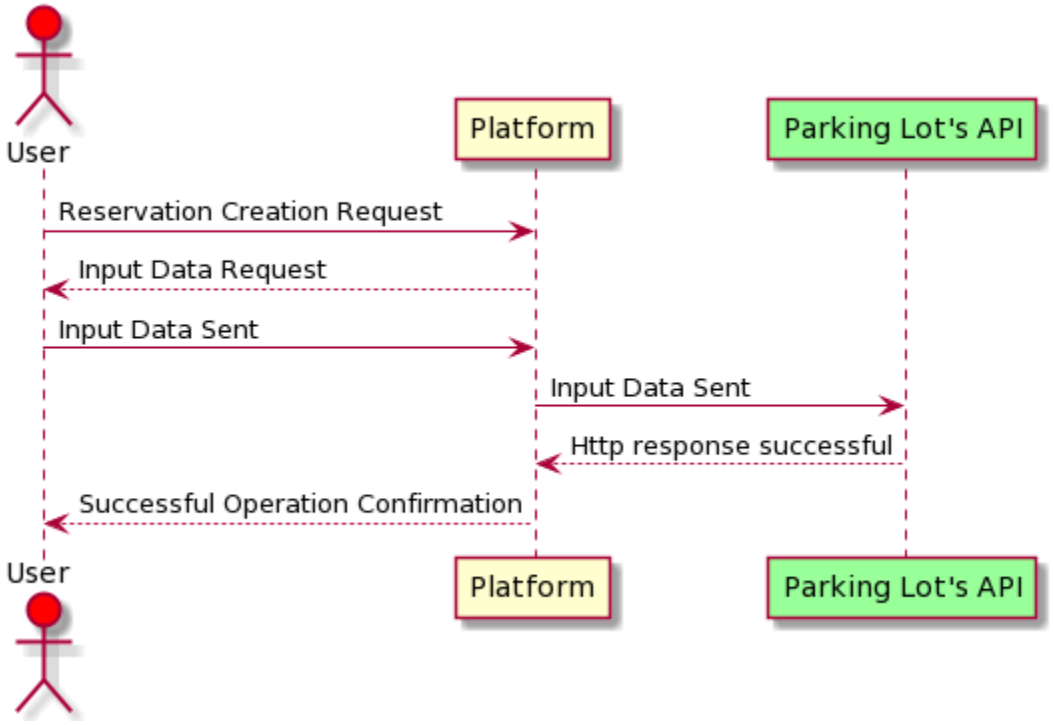
UC 3 – Reservation Management

BRIEF

User accesses main platform so he can manage a reservation. Create a new one, read the details of already made ones, update information about them (Parking spot, Start date, End date) and/or Delete a previously made one. The user selects an operation (CRUD)* and inputs the data required. The Platform sends the data to a Parking Lot's API so it can be stored. The Parking Lot's API sends a HTTP response to the Platform which then sends the confirmation of a successful operation to the User.

*in case of a Create, it is needed the validation of availability of a parking spot.



Fully Dressed

Designation: Create Reservation

Primary Actor: User

Stake Holders and Interests:

- **User:** possibility of user creating a Reservation for a parking spot of his choice.
- **Parking Lot and Platform**: selling their services.

Preconditions:

- User must be authenticated.
- Central API must possess a valid authentication token for the Parking Lot's API so it can be authenticated.
- Parking spot must be available upon reservation creation (can´t be reserved already for that date).

Postconditions:

- Parking Spot is not available for the duration of the reservation.
- Reservation is saved for future consultation.

Main Success Scenario (or standard one):

1. User accesses platform to create a new reservation.
2. Platform asks for the Parking Lot as well the time/date of the reservation.
3. User inputs said data.
4. Platform queries the Parking Lot's API to get a return of the available parking spots for that Parking Lot and date.
5. Parking Lot's API returns all the available parking spots as well as the HTTP response assigned (HTTP 200 OK).
6. Platform displays those parking spots with their details (base price per hour, location) and orders the user input on the duration of said reservation.
7. User inputs the data and confirms.
8. Platform returns the total price for said reservation as well its details and asks for a confirmation.
9. User confirms.

10. Platform sends the data to the Parking Lot's API which in turn gets a HTTP response as confirmation (HTTP 201 Content Created).
11. Platform sends reservation confirmation via email and returns a "Successful Operation" Message to the User.

Extensions (alternative flow):

Platform tries to Read, Update or Delete a reservation that does not exist.

- API returns a HTTP response of "Not Found" (HTTP 404)

User wants to UPDATE a reservation.

1. User accesses platform and picks the reservation he wants to UPDATE.
2. Platform gives the option to change date and or parking lot.
3. User inputs new data.
4. Platform shows the changes and asks user for confirmation.
5. User confirms.
6. Platform sends data to Parking Lot's API to UPDATE the reservation data.
7. API returns a HTTP response of success "No Content" (HTTP 204).
8. Platform displays a "Successful Operation" message to the User.

User wants to Delete A Reservation.

1. User accesses platform and picks the reservation he wants to DELETE.
2. Platform shows its details and asks for a confirmation.
3. User confirms.
4. Platform sends request Parking Lot's API.
5. API returns a HTTP response of success "No Content" (HTTP 204).
6. Platform displays a "Successful Operation" message to the User.

INPUT DATE IS NOT VALID.

- Platform shows a “Wrong Input” message to the user.
  OR
- API returns a HTTP response of “Bad Request” (HTTP 400).

Platform validation TOKEN is faulty and or has insufficient permissions.

1. API returns an “Unauthorized” (HTTP 401) response or “Forbidden” (HTTP 403).