Park Around Documentation

Inglorious Developers

Team Members:

Caio Reis

João Martins

Mariana Gomes

Sérgio Pinto

Tiago Azevedo

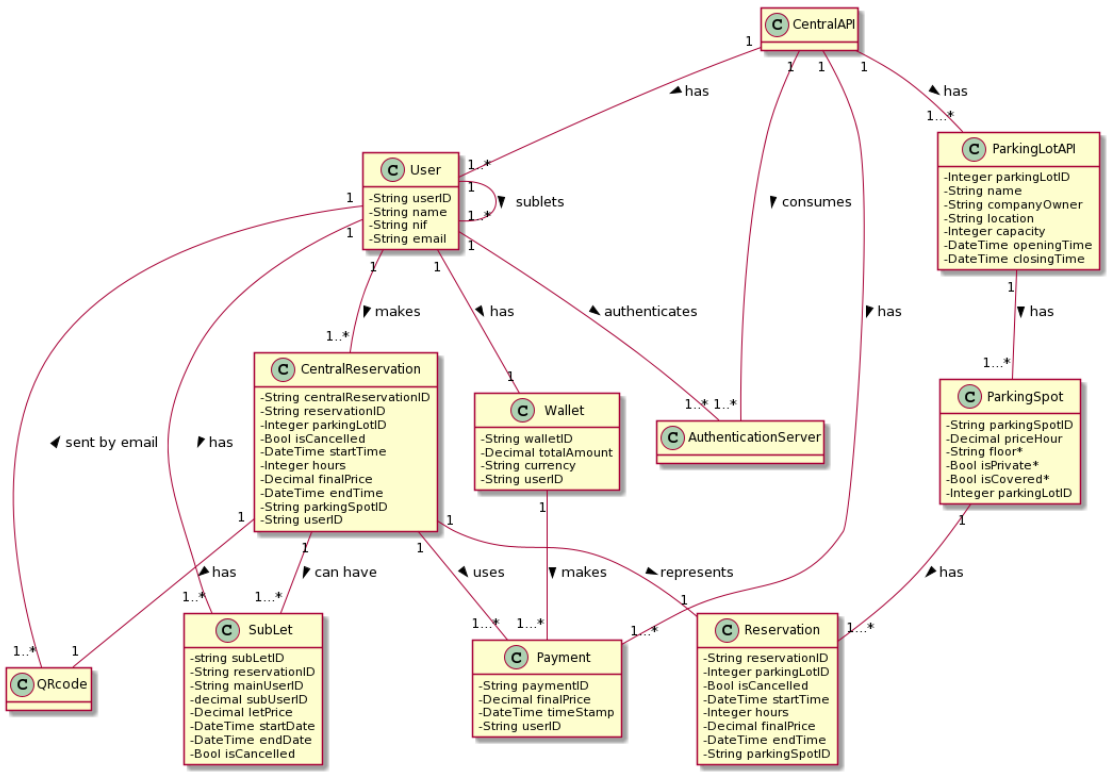
TABLE OF CONTENTS

1. Glossary

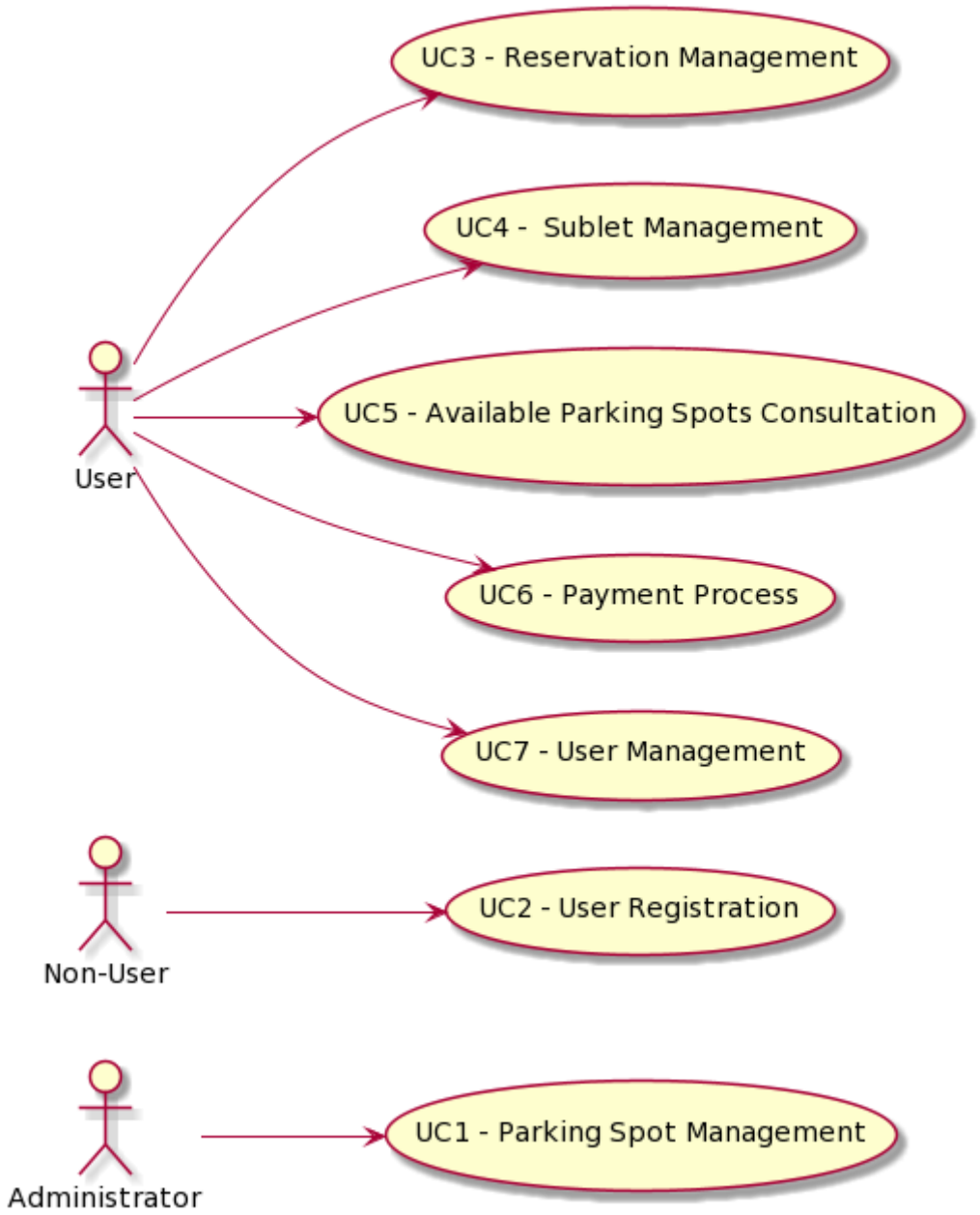
TERM	DESCRIPTION
CENTRAL WEB API	API responsible for consuming Parks API and Public APIs.
CRUD	Create, Read, Update and Delete.
MUNICIPALITY	Territorial district in which a council exercises its jurisdiction.
PARKING SPOT	Delimited and defined area used to park a car.
PAY-AS-YOU-USE	The principle or 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.

2. Domain Model



3. Use Cases

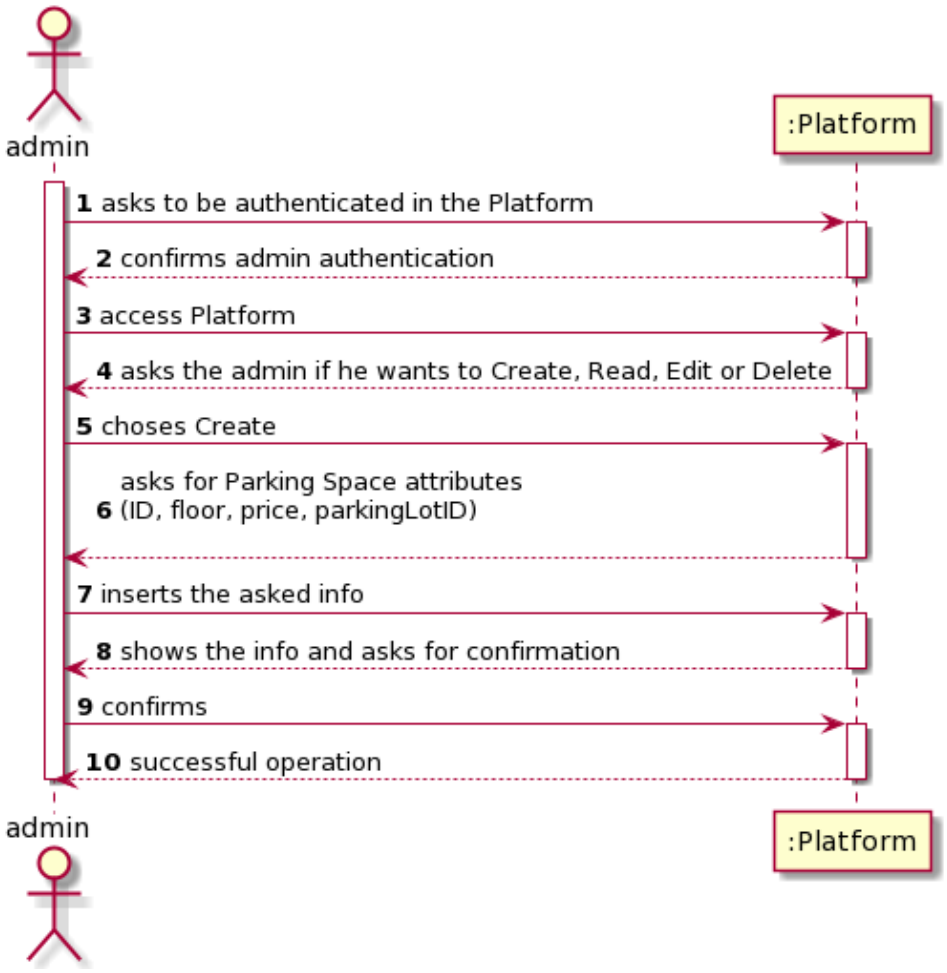

UC3 - Reservation Management
UC4 - Sublet Management
UC5 - Available Parking Spots Consultation
UC6 - Payment Process
UC7 - User Management
UC2 - User Registration
UC1 - Parking Spot Management
User
Non-User
Administrator

3.1. UC1 - Parking Spot Management

3.1.1. 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.

3.1.2. SSD



3.1.3. 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:

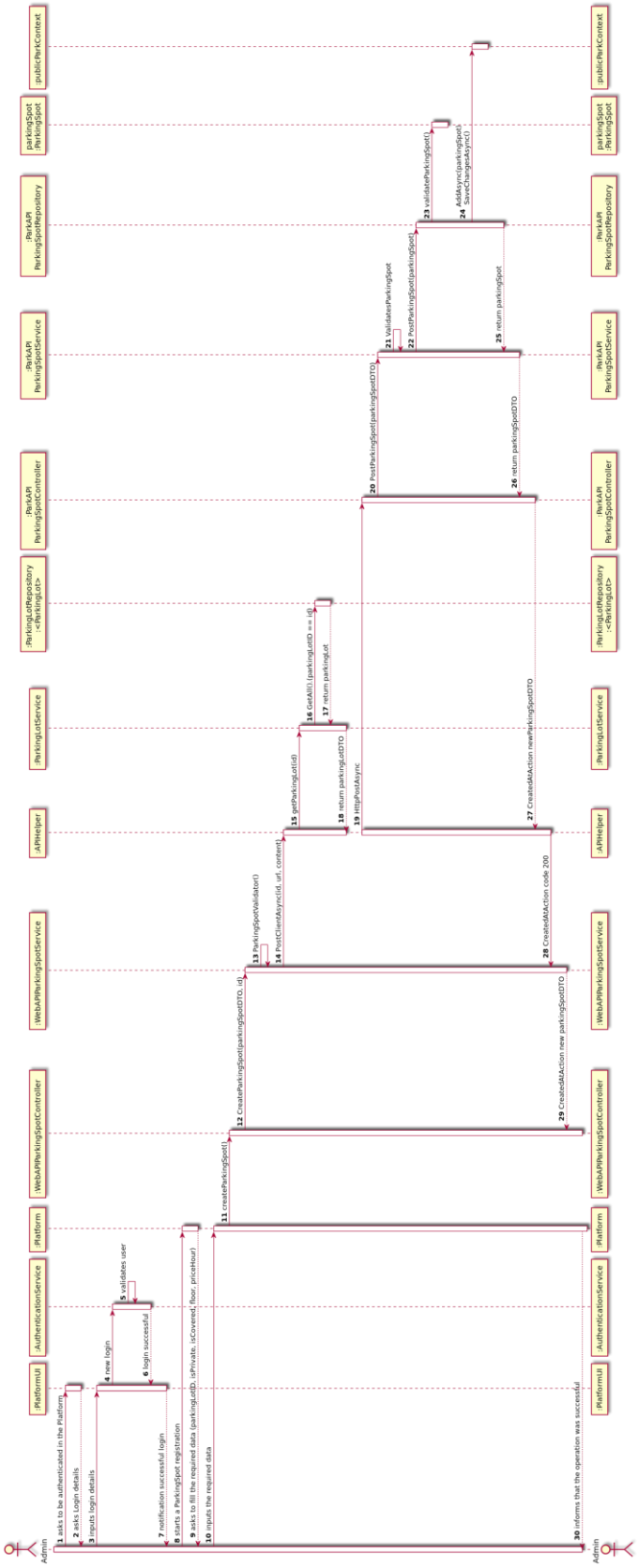
3.1.4. Class Table

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	Pure Fabrication
2.The Platform allows the administrator to be authenticated.	...shows the administrator that the authentication was or not successful?	AuthenticationService (?) / AuthenticationController (?) CentralAPIUI	

3.The administrator accesses the platform.	...interacts with the administrator?	CentralAPIUI	Pure Fabrication
4.The platform asks what action the administrator intends to perform (Register, Consult, Edit, Delete).	...is responsible for the Register action?	CentralAPI RegisterParkingSpotController	Controller
	...is responsible for the Consult action?	CentralAPI GetParkingSpotController	Controller
	...is responsible for the Edit action?	CentralAPI PutParkingSpotController	Controller
	...is responsible for the Delete action?	CentralAPI DeleteParkingSpotController	Controller
	...who coordinates the ParkingSpot methods?	ParkingSpotController	Controller
5.The administrator chooses to register a parking space.	...is responsible for creating ParkingSpot instances?	CentralAPI	Creator(rule1)
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	...who validates the ParkingSpot data?	ParkingSpot	IE: Has its own data

to the administrator, asking him to confirm it.	(local validation)		
	...who validates the ParkingSpot data? (global validation)	CentralAPI ParkingLot	IE: In the DM ParkingLot has registered ParkingSpots 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?	ParkingLot CentralAPI (?) DataBase	IE: In the DM ParkingLot has registered ParkingSpots

3.1.5. Sequence Diagram

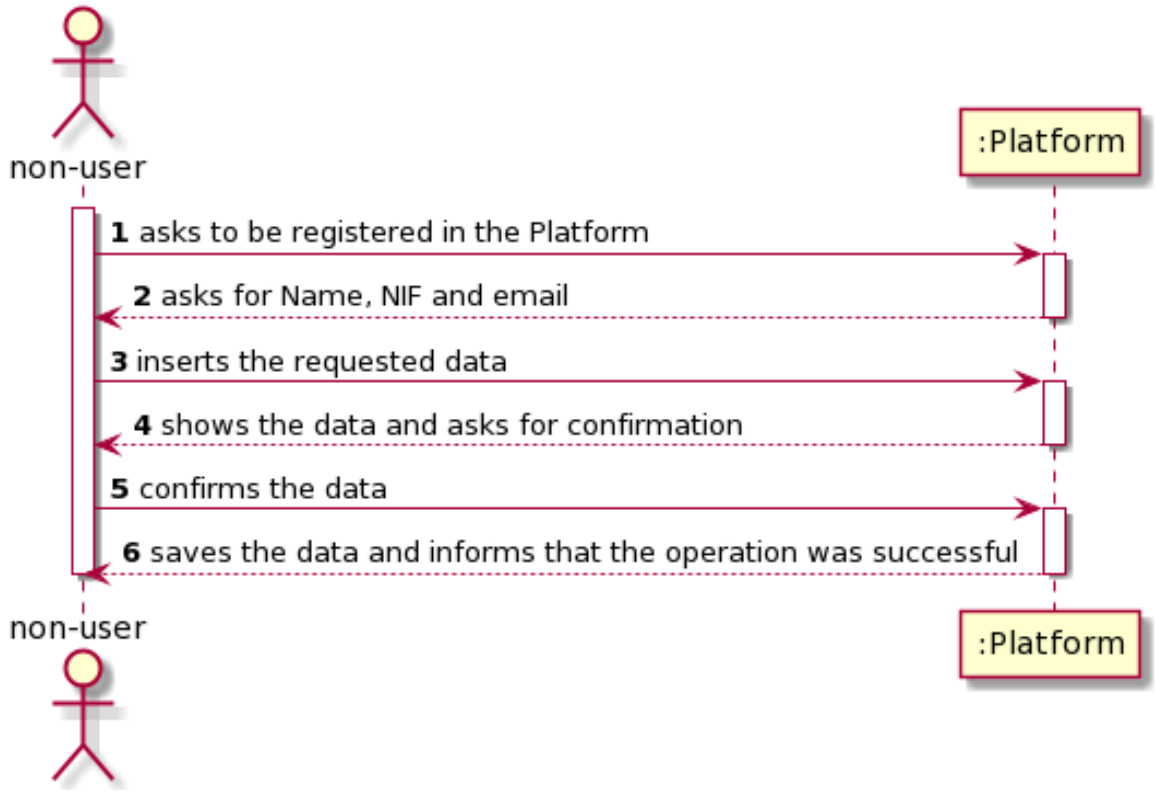


3.2. UC2 - User Registration

3.2.1. 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.

3.2.2. SSD



3.2.3. Fully Dressed

Name: User Registration

Primary Actor: Non-user

Stakeholders and interests:

- Non-User: Have the ability to register in the platform and being able to use the features it provides.
- Platform: Have registered users using the platform.

Preconditions: n/a

Postconditions:

- The Users list is updated according to the details provided by the user himself.

Main Success Scenario:

1. The non-user starts the registration process.
2. The Platform asks for its details (Name, nif and email).
3. The non-user inserts the requested data.
4. The platform shows the data to the non-user, asking him to confirm it.
5. The non-user confirms.
6. The platform saves the data and informs the non-user that the operation was successful and that he became a platform registered user.

Extensions (alternative scenarios):

***a.**

1. The non-user cancels the registration process.
2. The use case ends.

3a.

3. The non-user inserts non valid data.
4. The platform warns him about it.
5. The platform allows the non-user to change the data.
6. The non-user does not change the data.
7. The use case ends.

3b.

3. Required data was not inserted
4. The platform informs which details are required.
5. The non-user does not input the required data.
6. The use case ends.

6a. The platform detects that the user details already exist in the platform belonging to another user.

7. The platform alerts the non-user to that fact.
8. The platform allows the non-user to change the input details.
9. The non-user does not change the entered duplicated details.
10. The use case ends.

Special Requirements:

Variations in Technology and Data: N/A

Frequency of Occurrence: Whenever a non-user wants to get registered in the platform.

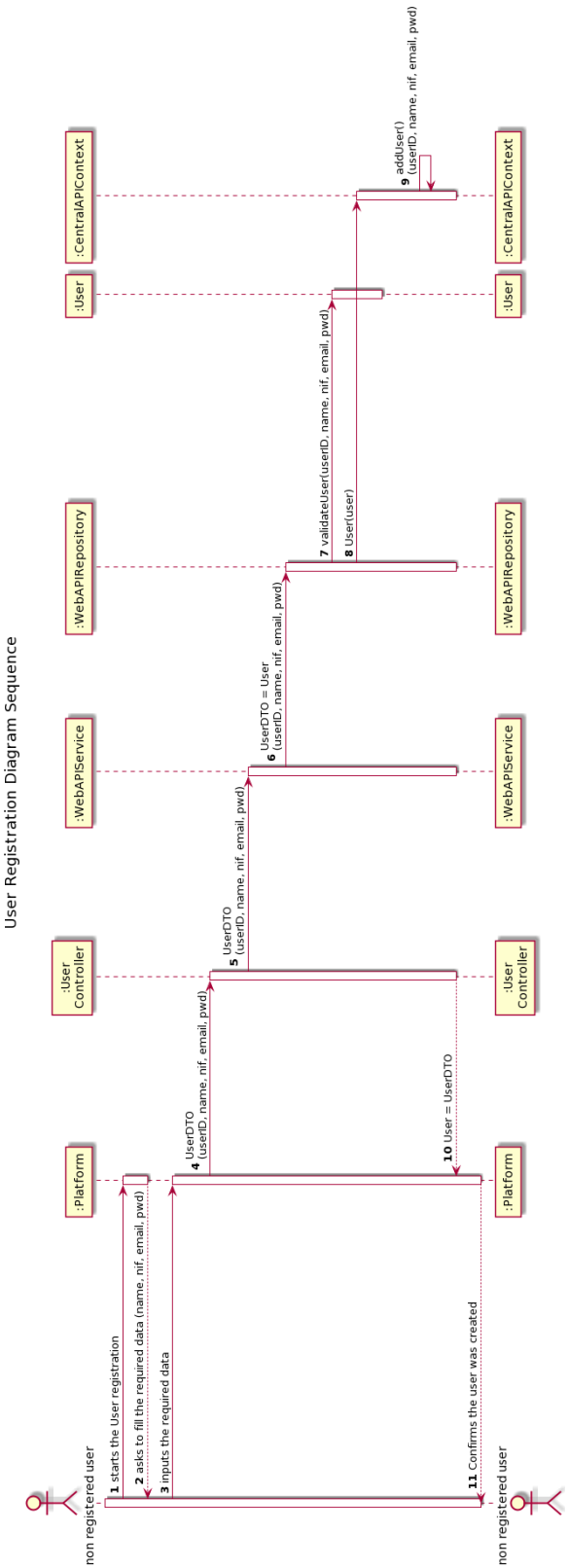
Miscellaneous:

3.2.4. Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1.The non-user starts the registration process.	...interacts with the non-user?	CentralAPIUI	Pure Fabrication
	...is responsible for the Register action?	CentralAPIUI RegisterUserController	Controller
	...creates User instances?	CentralAPI	Creator(rule1)
2.The Platform asks for its details (Name, nif and email).	n/a		
3.The non-user inserts the requested data.	…saves the inserted details?	CentralAPI User	IE: Instance created in step 1.

4.The platform shows the data to the non-user, asking him to confirm it.	...validates the user details? (local validation)	User	IE: Has its own data
	...validates the user details? (global validation)	CentralAPI	IE: Has registered users
5.The non-user confirms.	n/a		
6.The platform saves the data and informs the non-user that the operation was successful and that he became a platform registered user.	...saves the registered user?	CentralAPI	IE: In the DM CentralAPI has registered users.

3.2.5. Sequence Diagram


User Registration Diagram Sequence
non registered user
:Platform
:User Controller
:WebAPIService
:WebAPIRepository
:User
:CentralAPIContext
1 starts the User registration
2 asks to fill the required data (name, nif, email, pwd)
3 inputs the required data
4 UserDTO (userID, name, nif, email, pwd)
5 UserDTO (userID, name, nif, email, pwd)
6 UserDTO = User (userID, name, nif, email, pwd)
7 validateUser(userID, name, nif, email, pwd)
8 User(user)
9 addUser() (userID, name, nif, email, pwd)
10 User = UserDTO
11 Confirms the user was created
non registered user

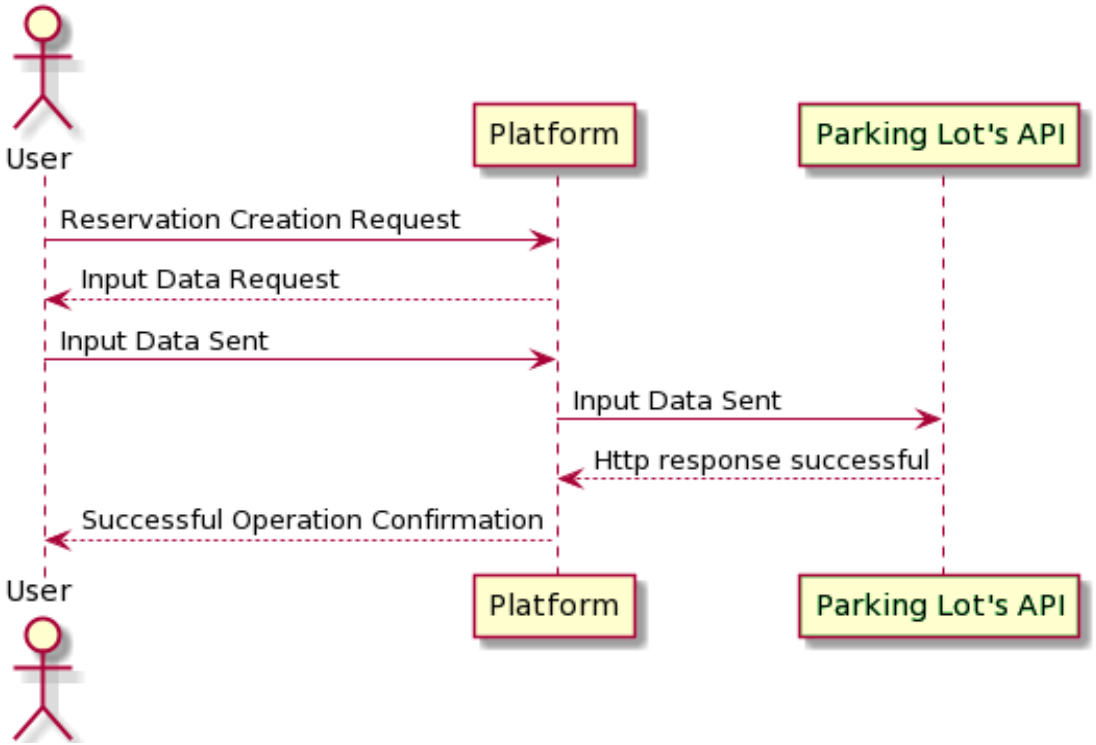
3.3. UC3 - Reservation Management

3.3.1. 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.

3.3.2. SSD



3.3.3. 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.

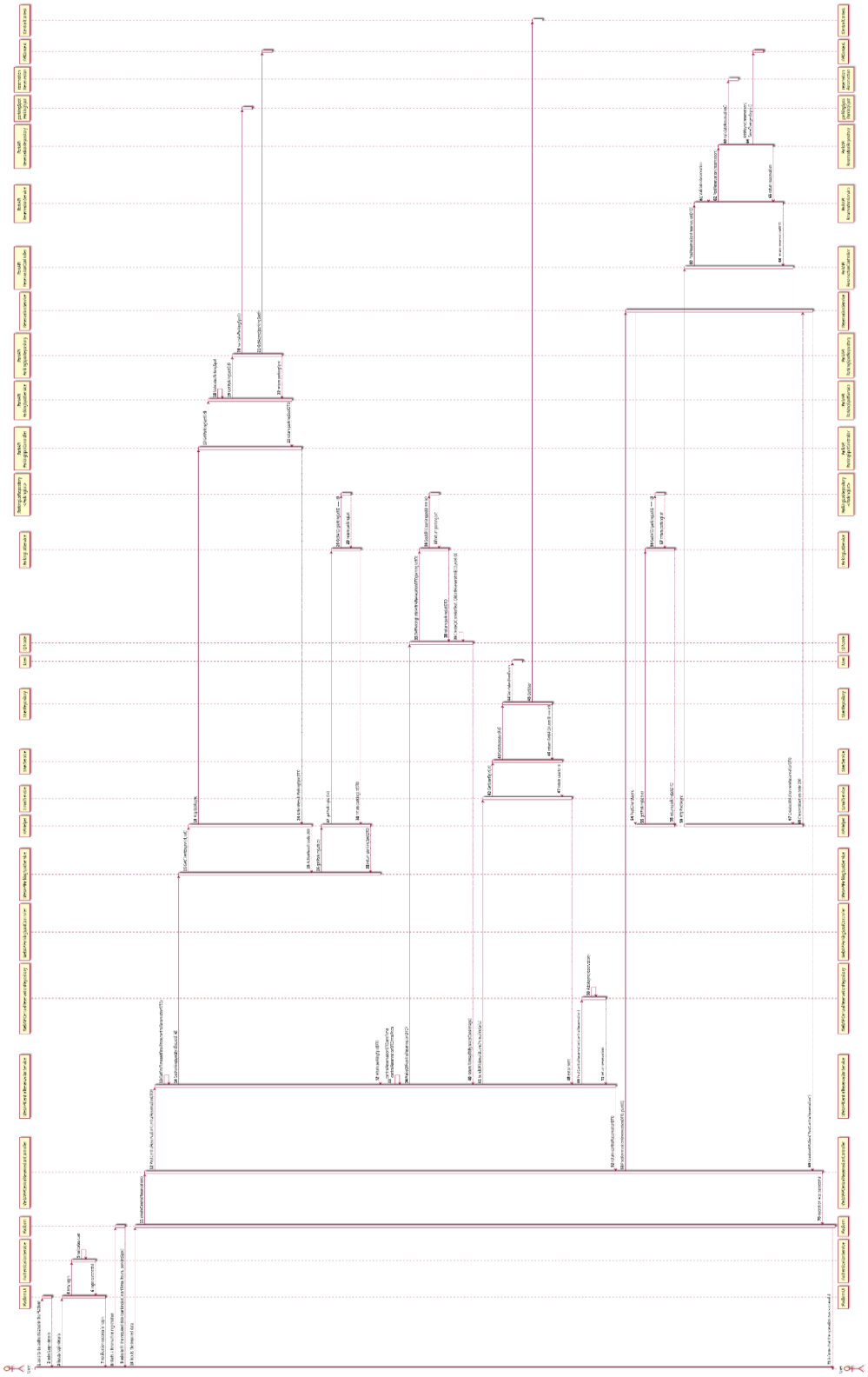
1. API returns an “Unauthorized” (HTTP 401) response or “Forbidden” (HTTP 403).

3.3.4. 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 duration of said reservation.	…displays data?	WebbAppUI	
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	

3.3.5. Sequence Diagram

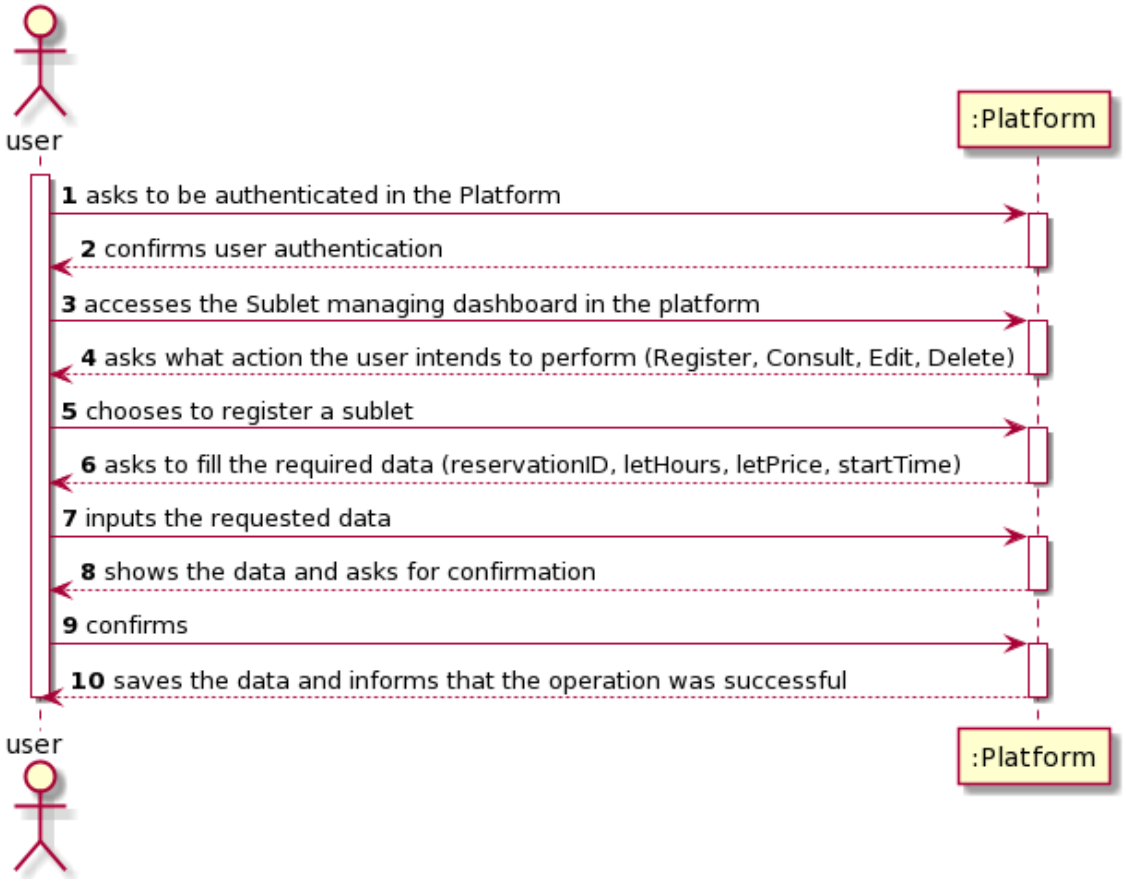


3.4. UC4 - Sublet Management

3.4.1. Brief

The user accesses the platform and authenticates in it. The user opens the SubLet Managing dashboard. 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.

3.4.2. SSD



3.4.3. Fully Dressed

Name: Sublet Management

Primary Actor: User

Stakeholders and interests:

- User: Have the ability to sublet his reservation passing it to another user and have the chance to get credit from this transaction.
- User: Have the chance of reserving an already booked parking spot.
- Platform: Give more rent options to its users.

Preconditions:

- A reservation must exist and must belong to the user.

Postconditions:

- The Sublet list is updated according to the details provided by the user..

Main Success Scenario:

1. The user accesses the Sublet managing dashboard in the platform.
2. The Platform asks what action he intends to perform (Register, Consult, Edit, Delete).
3. The user selects register a sublet.
4. The platform asks to fill the required data (reservationID, letHours, letPrice, startTime)
5. The user inserts the requested data.
6. The platform shows the data to the user, asking him to confirm it.
7. The user confirms.
8. The platform saves the data and informs the user that the operation was successful.

Extensions (alternative scenarios):

***a.**

1. The user cancels the sublet registration process.
2. The use case ends.

5a.

1. The user inserts non valid data.
2. The platform warns him about it.
3. The platform allows the user to change the data.
4. The user does not change the data.
5. The use case ends.

5b.

1. Required data was not inserted
2. The platform informs the user which details are required.
3. The user does not input the required data.
4. The use case ends.

5c. The platform cannot find the Reservation.

1. The platform alerts the user to that fact.
2. The platform allows the user to change the reservation unique identificator.
3. The user does not input an existing reservation.
4. The use case ends.

6a. The platform detects that the sublet details already exist in the platform.

1. The platform alerts the user to that fact.
2. The platform allows the user to change the input details.
3. The user does not change the entered duplicated details.
4. The use case ends.

Special Requirements:

Variations in Technology and Data: n/a

Frequency of Occurrence: Whenever a user wants to sublet his reservation.

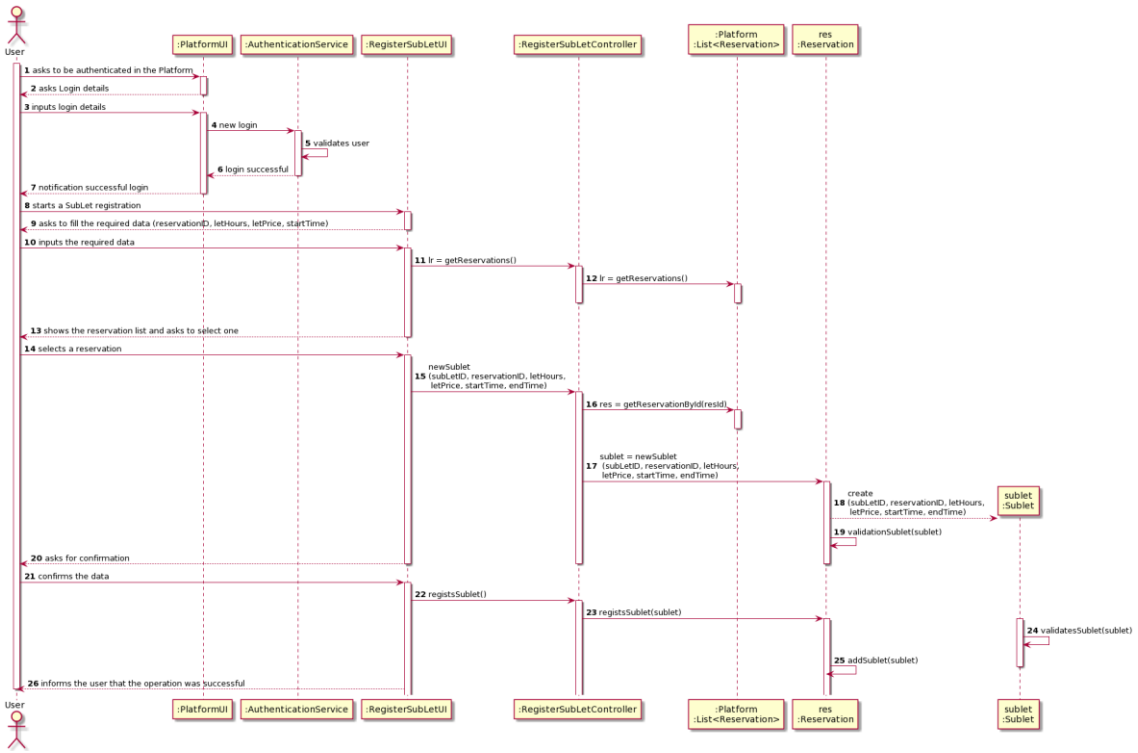
Miscellaneous:

3.4.4. Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1.The user accesses the Sublet managing dashboard in the platform.	...interacts with the user?	CentralAPIUI	Pure Fabrication
2.The Platform asks what action he intends to perform (Register, Consult, Edit, Delete).	…shows User the options?	CentralAPIUI	Pure Fabrication
3.The user selects register a sublet.	...interacts with the user?	CentralAPIUI	Pure Fabrication
	...is responsible for the Register action?	CentralAPIUI RegisterSubletController	Controller
	...creates Sublet instances?	CentralAPI	Creator(rule1)
4.The platform asks to fill the required data (reservationID, letHours, letPrice, startTime)	n/a		
5.The user inputs the requested data.	…saves the inserted details?	CentralAPI Sublet	IE: Instance created in step 1.
6.The platform shows the data to	...validates the user details?	Sublet	IE: Has its own data

the user, asking him to confirm it.	(local validation)		
	...validates the user details? (global validation)	CentralAPI Reservation	IE: Has registered sublets
7.The user confirms.	n/a		
6.The platform saves the data and informs the user that the operation was successful.	...saves the registered sublet?	CentralAPI Reservation	IE: Reservation has Sublet.

3.4.5. Sequence Diagram

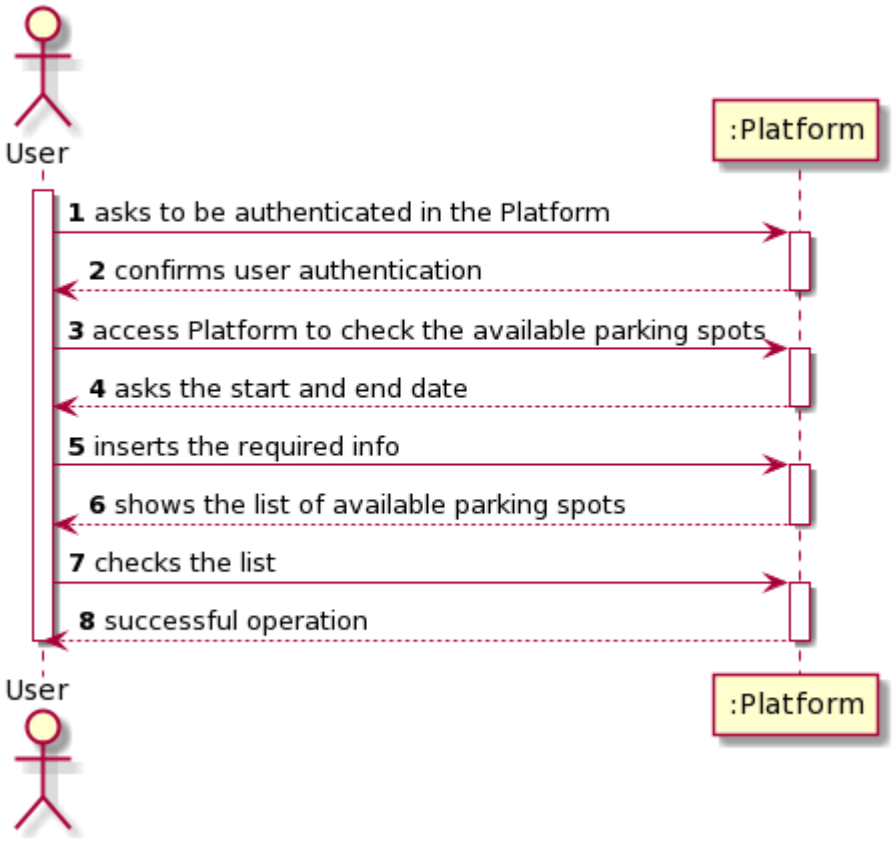


3.5. UC5 - Available Parking Spots Consultation

3.5.1. 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.

3.5.2. SSD



3.5.3. 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:

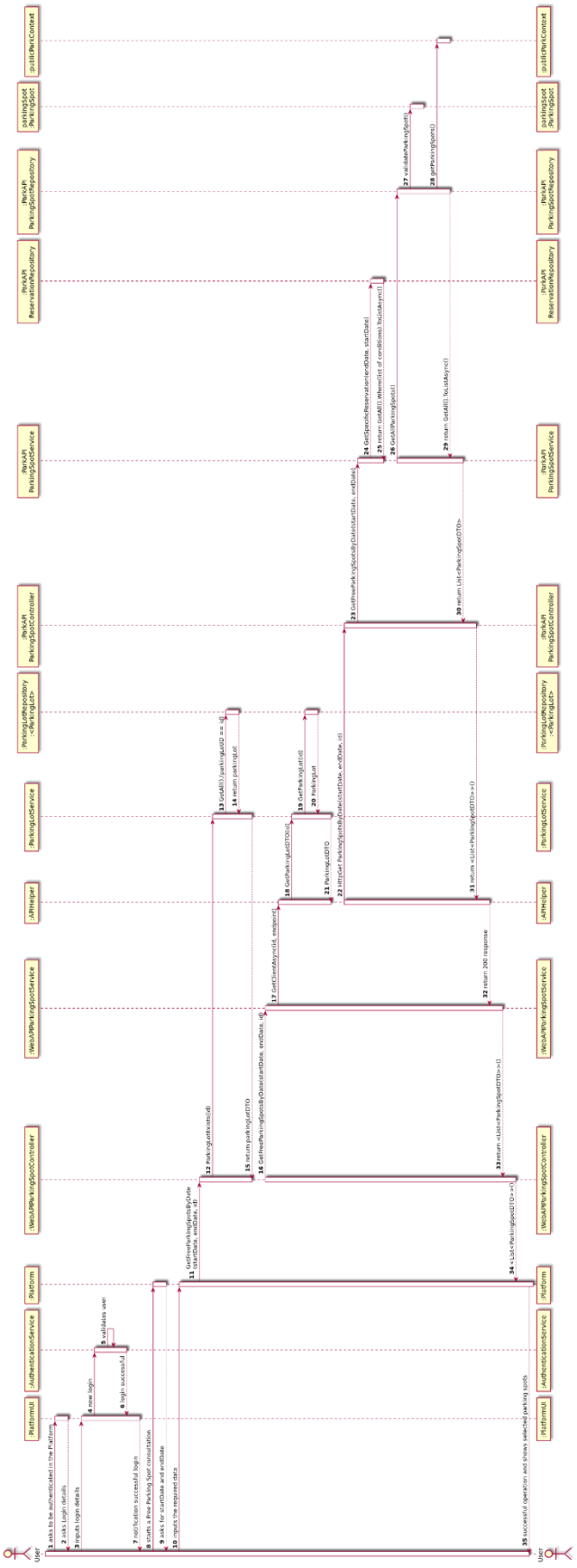
- A non-registered person can check availability?

3.5.4. 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	Pure Fabrication
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	Pure Fabrication
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	…is responsible for displaying the list?	CentralAPI ParkingLot	IE:ParkingLot has ParkingSpot

parking spots to the user.			
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

3.5.5. Sequence Diagram

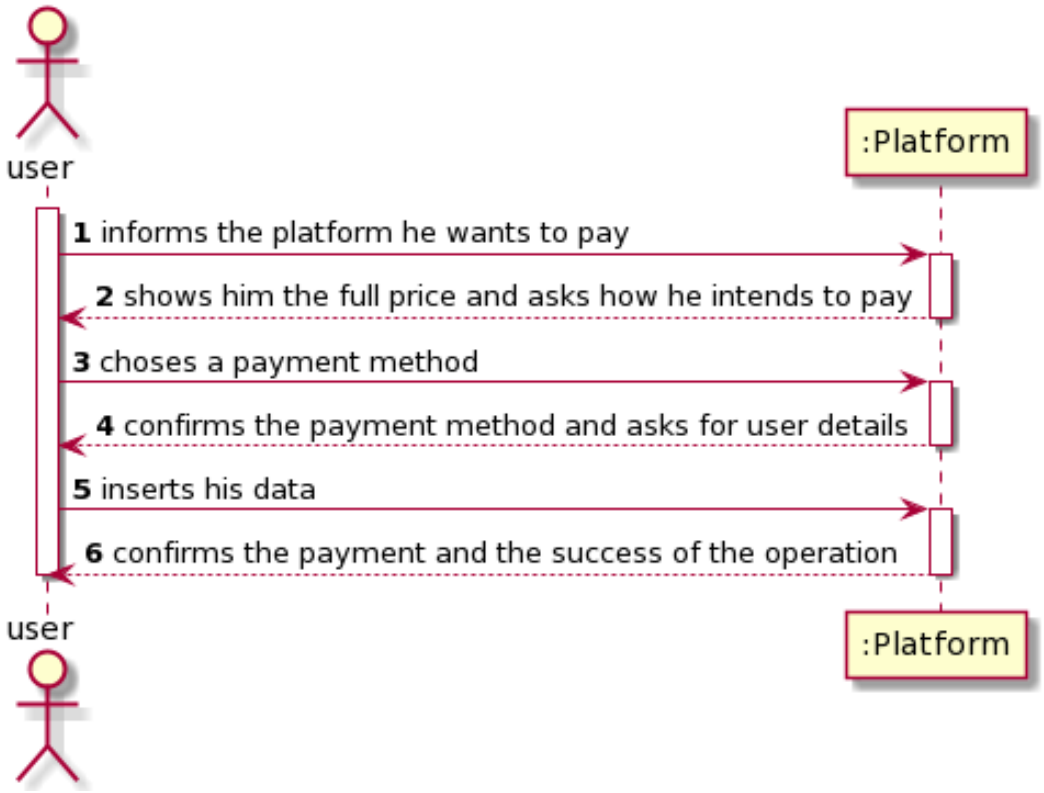


3.6. UC6 - Payment Process

3.6.1. 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.

3.6.2. SSD



3.6.3. Fully Dressed

Name: Payment Processing

Primary Actor: User

Stakeholders and interests:

User:

- Wants purchase and fast service with minimal effort.
- Wants to pay a parking spot reservation on the platform.

Platform:

- Wants to accurately receive the payment in the platform.

Preconditions: There is a previous reservation of a parking spot.

Postconditions:

- The payment approvals are recorded.
- QR-Code is generated.

Main Success Scenario:

1. The user informs the platform he wants to pay.
2. The platform shows him the full price and asks how he intends to pay.
3. The user chooses a payment method.
4. The platform confirms the payment method and asks for user details.
5. The user inserts his payment details.
6. The platform shows the data to the user asking him to confirm it.
9. The user confirms.
10. The platform confirms the payment and the successful reservation process.

Extensions (alternative scenarios):

***a. At any time, systems fail:**

1. To support recovery and correct accounting, ensure all transactions and events can be recovered from any step of the scenario.

***a.**

1. The user cancels the payment.
2. The use case ends.

***b.**

1. The user inserts non-valid data.

2. The platform warns him about it and allows the user to change the data.
3. The user does not change the data.
4. The use case ends.

***c.**

1. The payment is refused.
2. The platform requests the user to inserts new payment details.
3. The user inserts his data.
4. The platform shows the data to the user asking him to confirm it.
5. The user confirms.
6. The platform confirms the payment and the successful reservation process.

Special Requirements:

- Display international currencies and languages depending on the user's nationality.

Variations in Technology and Data: N/A

Frequency of Occurrence: Whenever a user wants to make a payment.

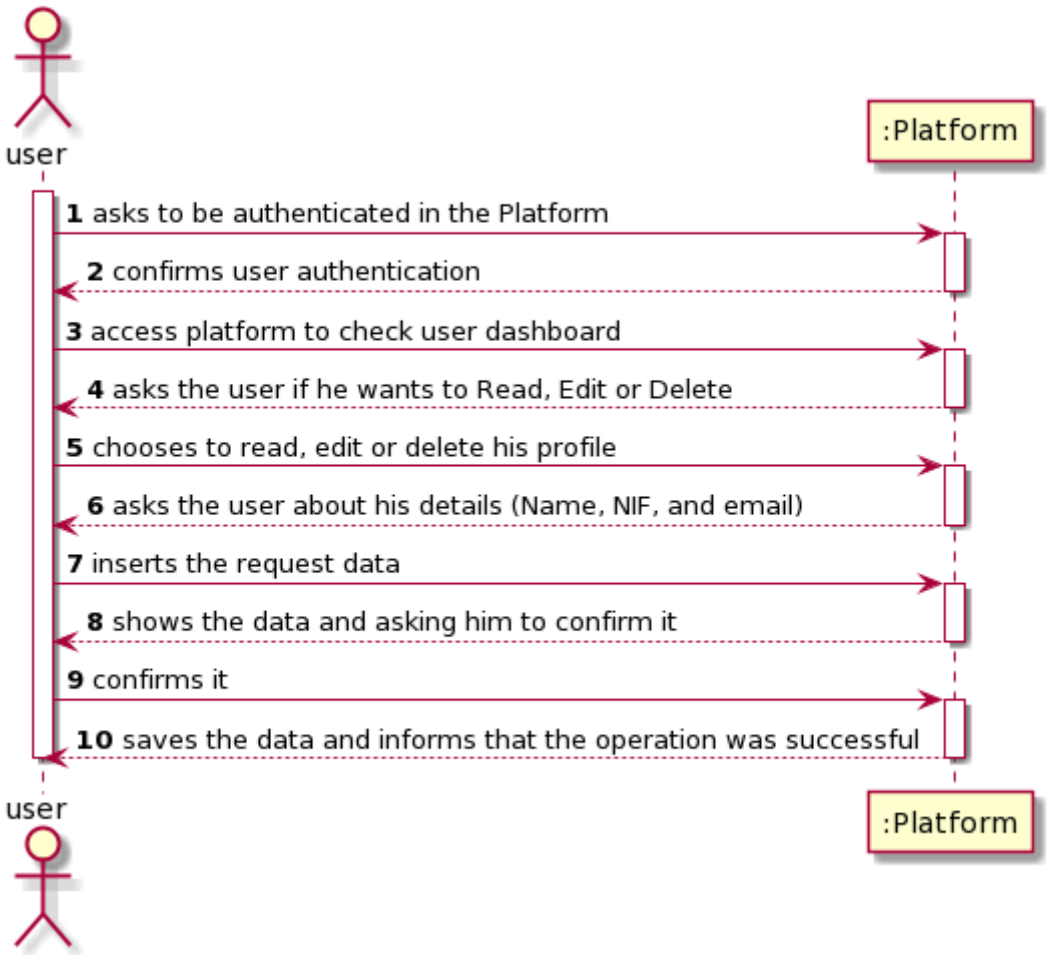
Miscellaneous: N/A

3.7. UC7 - User Management

3.7.1. 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.

3.7.2. SSD



3.7.3. Fully Dressed

Name: User Management

Primary Actor: User

Stakeholders and interests:

- User: Have the ability to read, update or delete his profile on the platform.
- Platform: Have user's information updated in the platform.

Preconditions: n/a

Postconditions:

- The Users dashboard is updated according to the details provided by the user himself.

Main Success Scenario:

1. The user authenticates himself in the platform.
2. The platform confirms his authentication.
3. The user asks the platform to check the user dashboard.
4. The platform shows him the requested info.
5. The user chooses to edit his profile.
6. The platform asks the user about his details (Name, NIF, and email).
7. The user inserts the requested data.
8. The platform shows the data to the user asking him to confirm it.
9. The user confirms.
10. The platform saves the data and informs the new user that the operation was successful.

Extensions (alternative scenarios):

***a.**

1. The user cancels the profile update.
2. The use case ends.

3a. The user only wants to check his dashboard:

1. The platform shows him the requested info.
2. The user checks his profile.

3. The use case ends.

5a. The user wants to delete his profile:

1. The platform shows the data to delete to the user asking him to confirm it.
2. The user confirms.
3. The platform deletes the user profile and informs that the operation was successful.

7a.

1. The user inserts non-valid data.
2. The platform warns him about it and allows the user to change the data.
3. The user does not change the data.
4. The use case ends.

Special Requirements:

Variations in Technology and Data: N/A

Frequency of Occurrence: Whenever a user wants to update his personal details on the platform.

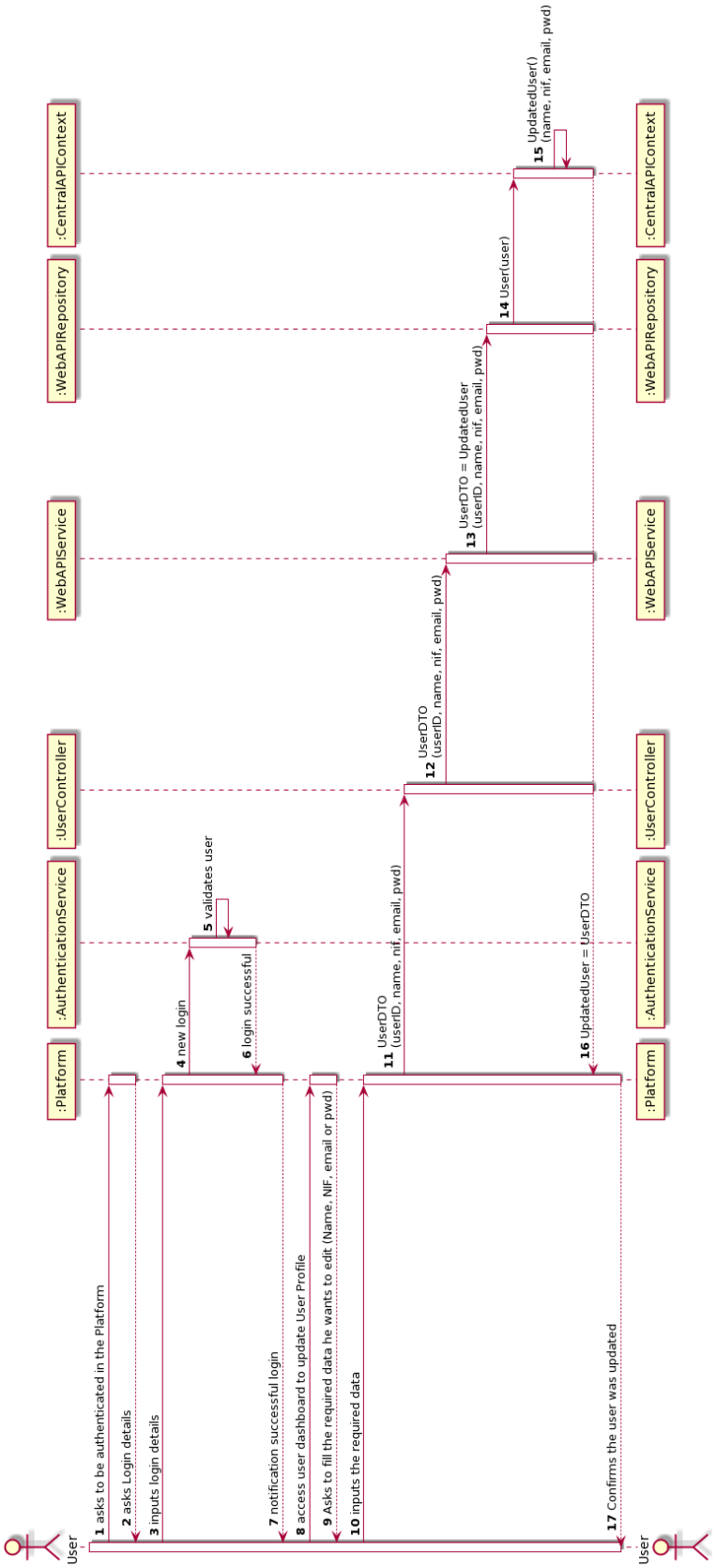
Miscellaneous:

3.7.4. Class Table

Main Flux	Question: Which class(es)...	Answer	Reason
1. The user asks the platform to check the user dashboard.	...interacts with the user?	CentralAPIUI	Pure Fabrication
2.The platform shows him the requested info.	…shows User the options?	CentralAPIUI	Pure Fabrication

3. The user chooses to edit his profile.	...interacts with the user?	CentralAPIUI	Pure Fabrication
	...is responsible for the Edit action?	CentralAPIUI EditUserController	Controller
	...edits User instances?	CentralAPI	Creator(rule1)
4.The platform asks the user about his details (Name, NIF, and email).	n/a		
5.The user inputs the requested data.	…saves the inserted details?	CentralAPI User	IE: Instance created in step 1.
6.The platform shows the data to the user, asking him to confirm it.	...validates the user details? (local validation)	User	IE: Has its own data
	...validates the user details? (global validation)	CentralAPI	IE: Has registered users
7.The user confirms.	n/a		
6.The platform saves the data and informs the user that the operation was successful.	...saves the edited user?	CentralAPI User	IE: CentralAPI has User.

3.7.5. Sequence Diagram


User
:Platform
:AuthenticationService
:UserController
:WebAPIService
:WebAPIRepository
:CentralAPIContext
1 asks to be authenticated in the Platform
2 asks Login details
3 inputs login details
4 new login
5 validates user
6 login successful
7 notification successful login
8 access user dashboard to update User Profile
9 Asks to fill the required data he wants to edit (Name, NIF, email or pwd)
10 inputs the required data
11 UserDTO (userID, name, nif, email, pwd)
12 UserDTO (userID, name, nif, email, pwd)
13 UserDTO = UpdatedUser (userID, name, nif, email, pwd)
14 User(user)
15 UpdatedUser() (name, nif, email, pwd)
16 UpdatedUser = UserDTO
17 Confirms the user was updated
User

4. Product Backlog

4.1. Sprint #2 Backlog

4.1.1. User Stories

(Functional Requirements)

- As a non-user, I want to be able to make an account, so that I can become a user.
- As a non-user, I want to be able to consult the available parking spaces, so that I can get the consulting information.
- As a user, I want to be able to make a parking spot reservation, so that I am able to park there.
- As a user, I want to be able to consult the available parking spots, so that I am able to make a reservation.
- As a user, I want to be able to edit a parking spot reservation, so that I am able to change its location, time or date.
- As a user, I want to be able to cancel a parking spot reservation, in case of a change of plans.
- As an admin, I want to be able to register a parking spot, so that I am able to make it available for further reservations.
- As an admin, I want to be able to edit a parking spot, so that I am able to change and update its properties.
- As an admin, I want to be delete a parking spot, in case of change of its availability.
- As an admin, I want to be able to consult all the parking spots, so that I can manage the parking lot’s readiness.
- As an admin, I want to be able to consult all the parking lot’s reservations, so that I can manage the parking lot.

(Non-Functional Requirements)

- As a user, I need to be authenticated, so that I am able to take advantage of the API's functionalities.

4.2. Sprint #3 Backlog

4.2.1. User Stories

- As a non-user, I want to be able to make an account, so that I can become a user.
- As a user, I want to be able to access a dashboard and change my details, so that I can update my profile and login elements.
- As a user, I want to be able to delete my account, so that I leave the app.
- As a user, I want to be able to consult all the reservation history, so that I can check the active, completed and cancelled reservations.
- As a user, I want to be able to make reservations, so that I am able to park my car.
- As a user, I want to be able to cancel my reservation, so that I am able to go back with my previous choice.
- As a user, I want to be able to create a parking spot sublet, so that I am able to rent it to another user when needed.
- As a user, I want to be able to make payments, so that I can pay for a reservation.
- As a user, I want to have a virtual wallet with credit, so that I can pay through it.
- As a user, I want to access my virtual wallet, so that I can check my credit.
- As a user, I want to be able to make a deposit in my virtual wallet, so that I can update my credit.