Parqueoya API Docs

Data Model

Customer

- customer_id (unique, primary key, indexed)
- customer_name (varchar)
- email (varchar)
- password (bcrypted hash)
- balance (integer)
- available_balance (integer)

Vehicle

- vehicle_id (unique, primary key, indexed)
- license_plate (varchar, indexed)
- customer_id (FK)

Vendor

- vendor_id (unique, primary key, indexed)
- vendor_name (varchar)
- email (varchar)
- password (bcrypted hash)
- latitude (float)
- longitude (float)
- vendor_description (long text)

VendorTerms

- terms_id (unique, primary key, indexed)
- creation_date (timestamp)
- updated_at (timestamp)
- enabled (bool)
- vendor_id (FK)
- hourly_rate (float)
- max_hourly_hours (float)
- flat_rate (float)
- max_flat_hours (float)

VendorUser

- vendoruser_id (unique, primary key, indexed)
- vendoruser_name (varchar)
- email (varchar)
- password (bcrypted hash)

ServiceRequest

- request_id (unique, primary key, indexed)
- creation_date (timestamp)
- vehicle_id (FK)
- latitude (float)
- longitude (float)
- expiry (timestamp)
- client_guid (varchar)

ServiceOffer

- offer_id (unique, primary key, indexed)
- creation_date (timestamp)
- updated_at (timestamp)
- vendor_id (FK)
- terms_id (FK)
- request_id (FK)
- state (enum: pending, accepted, refused)

ServiceContract

- contract_id (unique, primary key, indexed)
- creation_date (timestamp)
- updated_at (timestamp)
- parked_at (timestamp)
- departed_at (timestamp)
- offer_id (FK)
- state (enum: pending, noshow, cancelled, parked, departed)

PaymentTransaction

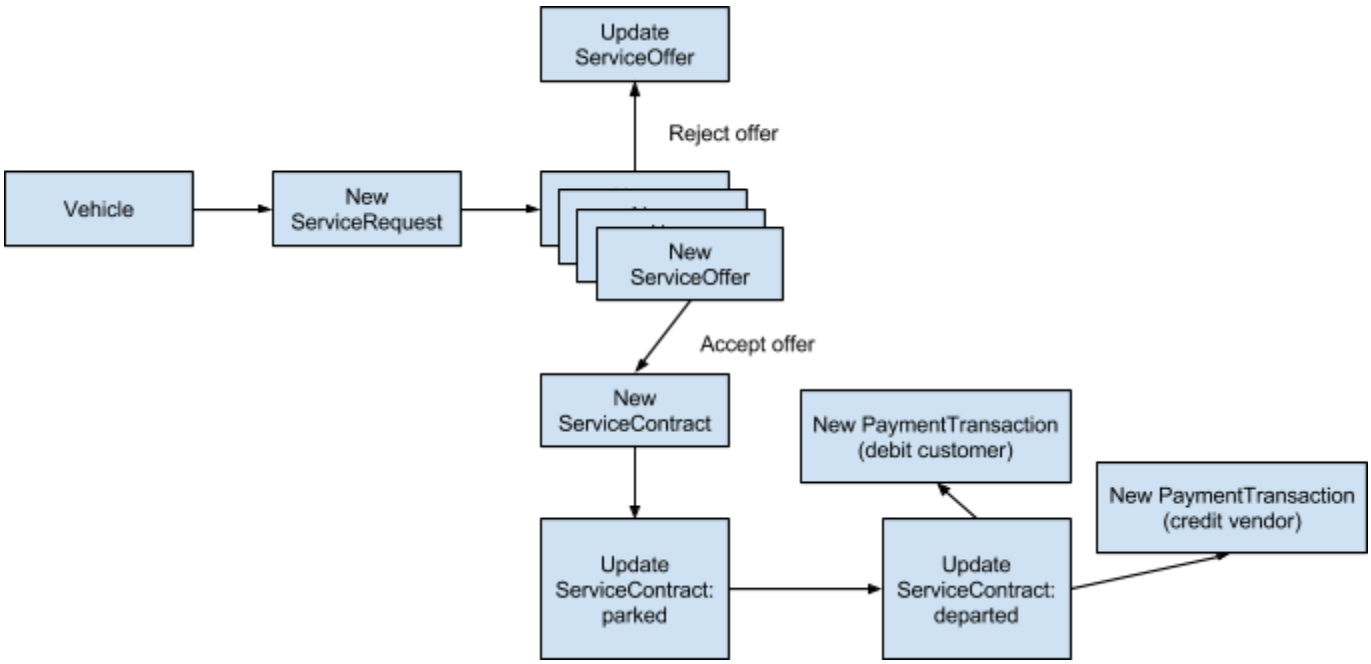
- transaction_id (unique, primary key, indexed)
- creation_date (timestamp)
- updated_at (timestamp)
- sender_type (enum: customer, vendor, system)
- sender_id (FK)

- receiver_type (enum: customer, vendor, system)
- receiver_id (FK)
- contract_id (FK, nullable)
- debit (float)
- credit (float)
- transaction_foreign_id (varchar)

PaymentHold

- hold_id (unique, primary key, indexed)
- creation_date (timestamp)
- customer_id (FK)
- contract_id (FK)
- debit (float)
- credit (float)
- contract_id (FK)

API Flow Diagram



Customer-Facing Endpoints

POST /customer/login

Parameters

- email AND password
- OR
- customer_token

Returns

- JSON customer object (without password field)
- List of customer's vehicles
- A login_token

Discussion

- A customer can login with their email/password combination OR a valid login_token if they already have one.
- A login_token should be revocable by the server. You may need another table for this, or use an existing gem. Implementor's discretion.
- If login fails (invalid token OR invalid credentials) server should return appropriate HTTP error code.

POST /customer/forgot_password/

Parameters

- email

Returns

- HTTP 200 if email sent or user not found
- HTTP 400 if server error occurs

Discussion

- If the email exists as a customer account, send them the reset password email (which includes personalized/tokenized link to reset password). Password reset will be handled via website, not API.
- If the email does not exist as a customer account, return HTTP200 also. **Important: an unrecognized email should not be an error case for security reasons**. If email does not exist send an invitation email to that email address asking them to create an account.

GET /customer/

Parameters

- customer_token

Returns

- JSON customer object if login_token valid

- Appropriate HTTP error code if login_token invalid

POST /customer/

Parameters

- customer_name
- email
- password

Returns

- Appropriate HTTP code
- JSON customer object if creation successful

Discussion

- This endpoint creates a new customer account, with an initial balance of zero.

PUT /customer/

Parameters

- customer_token
- email AND/OR password

Returns

- Appropriate HTTP code
- Updated JSON customer object
- List of customer's vehicles

Discussion

- This endpoint updates email or password fields to their new values.
- Fields not specified should not be altered.

POST /customer/logout

Parameters

- customer_token

Returns

- Appropriate HTTP code

Discussion

- This endpoint validates the login_token and revokes it in order to log a customer out.

GET/customer/vehicles

Parameters

- customer_token

Returns

- JSON list of customer vehicles

Discussion

- This endpoint gets a list of all vehicles for the customer

POST /customer/vehicles

Parameters

- customer_token
- license_plate

Returns

- Appropriate HTTP code
- JSON vehicle object for the vehicle just created

Discussion

- This endpoint creates a new vehicle.
- This endpoint should error out if the customer already has a vehicle with the same license plate value. License plate value should not be case-sensitive.

DELETE /customer/?vehicle_id

Parameters

- customer_token

Returns

- Appropriate HTTP code

Discussion

- This endpoint deletes the vehicle specified by vehicle_id
- This endpoint should error out if the vehicle_id does not exist, or if the vehicle_id does not belong to the current user.

POST /service_request

Parameters

- customer_token

- vehicle_id
- latitude
- longitude
- client_guid

Returns

- Appropriate HTTP code
- The JSON service request object, with an appropriate expiry (this is determined as a config in the Rails app).

Discussion

- This creates a new ServiceRequest with the current time for creation_date. The expiry should be the creation_date + a value determined in the configuration. Vendors have until this expiry to respond to a service request.
- This endpoint should de-duplicate and issue an appropriate error if this service request has the same client_guid as an existing service request.

GET /service_request/?request_id

Parameters

- customer_token

Returns

- JSON service request object for the specified request_id
- List of all ServiceOffers associated with this request

Discussion

- This endpoint should error if request_id does not exist, or does not belong to the logged in user.

POST /service_offer/?offer_id/accept

Parameters

- customer_token

Returns

- Appropriate HTTP code
- JSON service contract object for the created service contract

Discussion

- This endpoint should create a new ServiceContract in pending state for the specified offer_id.
- The offer_id being accepted should have its state set to ACCEPTED.
- All other offer_ids associated with the ServiceRequest should have its state set to REJECTED.

- Vendors who have been rejected should receive a push notification informing them.
- Vendor who has been accepted should receive a confirmation push notification.
- The endpoint should fail if **available customer balance** is below **hold amount**. An appropriate JSON error should be returned indicating insufficient balance, as well as the **hold amount** required for this offer.
- If customer passes the threshold check, a PaymentHold should be created debiting the **hold amount** from the customer, referencing the relevant ServiceContract object.
- Customer balance and available balance should be recalculated.

POST /service_contract/?contract_id/cancel

Parameters

- customer_token
- contract_id

Returns

- Appropriate HTTP code

Discussion

- This endpoint is used to cancel a ServiceContract. Upon successful cancellation, a new PaymentHold object should be created to reverse the **hold amount** previously deducted from customer. (e.g., if a debit of 1000 occurred, there should be a 1000 credit).
- Customer balance and available balance should be recalculated.

GET /customer/transactions

Parameters

- customer_token
- since (timestamp, optional)

Returns

- List of PaymentTransactions created or updated since the “since” parameter
- If “since” is not provided, list all transactions.

Discussion

- This endpoint should error if request_id does not exist, or does not belong to the logged in user.

Attendant-Facing Endpoints

POST /vendor_user/login

Parameters

- vendor_user_token OR
- email & password

Returns

- JSON object for VendorUser

Discussion

- This endpoint logs in a vendor_user either with an existing vendor_user_token or their credentials. It errors out if provided information is invalid.

POST /vendor_user/logout

Parameters

- vendor_user_token

Returns

- Appropriate HTTP code

Discussion

- This endpoint logs out an existing vendor_user and revokes their token.

GET /vendor_user

Parameters

- vendor_user_token

Returns

- JSON object for logged in vendor_user

Discussion

- None.

PUT /vendor_user

Parameters

- vendor_user_token
- email AND/OR password

Returns

- JSON object for updated vendor_user

Discussion

- This endpoint updates the user information for a vendor_user. It errors if token is invalid or if email does not pass validation.

POST /vendor_user

Parameters

- vendoruser_name
- email
- password
- invitation_token

Returns

- JSON object for newly created vendor_user

Discussion

- This endpoint should fail if email does not pass validation or if any field is missing.
- This endpoint should ensure a valid invitation_token is passed. Invitation tokens are generated by the vendor and sent via email.

GET /service_request/

Parameters

- vendor_user_token
- latitude
- longitude

Returns

- List of all service_requests not yet expired that is within range of the provided latitude and longitude

Discussion

- “within range” should be a configuration option within the Rails app

POST /service_request/?request_id/accept

Parameters

- vendor_user_token

Returns

- JSON ServiceOffer object just created.

Discussion

- Creates a new ServiceOffer object based on the accepted request_id.
- The terms_id used should be the latest VendorTerms associated with the vendor where enabled = true.

GET /service_contract/

Parameters

- vendor_user_token
- since (timestamp, optional)

Returns

- List of all service_contracts for this vendor, updated or created from the “since” time stamp till now.
- If no “since” is provided, list all.

Discussion

- None.

POST /service_contract/?contract_id

Parameters

- vendor_user_token
- state (enum)

Returns

- Updated JSON ServiceContract object.

Discussion

- This endpoint should error if an illegal state transition is specified (states can only go from pending -> parked/noshow/cancelled -> departed) and never backwards.
- If the new state is “parked” - updated the parked_at timestamp.
- If the new state is “departed” - update the departed_at timestamp.
- If a PaymentHold was created for this service_contract earlier, reverse it by creating a PaymentHold credit object. e.g., if 1000 was debited in PaymentHold relating to this contract_id, there should be a 1000 credit inserted.
- If the new state is “departed”, generate the appropriate PaymentTransaction debiting the **total cost** from customer. The vendor should receive a credit for the **vendor cost**. Parqueoya should receive a credit for the **parqueoya margin**
- **IMPORTANT: The reversal of the PaymentHold and the creation of the PaymentTransaction must be atomic.**

Vendor Endpoints

POST /vendor/login

Parameters

- vendor_token OR
- email & password

Returns

- JSON object for Vendor

Discussion

- This endpoint logs in a vendor either with an existing vendor_token or their credentials. It errors out if provided information is invalid.

POST /vendor/logout

Parameters

- vendor_token

Returns

- Appropriate HTTP code

Discussion

- This endpoint logs out an existing vendor and revokes their token.

GET /vendor

Parameters

- vendor_token

Returns

- JSON object for logged in vendor.
- List of all vendor_users associated with this vendor.

Discussion

- None.

PUT /vendor

Parameters

- vendor_token

- email AND/OR password

Returns

- JSON object for updated vendor

Discussion

- This endpoint updates the user information for a vendor. It errors if token is invalid or if email does not pass validation.

POST /vendor

Parameters

- vendor_name
- email
- password
- latitude
- longitude
- vendor_description

Returns

- JSON object for newly created vendor

Discussion

- This endpoint should fail if email does not pass validation or if any field is missing.

POST /vendor/invite

Parameters

- vendor_token
- email

Returns

- Appropriate HTTP code

Discussion

- Sends an email to the specified email address containing a link with an embedded invitation_code which can be used to sign up a vendor_user. Validation and storage of invitation_codes is at discretion of implementer.

POST /vendor/terms

Parameters

- vendor_token

- hourly_rate
- max_hourly_hours
- flat_rate
- max_flat_hours

Returns

- JSON object for the new terms

Discussion

- This creates a new VendorTerms object with the specified parameters. The parameters need to be validated as below:
    - if hourly_rate != 0, flat_rate *must* be zero.
    - if flat_rate != 0, hourly_rate *must* be zero.
- Any previous VendorTerms objects associated with this vendor should have its enabled flag marked FALSE.

POST /vendor_user/?vendor_user_id/revoke

Parameters

- vendor_token

Returns

- Appropriate HTTP success or failure code.

Discussion

- This unlinks the specified vendor_user from the vendor.
- The endpoint should validate that the vendor_user_id specified is currently linked to the vendor. It should fail if not.

Calculations

Calculating Balance

- **Customer balance** is the sum of all credits and debits in PaymentTransactions relating to the customer
- **Available customer balance** is the sum of all credits and debits in PaymentTransactions + the sum of all credits and debits in PaymentHold relating to the customer
- **Vendor balance** is the sum of all credits and debits in PaymentTransactions relating to the vendor

Hold Amount

- The amount to be held is the maximum that can be incurred under the VendorTerms in question - it should be the **total_cost** as calculated below.

Service Charge

- Profit_margin is defined as a Rails config somewhere, it is a float
- The final ServiceContract cost is calculated as such:
    - if hourly_rate > 0
        - vendor_cost = hourly_rate * (hours between departed_at and parked_at, rounded up)
        - parqueoya_margin = vendor_cost * profit_margin
        - total_cost = vendor_cost + parqueoya_margin
    - if hourly_rate == 0
        - vendor_cost = flat_rate
        - parqueoya_margin = vendor_cost * profit_margin
        - total_cost = vendor_cost + parqueoya_margin

Not Covered in This Doc

- Creation of vendor on the web
- Invitation of vendor users on the web
- Creation of VendorTerms objects.