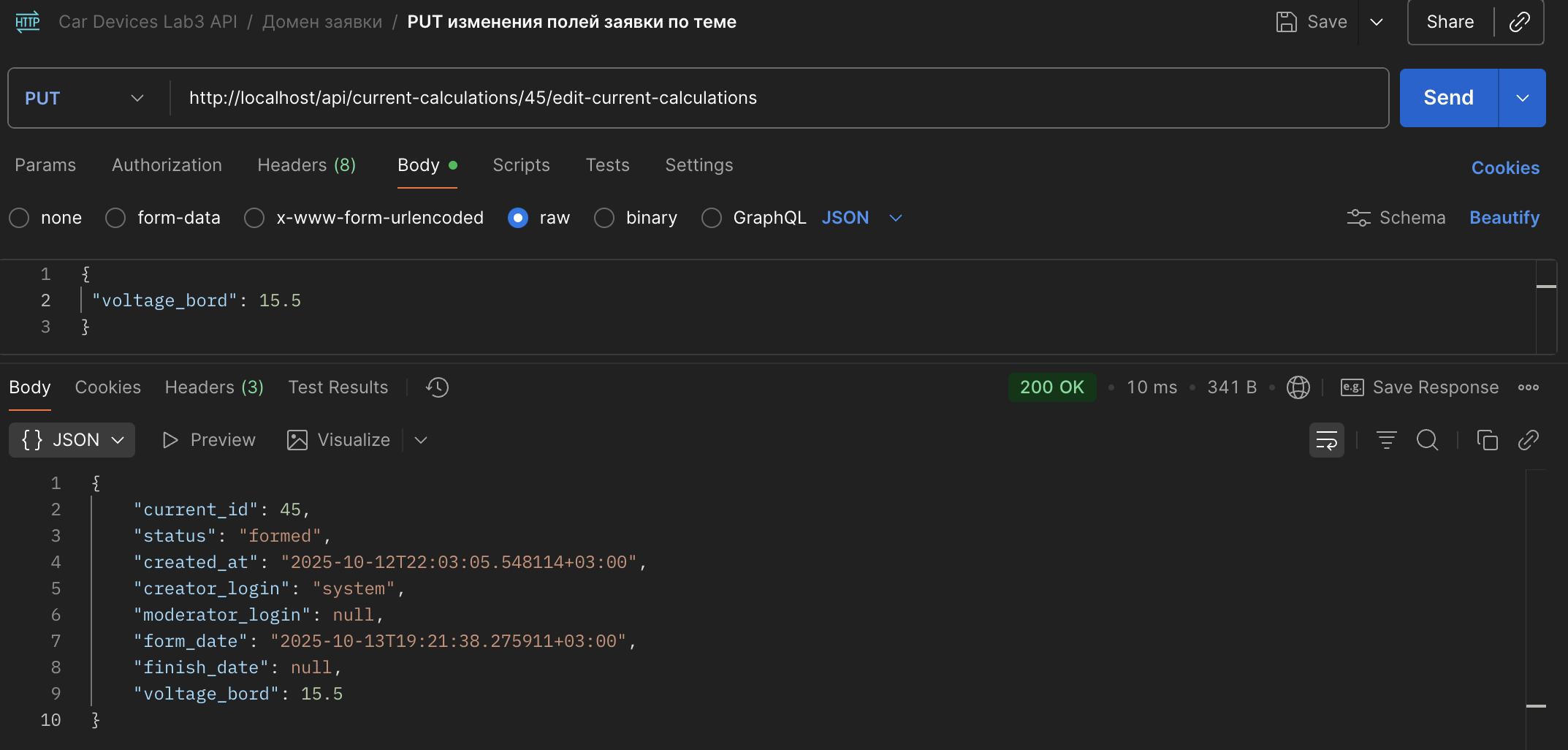
Изменить поле м-м





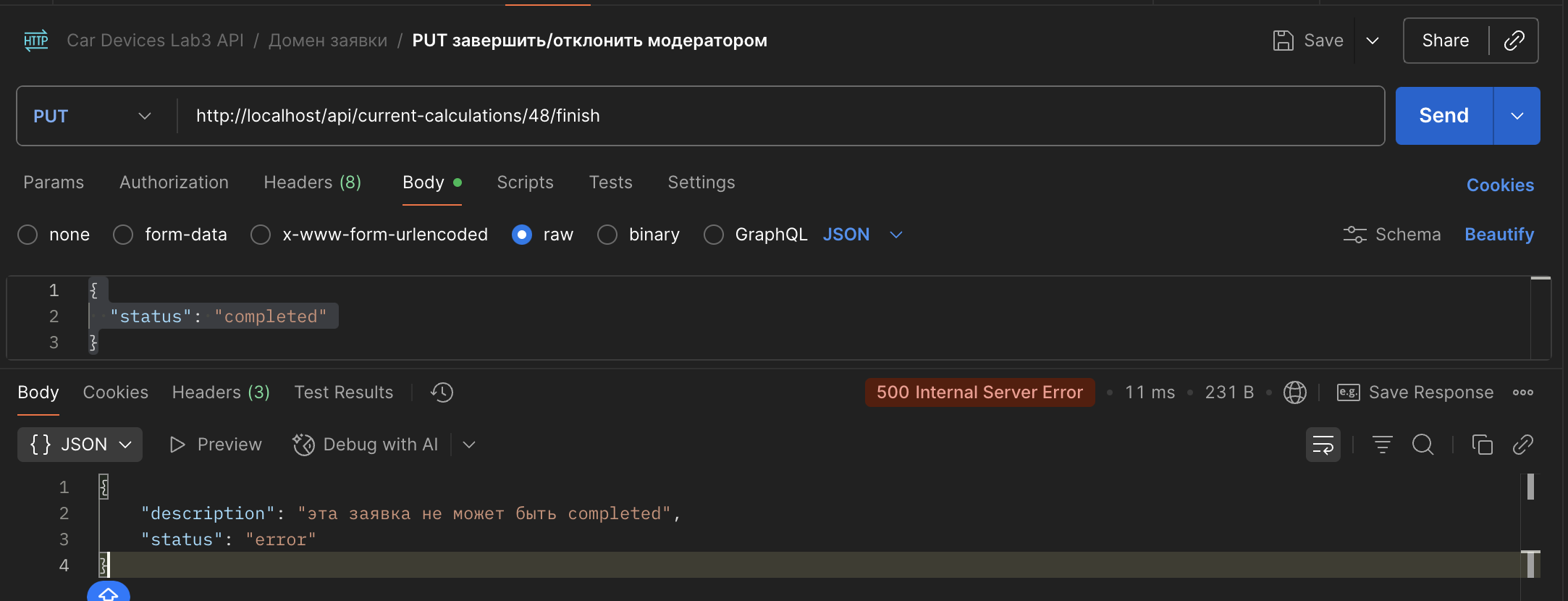
Изменить заявку



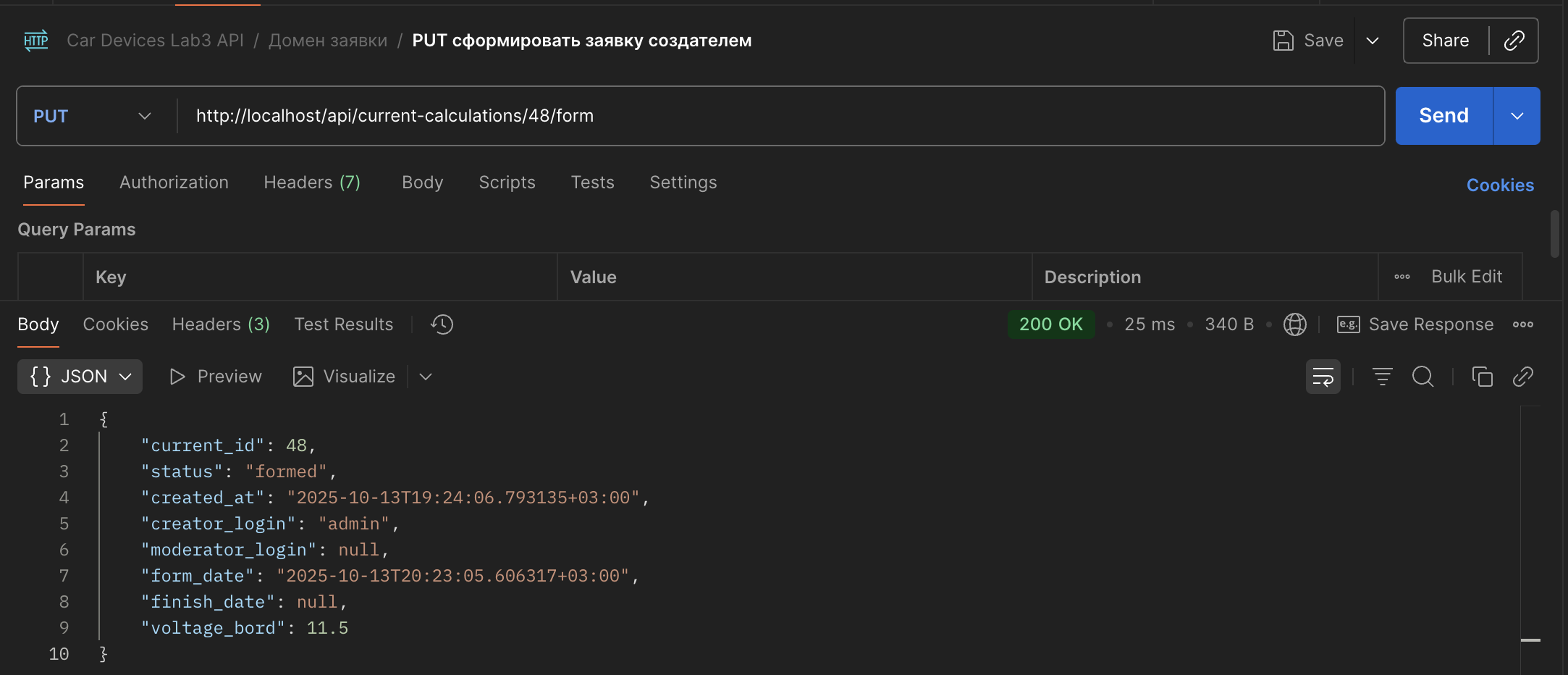


завершить введенную заявку (показать ошибку)

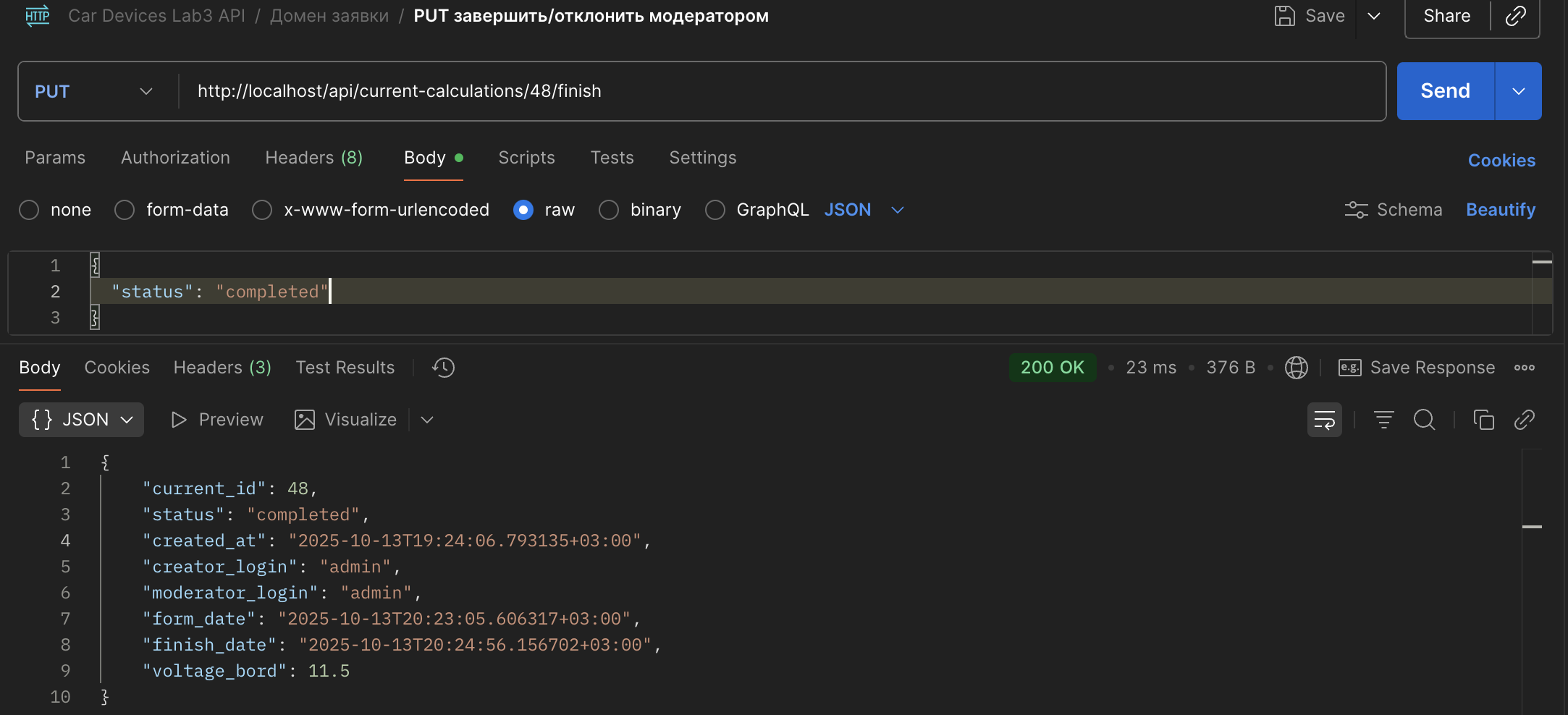


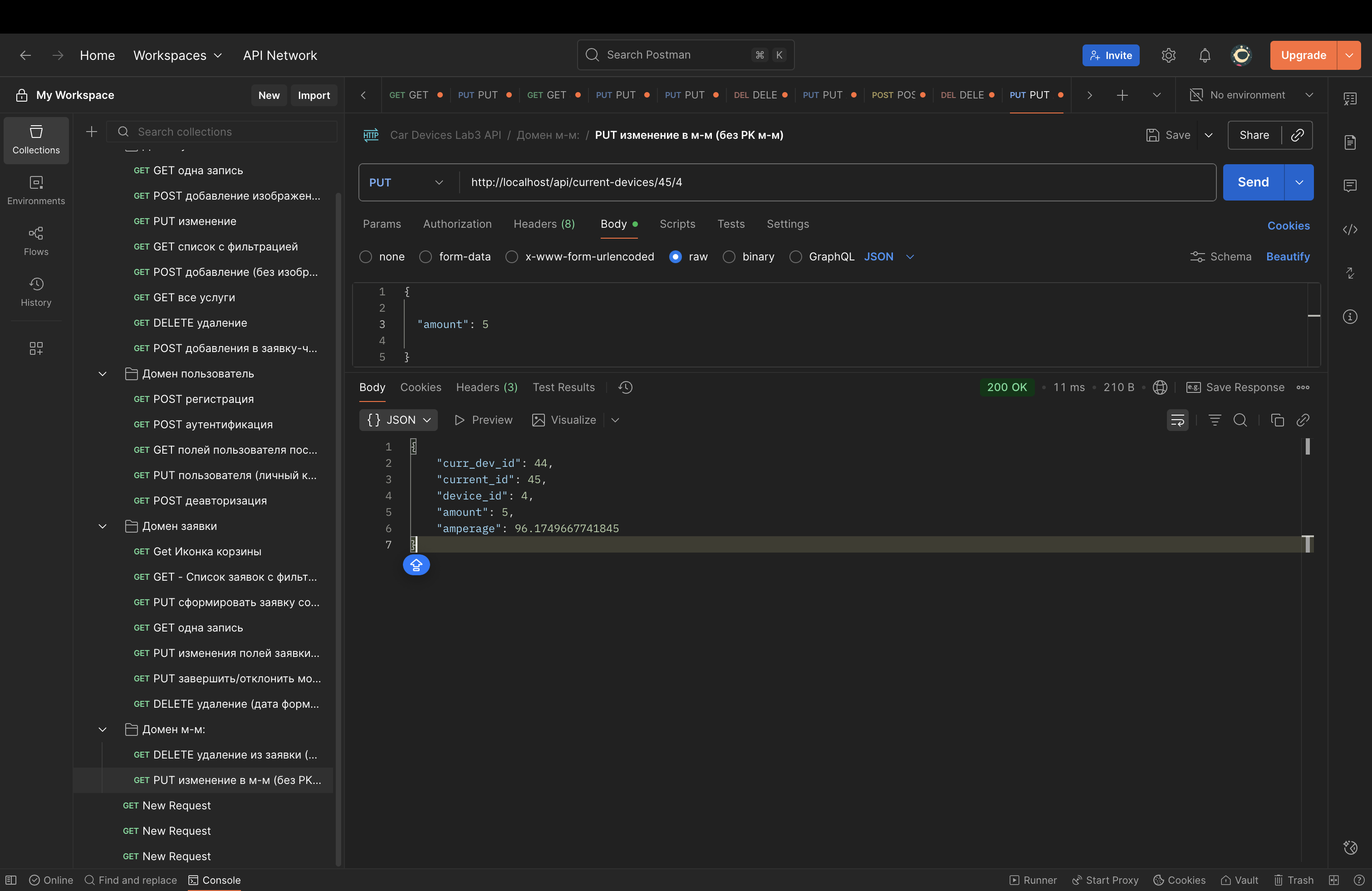


сформировать заявку

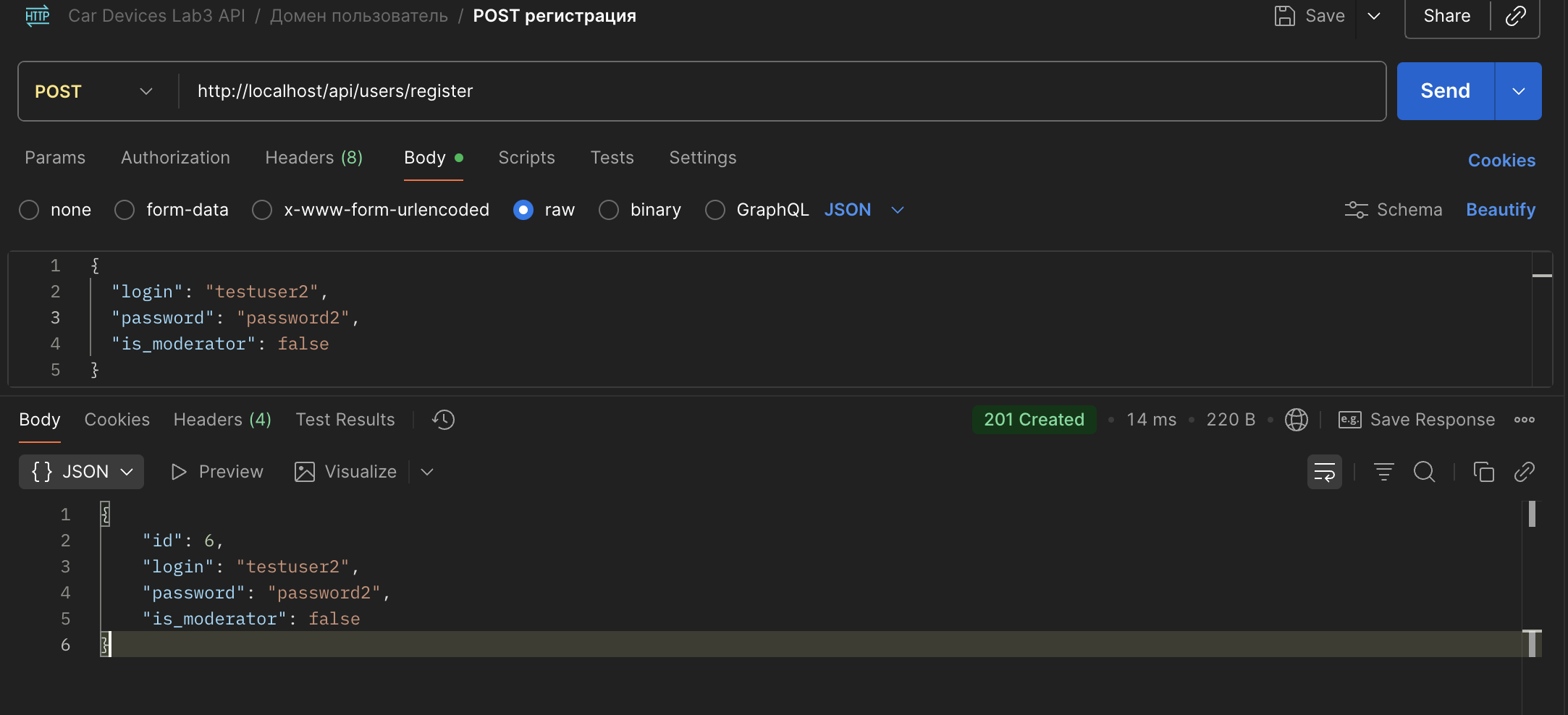


завершить сформированную заявку (вычисление стоимости/доставки)

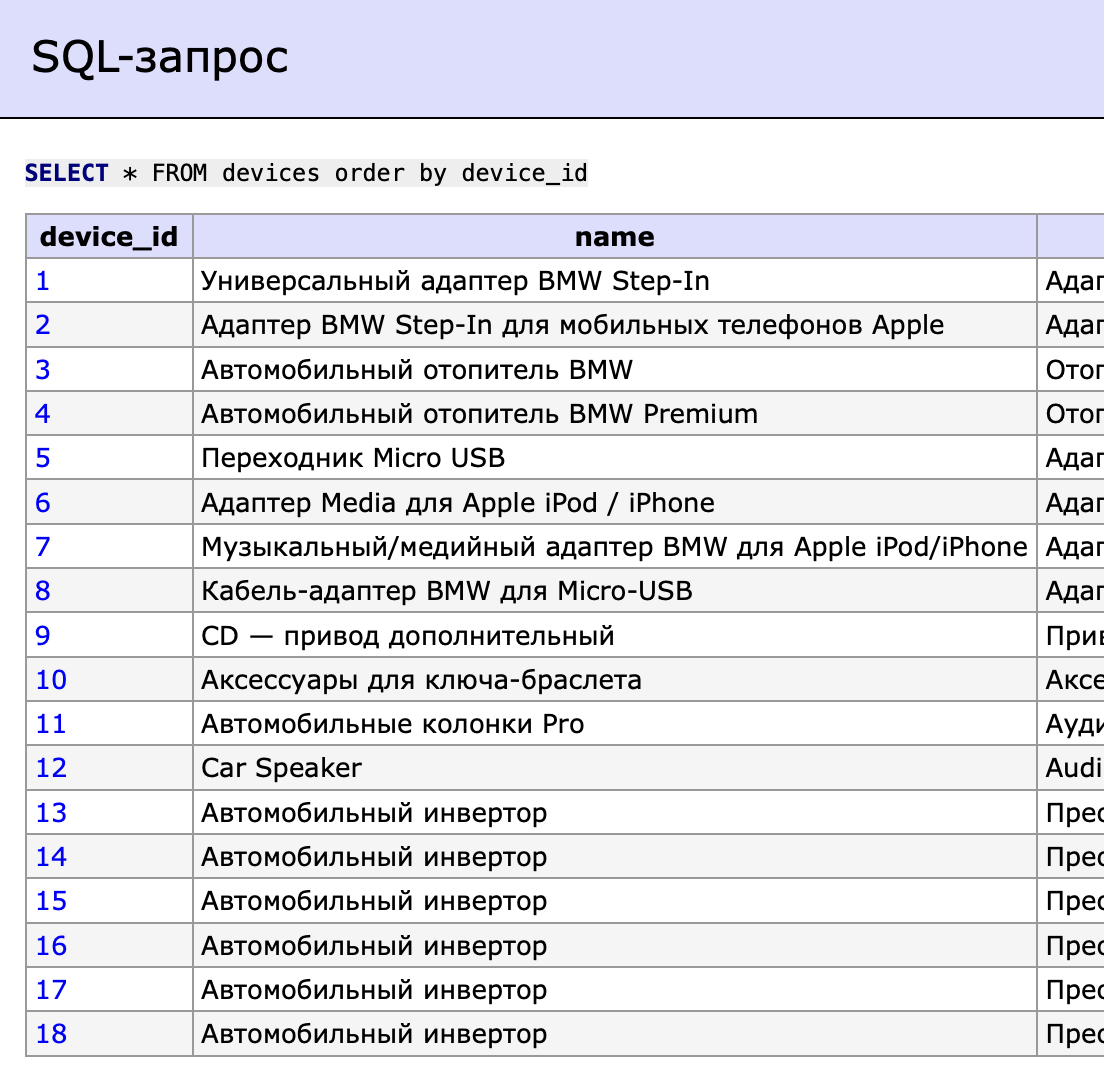


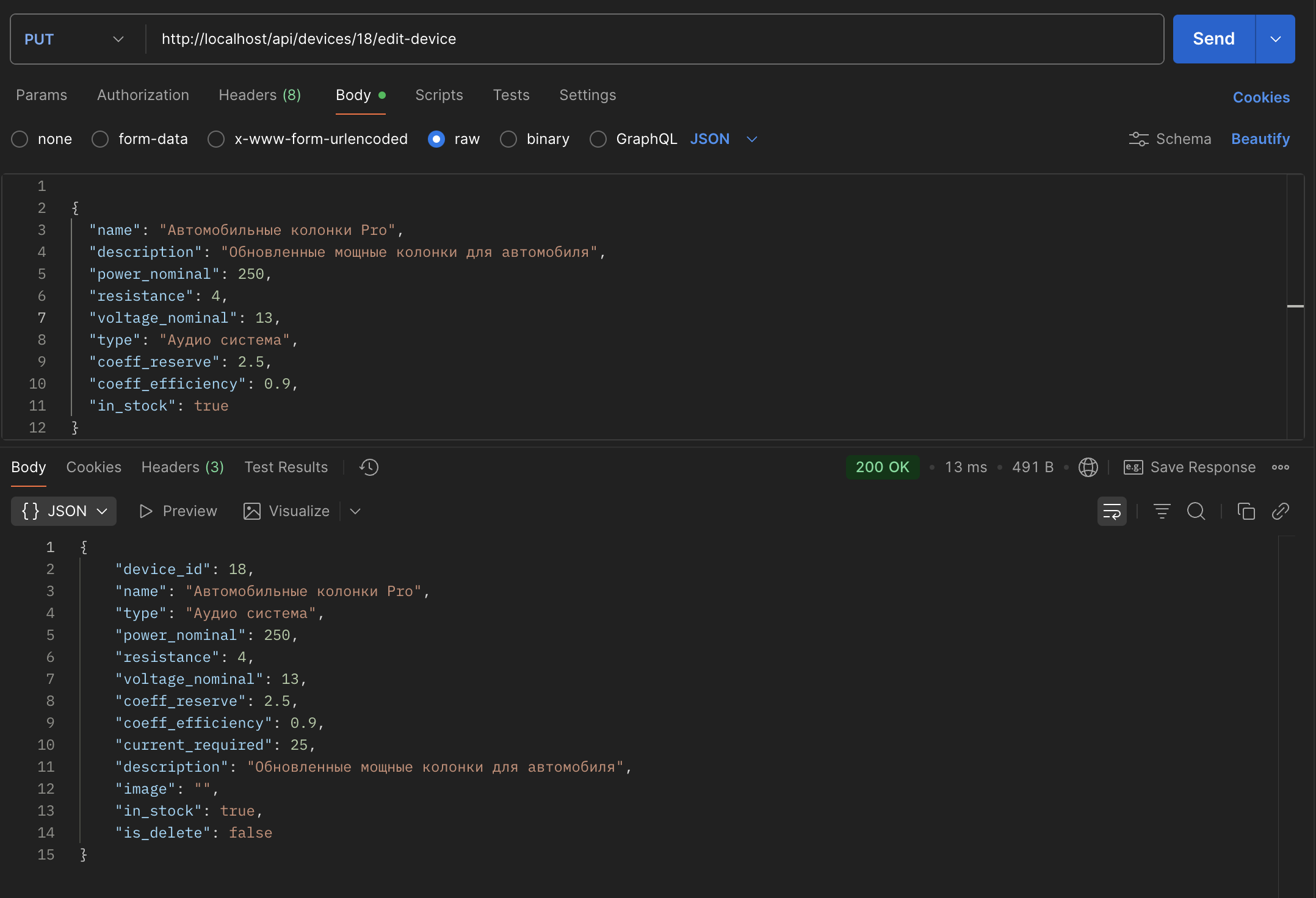


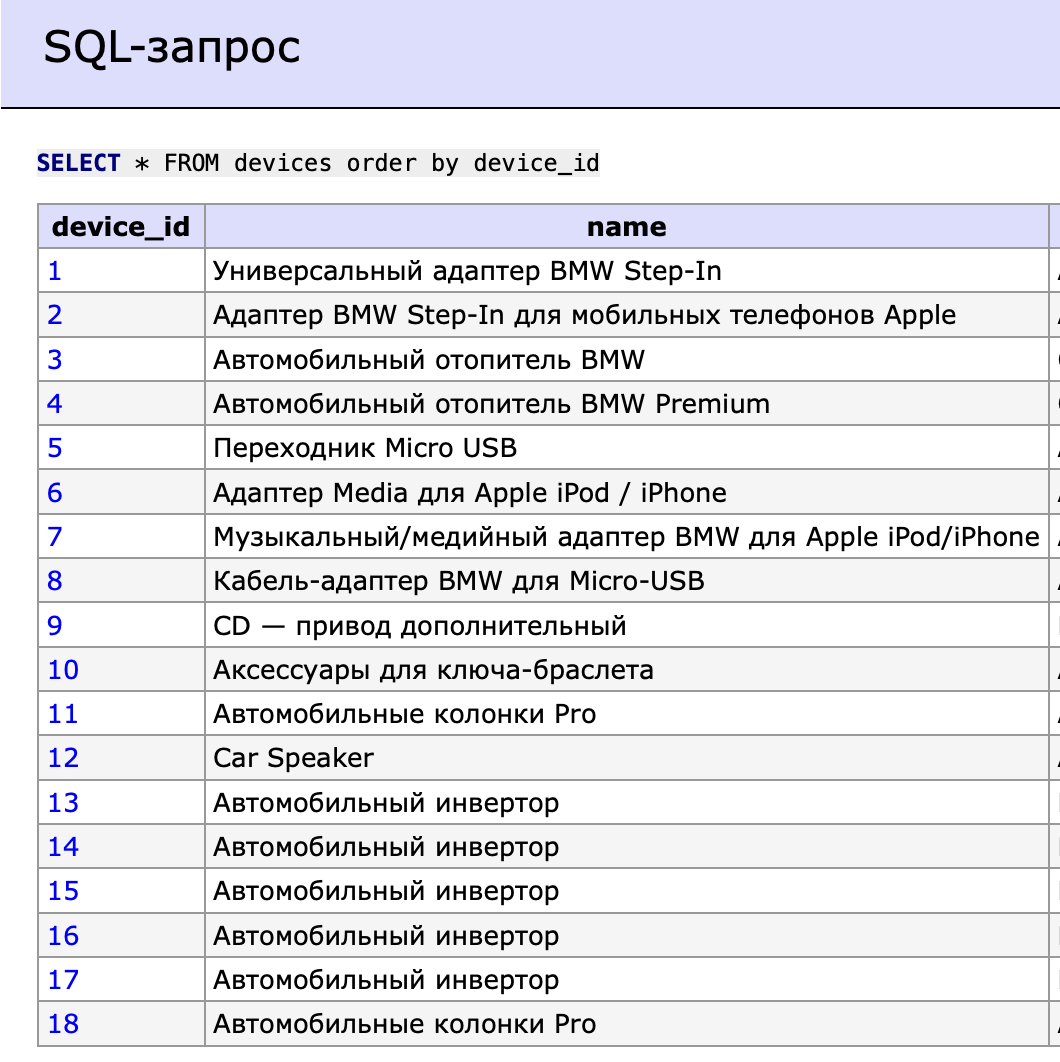
зарегистрировать нового пользователя



Показать измененные данные через select







Модели

*type* Device *struct* {

Device\_ID uint `gorm:"primaryKey" json:"device\_id"`

Name string `gorm:"type:varchar(100);not null" json:"name"`

Type string `gorm:"type:varchar(30)" json:"type"`

PowerNominal float64 `gorm:"type:decimal(10,2);not null" json:"power\_nominal"`

Resistance float64 `gorm:"type:decimal(10,2)" json:"resistance"`

VoltageNominal float64 `gorm:"type:decimal(10,2);default:12.0" json:"voltage\_nominal"`

CoeffReserve float64 `gorm:"type:decimal(10,2);default:2.0" json:"coeff\_reserve"`

CoeffEfficiency float64 `gorm:"type:decimal(10,2);default:0.85" json:"coeff\_efficiency"`

CurrentRequired float64 `gorm:"type:decimal(10,2)" json:"current\_required"`

Description string `gorm:"type:text" json:"description"`

Image string `gorm:"type:varchar(100)" json:"image"`

InStock bool `gorm:"type:boolean;default:true" json:"in\_stock"`

IsDelete bool `gorm:"type:boolean;default:false" json:"is\_delete"`

*// Amount int `json:"amount"`*

}

*type* Current *struct* {

Current\_ID uint `gorm:"primaryKey; not null"`

Status string `gorm:"type:varchar(15); not null"`

Created\_At time.Time `gorm:"not null"`

Creator\_ID uint `gorm:"type:integer(15); not null"`

Moderator\_ID uint `gorm:"type:integer(15); default: null"`

Forming\_Date sql.NullTime `gorm:"default:null"`

Finish\_Date sql.NullTime `gorm:"default:null"`

*//Amperage float64 `gorm:"type:numeric(3,1)"`*

*//Amount int `json:"amount"`*

VoltageBord float64 `gorm:"type:decimal(10,2);default:11.5" json:"voltage\_bord"`

Creator Users `gorm:"foreignKey:Creator\_ID"`

Moderator Users `gorm:"foreignKey:Moderator\_ID"`

}

*type* CurrentDevices *struct* {

CurrDev\_ID uint `gorm:"primaryKey" json:"curr\_dev\_id"`

Current\_ID uint `gorm:"not null" json:"current\_id"`

Device\_ID uint `gorm:"not null" json:"device\_id"`

Amount int `json:"amount"`

*//VoltageBord float64 `gorm:"type:decimal(10,2);default:11.5" json:"voltage\_bord"`*

Amperage float64 `gorm:"type:numeric(10,3)" json:"amperage"`

Device Device `gorm:"foreignKey:Device\_ID;references:Device\_ID"`

Current Current `gorm:"foreignKey:Current\_ID;references:Current\_ID"`

}

*type* Users *struct* {

User\_ID uint `gorm:"primary\_key" json:"id"`

Login string `gorm:"type:varchar(20);unique;not null" json:"login"`

Password string `gorm:"type:varchar(20);not null" json:"-"`

IsModerator bool `gorm:"type:boolean;default:false" json:"is\_moderator"`

}

Сериализаторы

device\_json.go

*package* serializer

*import* "lab3/internal/app/ds"

*// DeviceJSON представляет устройство в формате JSON*

*type* DeviceJSON *struct* {

Device\_ID uint `json:"device\_id"`

Name string `json:"name"`

Type string `json:"type"`

PowerNominal float64 `json:"power\_nominal"`

Resistance float64 `json:"resistance"`

VoltageNominal float64 `json:"voltage\_nominal"`

CoeffReserve float64 `json:"coeff\_reserve"`

CoeffEfficiency float64 `json:"coeff\_efficiency"`

CurrentRequired float64 `json:"current\_required"`

Description string `json:"description"`

Image string `json:"image"`

InStock bool `json:"in\_stock"`

IsDelete bool `json:"is\_delete"`

}

*// DeviceToJSON преобразует ds.Device в DeviceJSON*

*func* *DeviceToJSON*(device ds.Device) DeviceJSON {

*return* DeviceJSON{

Device\_ID: device.Device\_ID,

Name: device.Name,

Type: device.Type,

PowerNominal: device.PowerNominal,

Resistance: device.Resistance,

VoltageNominal: device.VoltageNominal,

CoeffReserve: device.CoeffReserve,

CoeffEfficiency: device.CoeffEfficiency,

CurrentRequired: device.CurrentRequired,

Description: device.Description,

Image: device.Image,

InStock: device.InStock,

IsDelete: device.IsDelete,

}

}

*// DeviceFromJSON преобразует DeviceJSON в ds.Device*

*func* *DeviceFromJSON*(deviceJSON DeviceJSON) ds.Device {

*return* ds.Device{

Device\_ID: deviceJSON.Device\_ID,

Name: deviceJSON.Name,

Type: deviceJSON.Type,

PowerNominal: deviceJSON.PowerNominal,

Resistance: deviceJSON.Resistance,

VoltageNominal: deviceJSON.VoltageNominal,

CoeffReserve: deviceJSON.CoeffReserve,

CoeffEfficiency: deviceJSON.CoeffEfficiency,

CurrentRequired: deviceJSON.CurrentRequired,

Description: deviceJSON.Description,

Image: deviceJSON.Image,

InStock: deviceJSON.InStock,

IsDelete: deviceJSON.IsDelete,

}

}

current\_json.go

*package* serializer

*import* (

"database/sql"

"lab3/internal/app/ds"

"time"

)

*// CurrentJSON представляет заявку на расчет силы тока в формате JSON*

*type* CurrentJSON *struct* {

ID uint `json:"current\_id"`

Status string `json:"status"`

Created\_At time.Time `json:"created\_at"`

Creator\_Login string `json:"creator\_login"`

Moderator\_Login \*string `json:"moderator\_login"`

Forming\_Date \*time.Time `json:"form\_date"`

Finish\_Date \*time.Time `json:"finish\_date"`

VoltageBord float64 `json:"voltage\_bord"`

}

*// StatusJSON представляет статус для обновления заявки*

*type* StatusJSON *struct* {

Status string `json:"status"` *// Статус заявки*

}

*// CurrentToJSON преобразует структуру ds.Current в JSON-формат с учетом логинов*

*func* *CurrentToJSON*(current ds.Current, creator\_login string, moderator\_login string) CurrentJSON {

*var* form\_date, finish\_date \*time.Time

*if* current.Forming\_Date.Valid {

form\_date = &current.Forming\_Date.Time

}

*if* current.Finish\_Date.Valid {

finish\_date = &current.Finish\_Date.Time

}

*var* m\_login \*string

*if* moderator\_login != "" {

m\_login = &moderator\_login

}

*return* CurrentJSON{

ID: current.Current\_ID,

Status: current.Status,

Created\_At: current.Created\_At,

Creator\_Login: creator\_login,

Moderator\_Login: m\_login,

Forming\_Date: form\_date,

Finish\_Date: finish\_date,

VoltageBord: current.VoltageBord,

}

}

*// CurrentFromJSON преобразует JSON-данные в структуру ds.Current*

*func* *CurrentFromJSON*(currentJSON CurrentJSON) ds.Current {

current := ds.Current{

Current\_ID: currentJSON.ID,

Status: currentJSON.Status,

Created\_At: currentJSON.Created\_At,

VoltageBord: currentJSON.VoltageBord,

}

*// Преобразуем Forming\_Date из \*time.Time в sql.NullTime*

*if* currentJSON.Forming\_Date != nil {

current.Forming\_Date = sql.NullTime{Time: \*currentJSON.Forming\_Date, Valid: true}

}

*// Преобразуем Finish\_Date из \*time.Time в sql.NullTime*

*if* currentJSON.Finish\_Date != nil {

current.Finish\_Date = sql.NullTime{Time: \*currentJSON.Finish\_Date, Valid: true}

}

*// Creator\_ID и Moderator\_ID остаются неизменными, так как логины не преобразуются в ID напрямую*

*return* current

}

dev\_cur\_json.go

*package* serializer

*import* "lab3/internal/app/ds"

*// CurrentDeviceJSON представляет связь заявки и устройства в формате JSON*

*type* CurrentDeviceJSON *struct* {

CurrDev\_ID uint `json:"curr\_dev\_id"`

Current\_ID uint `json:"current\_id"`

Device\_ID uint `json:"device\_id"`

Amount int `json:"amount"`

Amperage float64 `json:"amperage"`

}

*// CurrentDeviceToJSON преобразует ds.CurrentDevices в CurrentDeviceJSON*

*func* *CurrentDeviceToJSON*(currentDevice ds.CurrentDevices) CurrentDeviceJSON {

*return* CurrentDeviceJSON{

CurrDev\_ID: currentDevice.CurrDev\_ID,

Current\_ID: currentDevice.Current\_ID,

Device\_ID: currentDevice.Device\_ID,

Amount: currentDevice.Amount,

Amperage: currentDevice.Amperage,

}

}

*// CurrentDeviceFromJSON преобразует CurrentDeviceJSON в ds.CurrentDevices*

*func* *CurrentDeviceFromJSON*(deviceJSON CurrentDeviceJSON) ds.CurrentDevices {

*return* ds.CurrentDevices{

CurrDev\_ID: deviceJSON.CurrDev\_ID,

Current\_ID: deviceJSON.Current\_ID,

Device\_ID: deviceJSON.Device\_ID,

Amount: deviceJSON.Amount,

Amperage: deviceJSON.Amperage,

}

}

users\_json.go

*package* serializer

*import* "lab3/internal/app/ds"

*// UserJSON представляет пользователя в формате JSON*

*type* UserJSON *struct* {

ID uint `json:"id"`

Login string `json:"login"`

Password string `json:"password"`

IsModerator bool `json:"is\_moderator"`

}

*// UserToJSON преобразует ds.Users в UserJSON*

*func* *UserToJSON*(user ds.Users) UserJSON {

*return* UserJSON{

ID: user.User\_ID,

Login: user.Login,

Password: user.Password,

IsModerator: user.IsModerator,

}

}

*// UserFromJSON преобразует UserJSON в ds.Users*

*func* *UserFromJSON*(userJSON UserJSON) ds.Users {

*return* ds.Users{

User\_ID: userJSON.ID,

Login: userJSON.Login,

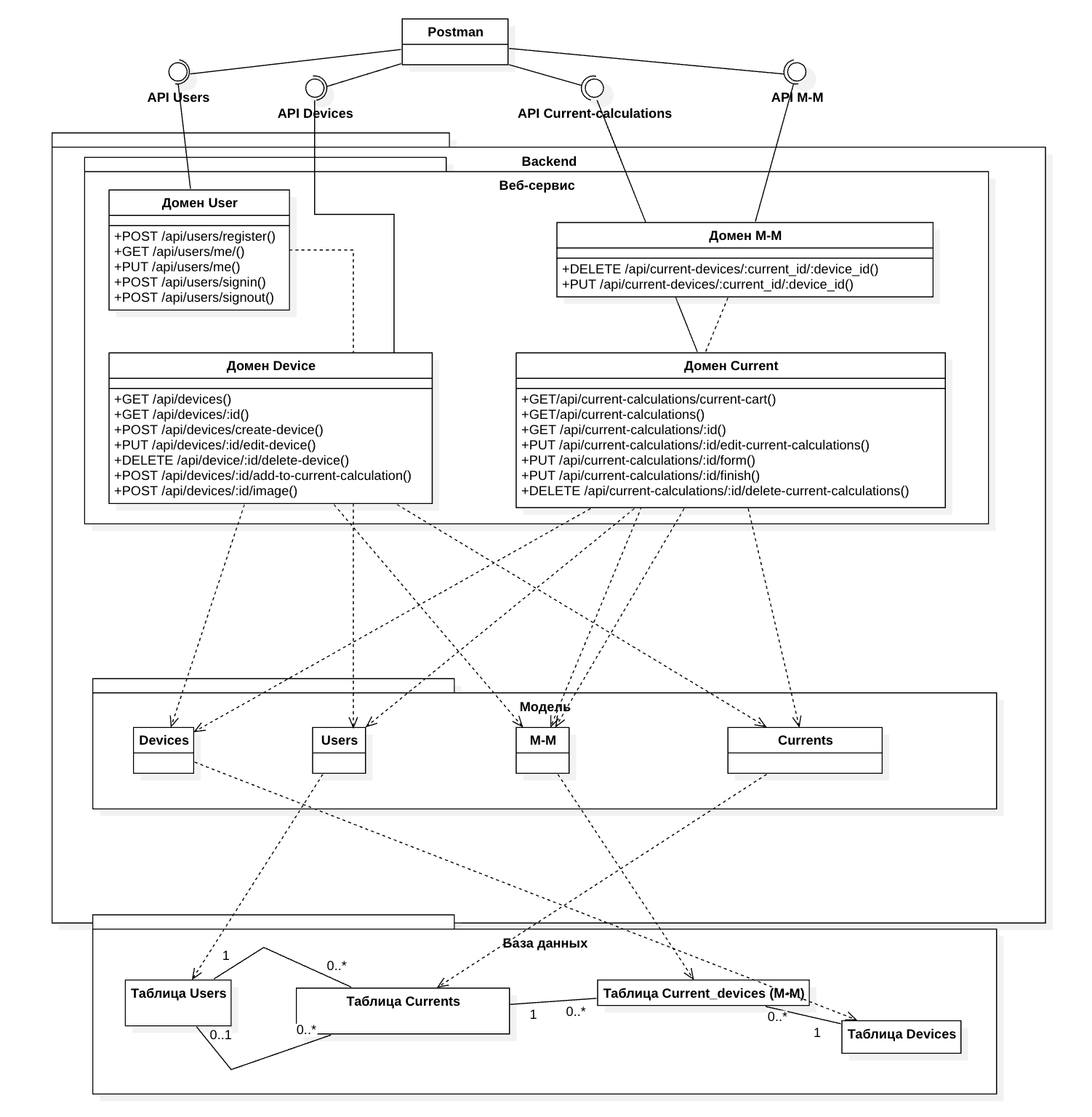
Password: userJSON.Password,

IsModerator: userJSON.IsModerator,

}

}

**Диаграмма классов**

****