Documentation

Inglorious Developers

Team Members:

Caio Reis

João Martins

Mariana Gomes

Sérgio Pinto

Tiago Azevedo

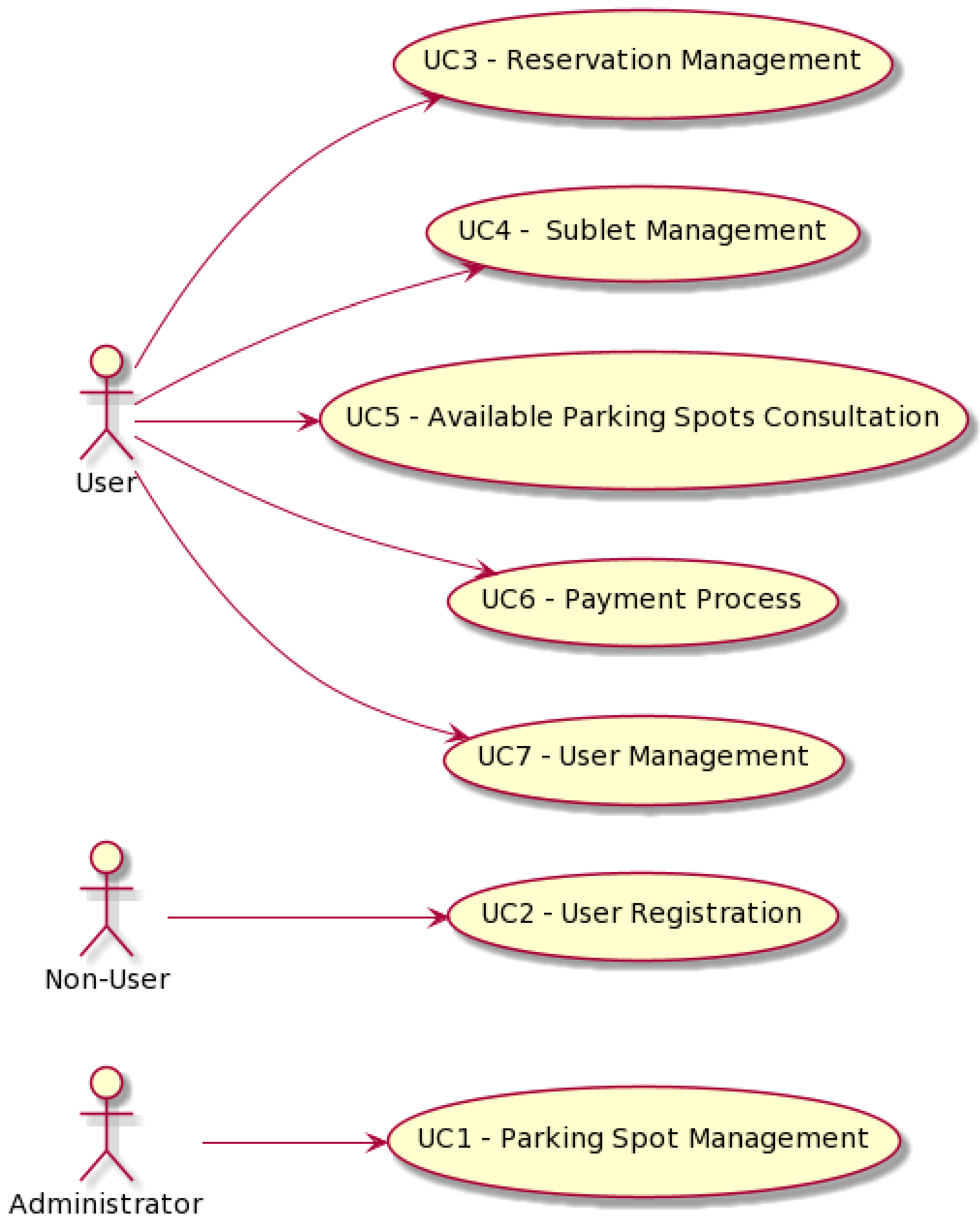
Index

Glossary

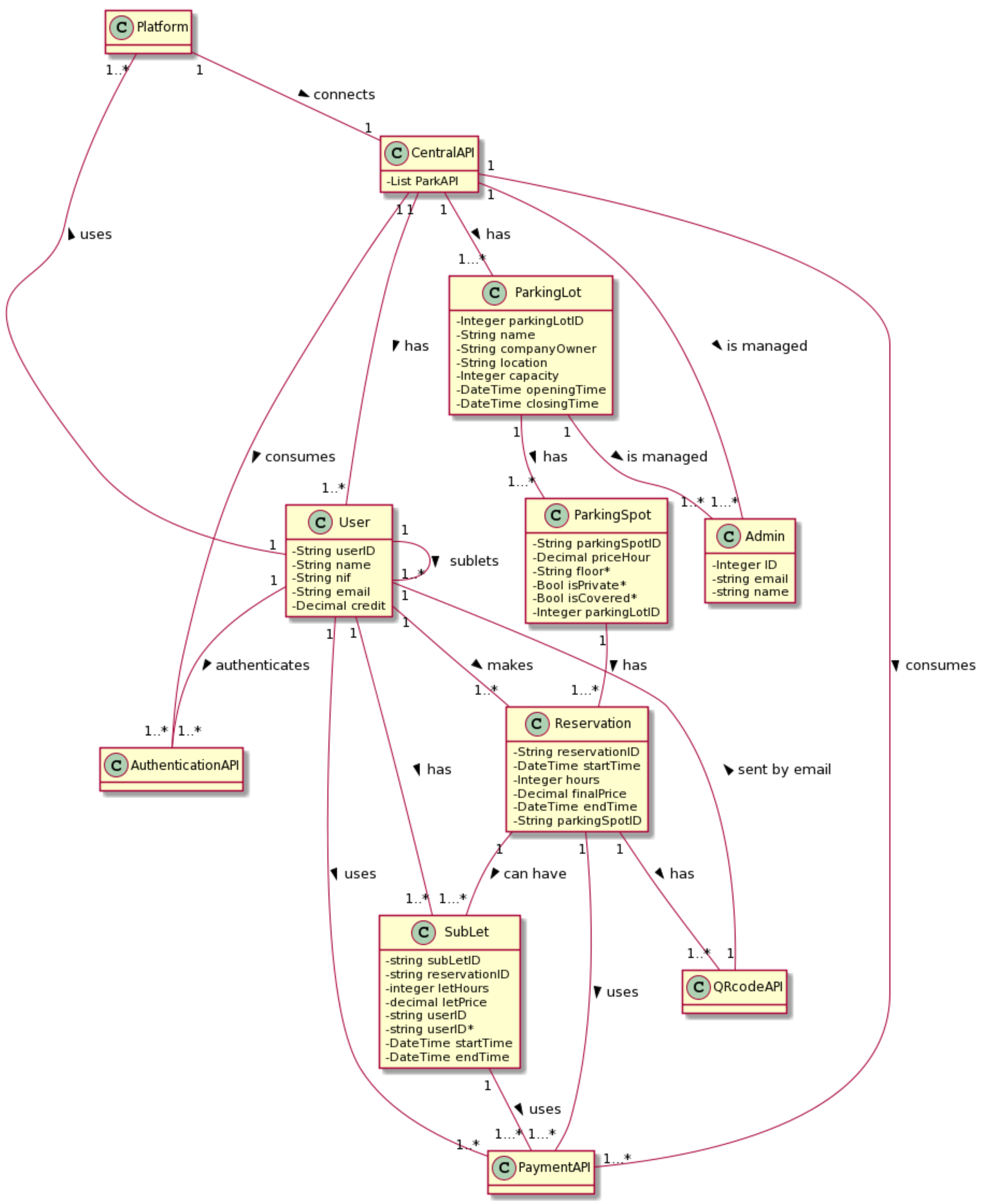
TERM	DESCRIPTION
CENTRAL WEB API	API responsible for consuming Parks API and Public APIs.
CRUD	Create, Read, Update and Delete.
MUNICIPALITY	Territorial district in which a council exercises its jurisdiction.
PARKING SPOT	Delimited and defined area used to park a car.
PAY-AS-YOU-USE	The principle or practice of paying for goods and services at the time of purchase, rather than relying on credit.
PAYMENT METHOD	Ways in which customers pay for their subscription services. A customer can choose a payment method based on your company's list of preferred payment methods.
PREPAID	Pay or arrange to pay beforehand or before due. It's related to the reservation service.
PRIVATE PARK	Car park category owned by a company.
PUBLIC PARK	Car park category owned by the municipality.
QR-CODE	Machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a smartphone.

RESERVATION	Operation done by the user in order to get a parking space in a limited period of time in a specific Parking Lot.
SUBLET	Operation that depends on an existing Reservation. It allows the user to transfer his right of using the space to another user for a limited period of time (never exceeding the period of time of the original reservation).
WEB APP	Application used by users and the client.

Use Cases


UC3 - Reservation Management
UC4 - Sublet Management
UC5 - Available Parking Spots Consultation
UC6 - Payment Process
UC7 - User Management
User
UC2 - User Registration
Non-User
UC1 - Parking Spot Management
Administrator

Domain Model

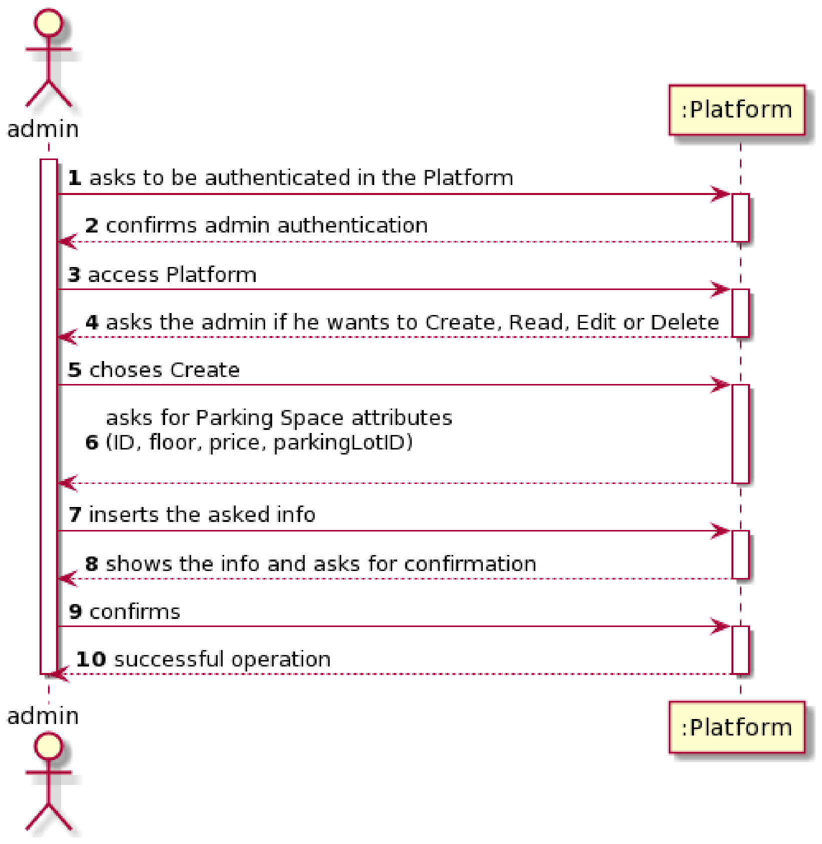


UC1 - Parking Spot Management

Brief

The administrator accesses the platform and authenticates in it. The platform asks what action the administrator intends to perform (Register, Consult, Edit, Delete). The administrator chooses to register a parking space. The platform asks the administrator to fill the required data (single designation, floor, price, parking lot name). The administrator inputs the requested data. The platform shows the data to the administrator asking him to confirm it. The administrator confirms. The platform saves the data and informs the administrator that the operation was successful.

SSD



admin
:Platform
1 asks to be authenticated in the Platform
2 confirms admin authentication
3 access Platform
4 asks the admin if he wants to Create, Read, Edit or Delete
5 choses Create
6 asks for Parking Space attributes (ID, floor, price, parkingLotID)
7 inserts the asked info
8 shows the info and asks for confirmation
9 confirms
10 successful operation
admin
:Platform

Fully Dressed

Name: Parking Spot Management

Primary Actor: Administrator

Stakeholders and interests:

- Administrator: Ability to manage parking spots and relate them to future reservations.
- Parking Lot: Have the Parking spaces full list with their info updated.

Preconditions:

- The Parking Lot to which the Parking Place belongs must exist in the database whether it is public or private.

Postconditions:

- The Parking Lot is updated, according to the Parking Spots that were added by the Administrator.

Main Success Scenario:

1. The administrator starts the authentication process.
2. The Platform allows the administrator to be authenticated.
3. The administrator accesses the platform.
4. The platform asks what action the administrator intends to perform (Register, Consult, Edit, Delete).
5. The administrator chooses to register a parking space.
6. The platform asks the administrator to fill the required data (single designation, floor, price, parking lot name).
7. The administrator inputs the requested data.
8. The platform shows the data to the administrator, asking him to confirm it.
9. The administrator confirms.
10. The platform saves the data and informs the administrator that the operation was successful.

Extensions (alternative scenarios):

***a.**

1. The Administrator can’t authenticate on the platform.
2. The use case ends.

***b.**

1. The administrator asks for the cancelation of the registration of the parking space.
2. The use case ends.

5a. The administrator chooses Consult, Edit, Delete but there are no registered parking spots

1. The platform asks if the Administrator wishes to Consult, Edit or Delete a parking spot.
2. The Administrator chooses an option.
3. The platform asks which parking spot the administrator is interested in.
4. The Administrator inserts the id of the parking spot to be searched.
5. The platform informs the administrator that the parking spot does not exist.
6. The use case ends. Starts again on step 1.

5b. The administrator just wants to consult the data

1. The administrator chooses Consult.
2. The platform shows the list of the registered parking spots.
3. The administrator selects a parking spot.
4. The platform shows the selected parking spot info.
5. The use case ends.

5c. The administrator wants to edit a registered parking spot.

1. The administrator asks to edit a specific parking spot.
2. The platform shows the list of all the parking spots that were registered.
3. The administrator picks one of the parking spots of the list.
4. The platform shows the chosen parking spot info.
5. The administrator edits the info as it wishes.
6. The platform confirms the edited info.
7. The administrator confirms.
8. The platform registers the info and the operation is successful.
9. The use case ends.

5d. The administrator wants to delete a parking spot

1. The administrator selects Delete.
2. The platform shows the list of all the parking spots that were registered.
3. The administrator selects a parking spot.
4. The platform shows the selected parking spot info and asks the administrator to confirm the action.
5. The administrator confirms he wants to delete that parking spot.
6. The platform deletes the selected parking spot info.
7. The use case ends.

8a. The platform detects that the parking spot Unique Designation already exists.

1. The platform alerts the administrator to that fact.
2. The administrator does not change the entered Unique Designation.
3. The use case ends.

Special Requirements:

Variations in Technology and Data: N/A

Frequency of Occurrence: Whenever a parking place needs to be registered.

Miscellaneous:

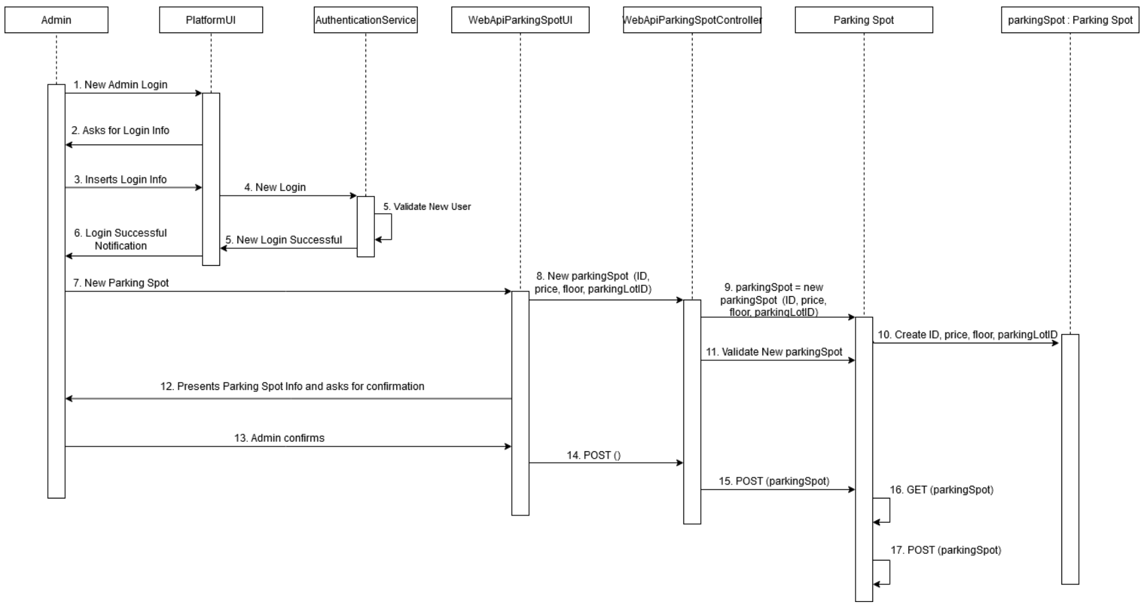
Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1.The administrator starts the authentication operation.	...is responsible for the authentication?	JWToken	
	...interacts with the administrator?	CentralAPIUI	
2.The Platform allows the administrator to be authenticated.	...shows the administrator that the authentication	AuthenticationService (?) / AuthenticationController (?) CentralAPIUI	

	was or not successful?		
3.The administrator accesses the platform.	...interacts with the administrator?	CentralAPIUI	
4.The platform asks what action the administrator intends to perform (Register, Consult, Edit, Delete).	...is responsible for the Register action?	CentralAPI	
	...is responsible for the Consult action?	CentralAPI	
	...is responsible for the Edit action?	CentralAPI	
	...is responsible for the Delete action?	CentralAPI	
	...who coordinates the ParkingSpot methods?	ParkingSpotController	
5.The administrator chooses to register a parking space.	...is responsible for creating ParkingSpot instances?	CentralAPI	
6.The platform asks the administrator to fill the required data (single designation, floor, price, car park name).	n/a		
7.The administrator inputs the requested data.	...saves the input data?	CentralAPI ParkingSpot	IE:ParkingLot has ParkingSpot

8.The platform shows the data to the administrator, asking him to confirm it.	...who validates the ParkingSpot data? (local validation)	ParkingSpot	IE: Has its own data
	...who validates the ParkingSpot data? (global validation)	CentralAPI ParkingSpot	IE: In the DM ParkingSpot is registered in ParkingLot IE: ParkingLot is registered in CentralAPI
9.The administrator confirms.	n/a		
10.The platform saves the data and informs the administrator that the operation was successful.	...responsible to save the administrator input data?	ParkingSpot CentralAPI (?) DataBase	

Sequence Diagram

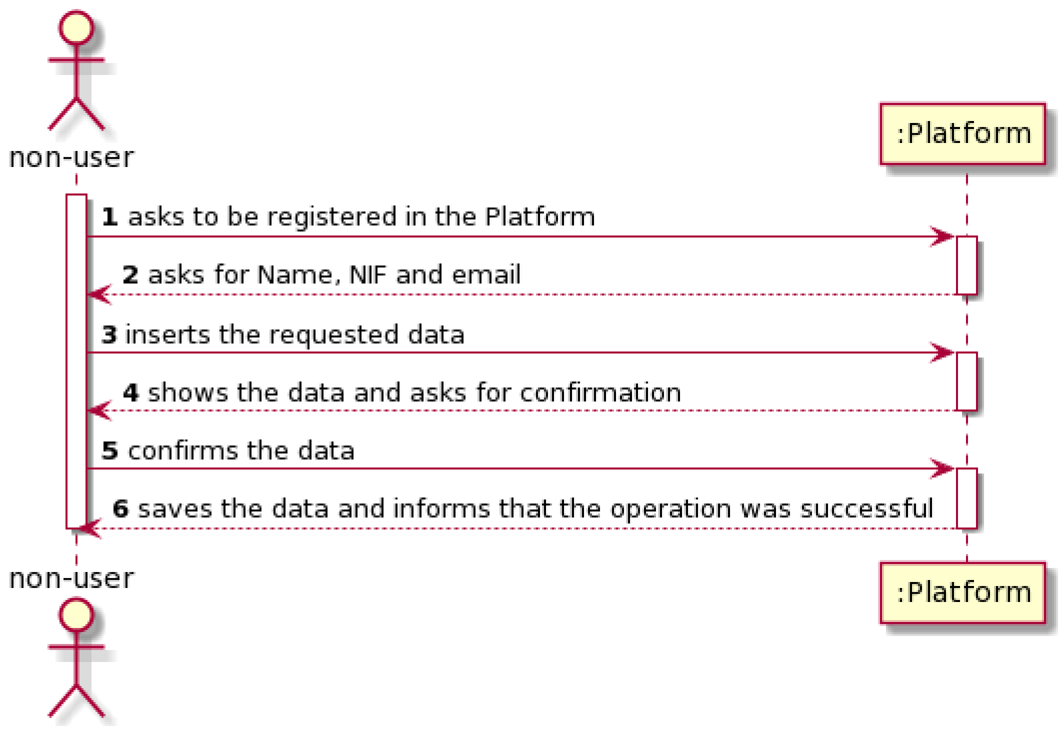

Admin
PlatformUI
AuthenticationService
WebApiParkingSpotUI
WebApiParkingSpotController
Parking Spot
parkingSpot : Parking Spot
1. New Admin Login
2. Asks for Login Info
3. Inserts Login Info
4. New Login
5. Validate New User
5. New Login Successful
6. Login Successful Notification
7. New Parking Spot
8. New parkingSpot (ID, price, floor, parkingLotID)
9. parkingSpot = new parkingSpot (ID, price, floor, parkingLotID)
10. Create ID, price, floor, parkingLotID
11. Validate New parkingSpot
12. Presents Parking Spot Info and asks for confirmation
13. Admin confirms
14. POST ()
15. POST (parkingSpot)
16. GET (parkingSpot)
17. POST (parkingSpot)

UC2 - User Registration

Brief

The non-registered user accesses the platform to register in it. The platform asks the non-registered user for Name, NIF, and email. The not registered user inserts the requested data. The platform shows the data to the non-registered user, asking him to confirm it. The non-registered user confirms. The platform saves the data and informs the new user that the operation was successful.

SSD


non-user
:Platform
1 asks to be registered in the Platform
2 asks for Name, NIF and email
3 inserts the requested data
4 shows the data and asks for confirmation
5 confirms the data
6 saves the data and informs that the operation was successful
non-user
:Platform

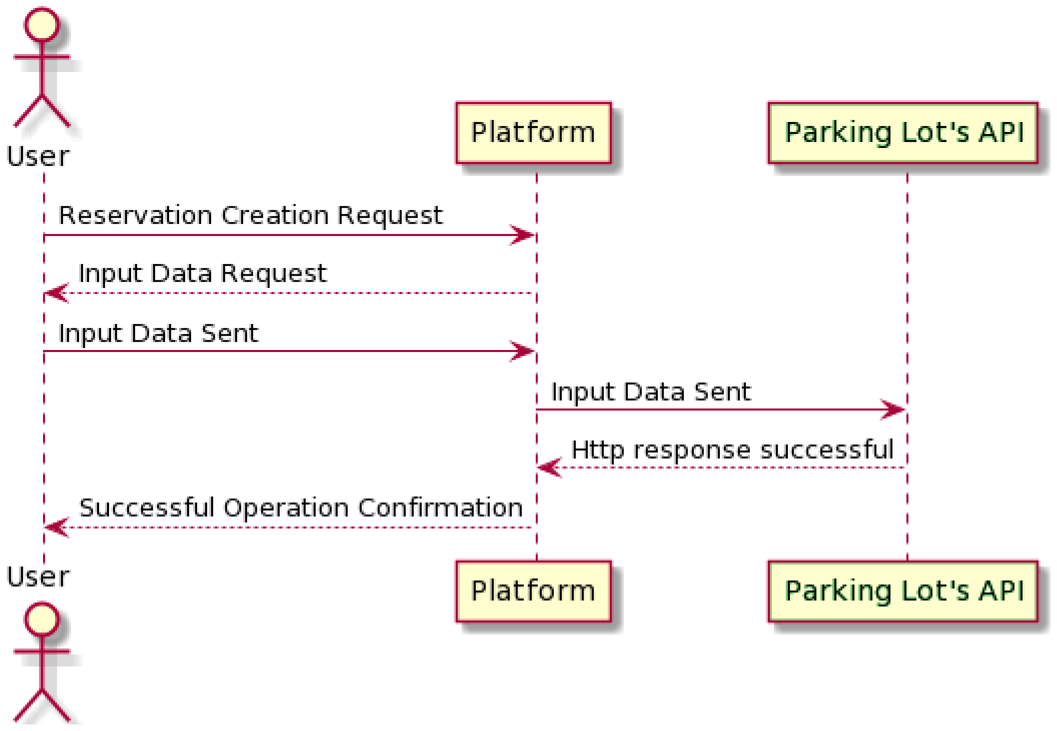
UC3 - Reservation Management

Brief

User accesses the main platform so he can manage a reservation. He can create a new reservation, read the details of already existing ones, update information about them (Parking spot unique designation, start date, hours) and/or Delete. 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 informs the user that the operation was successful.

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

SSD



Fully Dressed

Designation: Create Reservation

Primary Actor: User

Stakeholders and interests:

- **User:** The user can create 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):

*a. **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.

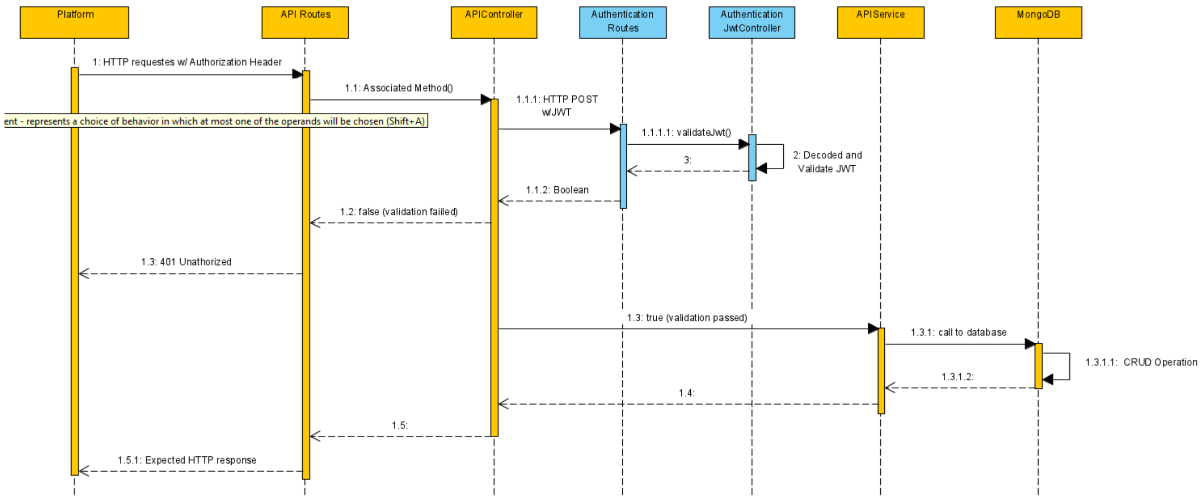
1. API returns an “Unauthorized” (HTTP 401) response or “Forbidden” (HTTP 403).

Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1.User accesses platform to create a new reservation.	... interacts with User?	WebAppUI	
2.Platform asks for the Parking Lot as well the time/date of the reservation	… shows User the options?	WebAppUI	
3. User inputs said data.	…validates data input?	ParkingLotApiController ReservationModel	IE: Knows the Business model and validation for data
4. Platform queries the Parking Lot's API to get a return of the available parking spots for that Parking Lot and date.	... makes the query?	CentralApi	
	... answers the query?	ParkingLotApi	
5. Parking Lot's API returns all the available parking spots as well as the HTTP response assigned (HTTP 200 OK).	…returns data?	ParkingLotApiController	IE: Knows the method to return requested data
		CentralApiController WebAppController	IE: Have the route mapping to the ParkingLotController
6. Platform displays those parking spots with their details (base price per hour, location) and orders the user input on the parking spot of his choice and the	…displays data?	WebbAppUI	

duration of said reservation.			
7. User inputs the data and confirms	N/A		
8.Platform returns the total price for said reservation as well its details and asks for a confirmation.	… returns data?	ParkingLotApiController	IE: Knows the method to return requested data
		CentralApiController WebAppController	IE: Have the route mapping to the ParkingLotController
9.Platform sends the data to the Parking Lot's API which in turn returns an HTTP response as confirmation (HTTP 201 Content Created).	…sends data?	WebApp	IE: Has the route mapping to the CentralApiController
		CentralApiController	IE: Knows the method to store the User info. Has the route mapping to the ParkingLotApiController
		ParkingLotApiController	IE: Knows the method to store reservation info.
	…saves data?	CentralApiDB	IE: Responsible for CentralApi info storage
		ParkingLotApiDB	IE: Responsible for ParkingLotApi info storage
	…returns response?	ParkingLotApiController	

Sequence Diagram

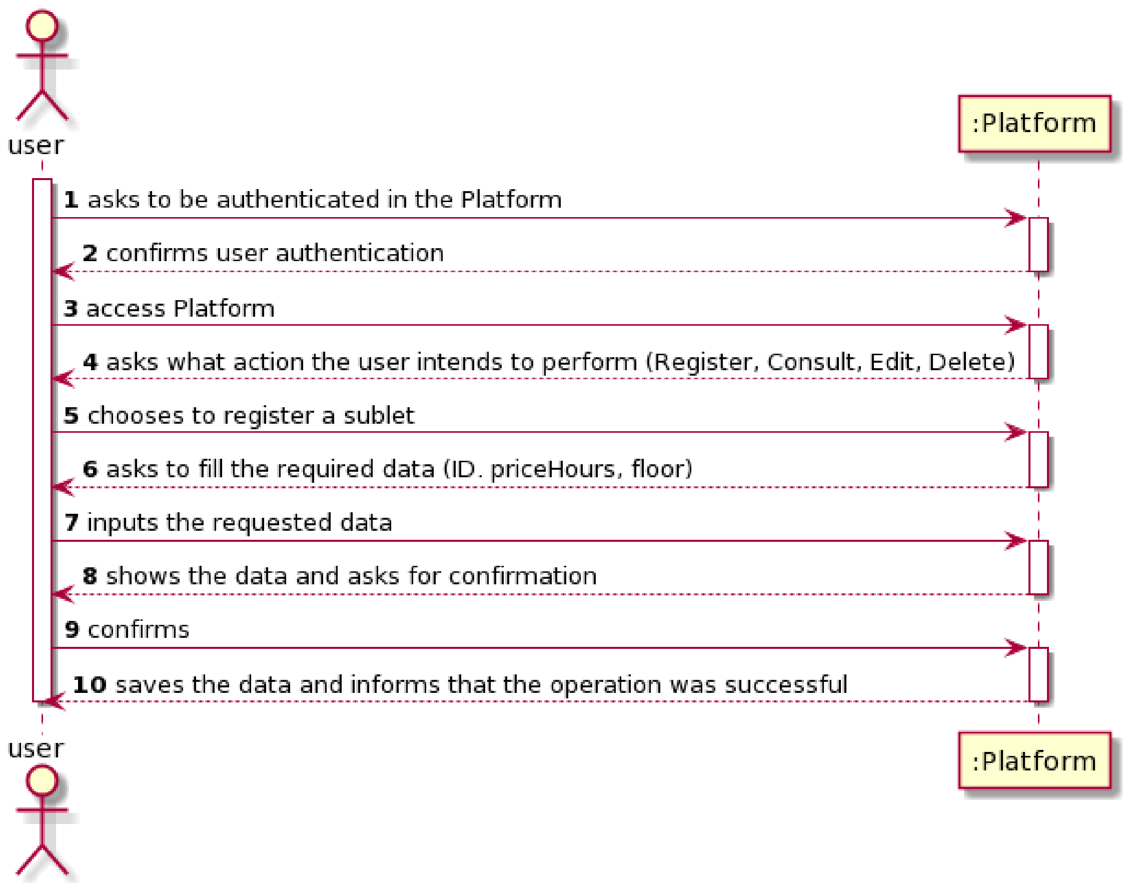

Platform
API Routes
APIController
Authentication Routes
Authentication JwtController
APIService
MongoDB
1: HTTP requestes w/ Authorization Header
1.1: Associated Method()
ent - represents a choice of behavior in which at most one of the operands will be chosen (Shift+A)
1.1.1: HTTP POST w/JWT
1.1.1.1: validateJwt()
2: Decoded and Validate JWT
3:
1.1.2: Boolean
1.2: false (validation failed)
1.3: 401 Unathorized
1.3: true (validation passed)
1.3.1: call to database
1.3.1.1: CRUD Operation
1.3.1.2:
1.4:
1.5:
1.5.1: Expected HTTP response

UC4 - Sublet Management

Brief

The user accesses the platform and authenticates in it. The platform asks what action the user intends to perform (Register, Consult, Edit, Delete). The user chooses to register a sublet. The platform asks the user to fill the required data (reservation identification, let hours, let price, start time). The user inputs the requested data. The platform shows the data to the user, asking him to confirm it. The user confirms. The platform saves the data and informs the user that the operation was successful.

SSD

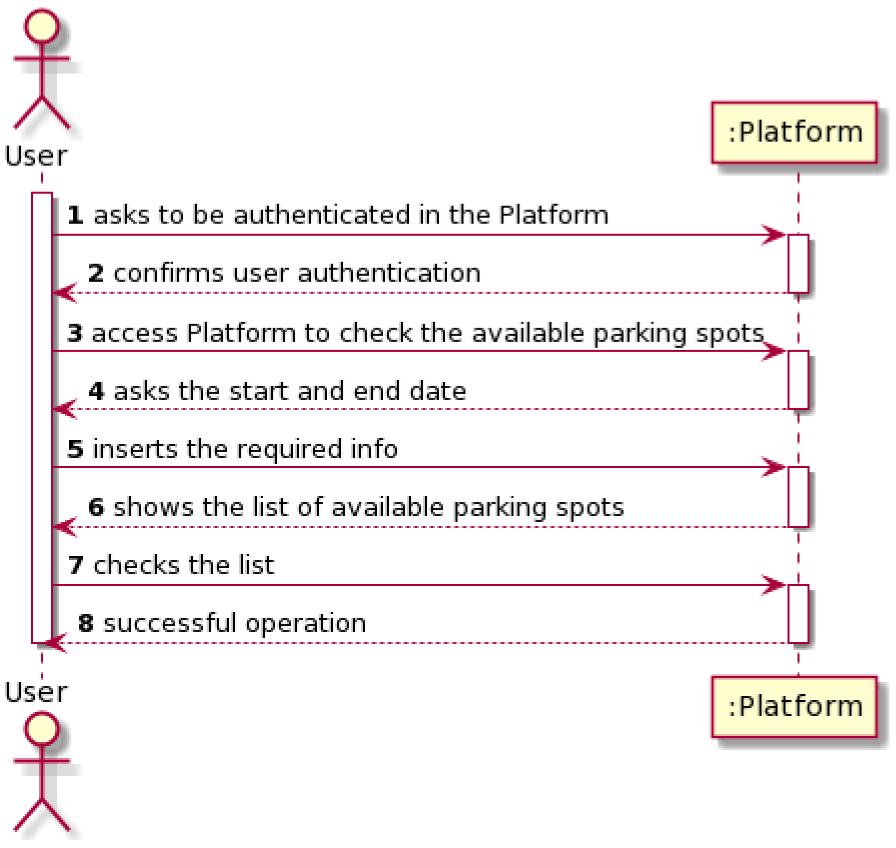

user
:Platform
1 asks to be authenticated in the Platform
2 confirms user authentication
3 access Platform
4 asks what action the user intends to perform (Register, Consult, Edit, Delete)
5 chooses to register a sublet
6 asks to fill the required data (ID. priceHours, floor)
7 inputs the requested data
8 shows the data and asks for confirmation
9 confirms
10 saves the data and informs that the operation was successful
user
:Platform

UC5 - Available Parking Spots Consultation

Brief

The user accesses the platform and authenticates in it in order to check the available parking spots. The platform asks the initial and end dates. The user inputs the required data. The platform shows the list of available parking spots. The user checks the list of the available parking spots. The platform informs the user that the operation was successful. The use case ends.

SSD


User
:Platform
1 asks to be authenticated in the Platform
2 confirms user authentication
3 access Platform to check the available parking spots
4 asks the start and end date
5 inserts the required info
6 shows the list of available parking spots
7 checks the list
8 successful operation
User
:Platform

Fully Dressed

Name: Available Parking Spots Consultation

Primary Actor: User

Stakeholders and interests:

- User: Ability to check available parking spots.
- Parking Lot: Display Parking spaces to potential customers.

Preconditions:

- The Parking Spots must exist in the database whether it is public or private.

Postconditions:

- N/A

Main Success Scenario:

1. The user starts the authentication operation.
2. The Platform allows the user to be authenticated.
3. The user accesses the platform to check the available parking spots.
4. The platform asks the start and end date.
5. The user inserts the requested data.
6. The platform displays the list of the available parking spots to the user.
7. The user checks the list.
8. The platform informs the user that the operation was successful.

Extensions (alternative scenarios):

***a.**

1. The User can’t authenticate on the platform.
2. The use case ends.

***b.**

1. The User cancels the list visualization.
2. The use case ends.

6a. There are no parking spots available

1. The platform informs the user that there are no parking spots available.
2. The use case ends. Starts again on step 1.

Special Requirements:

Variations in Technology and Data: N/A

Frequency of Occurrence: Whenever a user wants to check parking availability.

Miscellaneous:

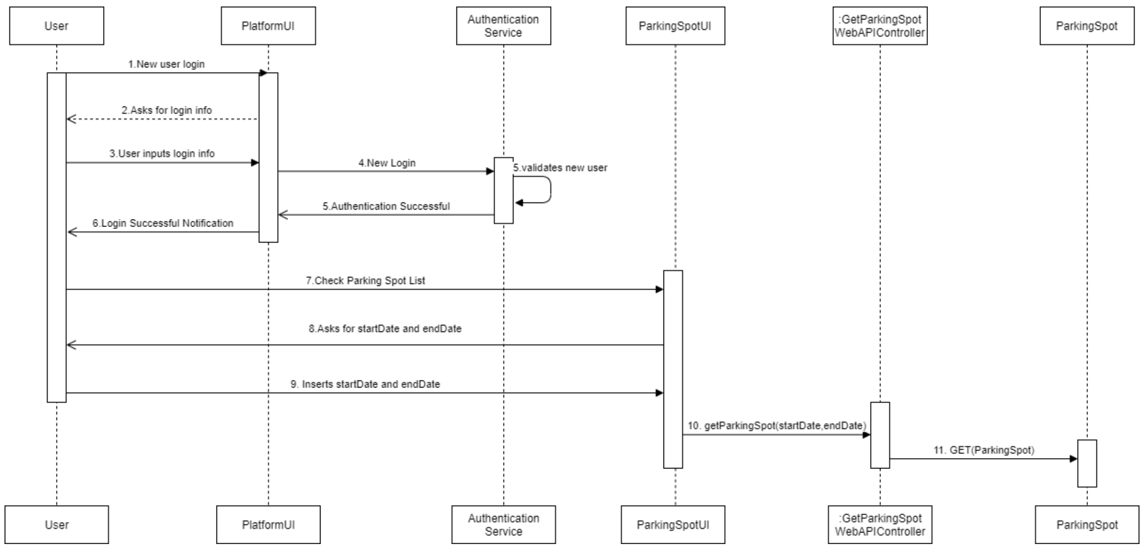
- A non-registered person can check availability?

Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1.The user starts the authentication operation.	...is responsible for the authentication?	JWToken	
	...interacts with the user?	CentralAPIUI	
2.The Platform allows the user to be authenticated	...shows the user that the authentication was or not successful?	AuthenticationService (?) / AuthenticationController (?) CentralAPIUI	
3.The user accesses the platform to check the available parking spots.	...interacts with the user?	CentralAPIUI	
4.The platform asks the start and end date.	n/a		
5.The user inserts the requested data.	...saves the input data?	CentralAPI ParkingLot	IE:ParkingLot has ParkingSpot
6.The platform displays the list of the available parking spots to the user.	…is responsible for displaying the list?	CentralAPI ParkingLot	IE:ParkingLot has ParkingSpot

7.The user checks the list.	n/a		
8.The platform informs the user that the operation was successful.	...who knows the parking spot list?	CentralAPI ParkingLot	IE:ParkingLot has ParkingSpot

Sequence Diagram

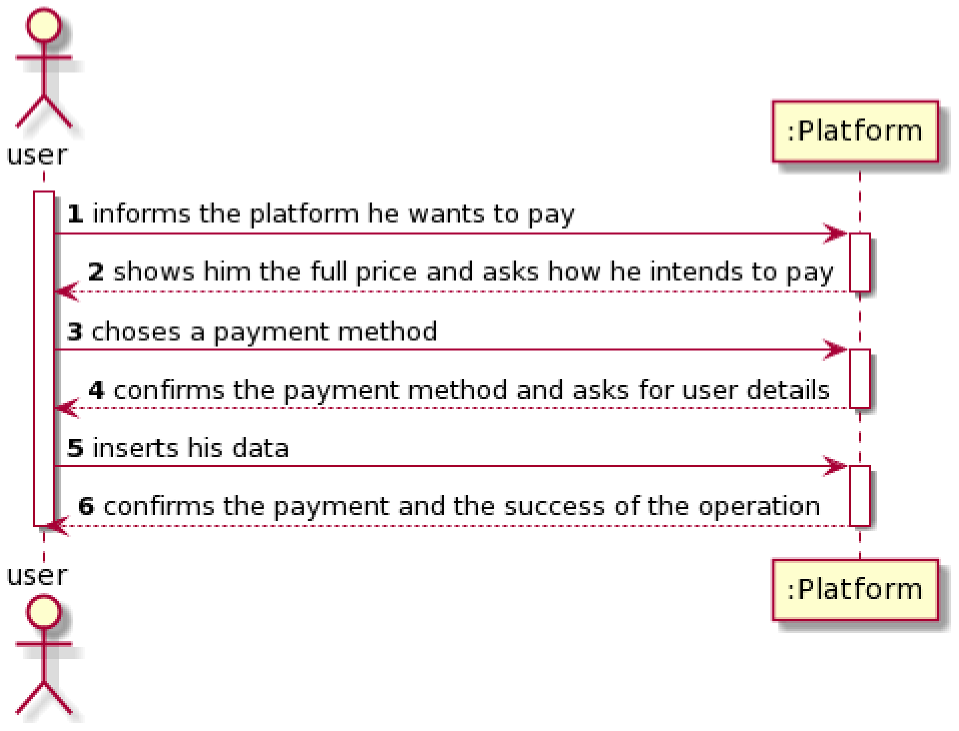

User
PlatformUI
Authentication Service
ParkingSpotUI
:GetParkingSpot WebAPIController
ParkingSpot
1.New user login
2.Asks for login info
3.User inputs login info
4.New Login
5.validates new user
5.Authentication Successful
6.Login Successful Notification
7.Check Parking Spot List
8.Asks for startDate and endDate
9. Inserts startDate and endDate
10. getParkingSpot(startDate,endDate)
11. GET(ParkingSpot)
User
PlatformUI
Authentication Service
ParkingSpotUI
:GetParkingSpot WebAPIController
ParkingSpot

UC6 - Payment Process

Brief

After choosing a parking spot and making a reservation the user proceeds to pay. He informs the platform he wants to pay. The platform shows him the full price and asks how he intends to pay. The user choses a payment method. The platform confirms the payment method and asks for user details. The user inserts his data. The platform confirms the payment and the successful reservation process.

SSD

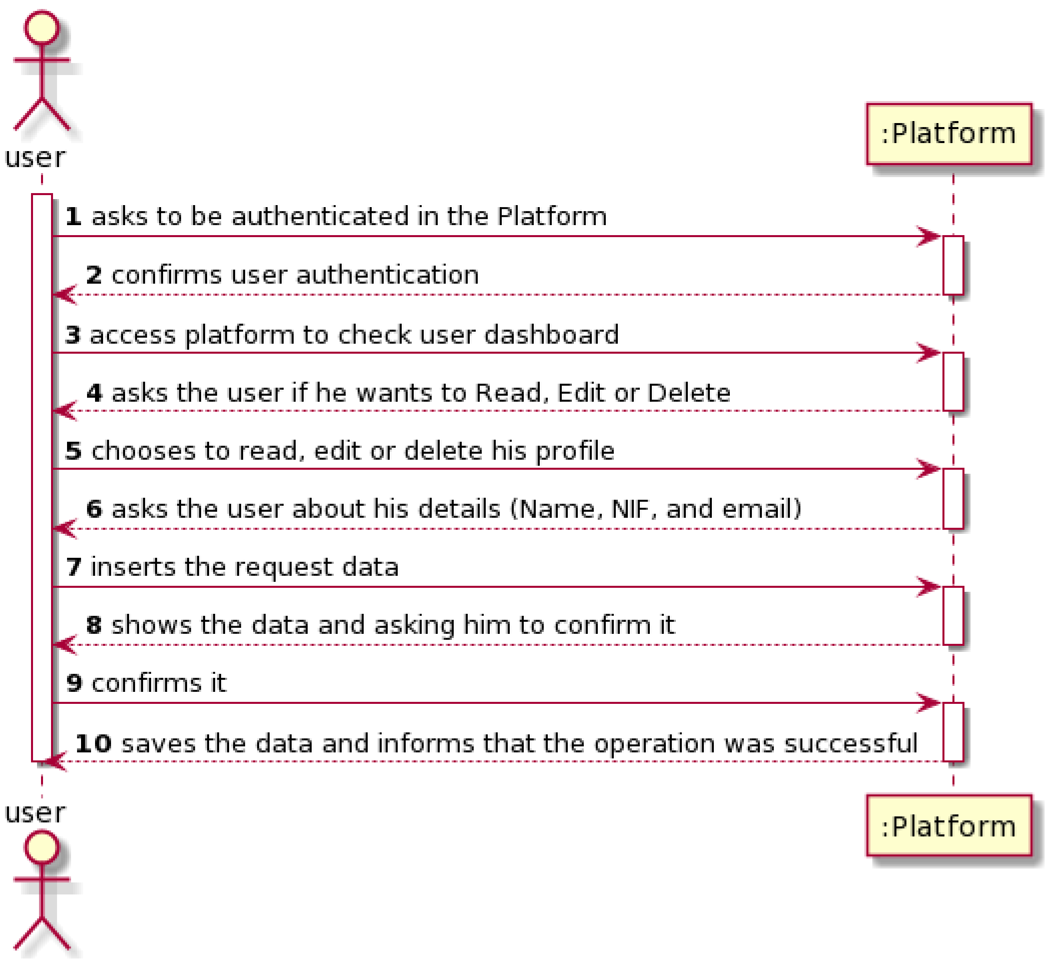

user
:Platform
1 informs the platform he wants to pay
2 shows him the full price and asks how he intends to pay
3 choses a payment method
4 confirms the payment method and asks for user details
5 inserts his data
6 confirms the payment and the success of the operation
user
:Platform

UC7 - User Management

Brief

The user authenticates himself in the platform. The platform confirms his authentication. The user asks the platform to check user dashboard. The platform shows him the requested info. The user chooses to read, edit or delete his profile. The platform asks the user about his details (Name, NIF and email). The user inserts the requested data. The platform shows the data to the user asking him to confirm it. The user confirms. The platform saves the data and informs the new user that the operation was successful.

SSD


user
:Platform
1 asks to be authenticated in the Platform
2 confirms user authentication
3 access platform to check user dashboard
4 asks the user if he wants to Read, Edit or Delete
5 chooses to read, edit or delete his profile
6 asks the user about his details (Name, NIF, and email)
7 inserts the request data
8 shows the data and asking him to confirm it
9 confirms it
10 saves the data and informs that the operation was successful
user
:Platform