**Documentation**

**Inglorious Developers**

**Team Members:**

**Caio Reis**

**João Martins**

**Mariana Gomes**

**Sérgio Pinto**

**Tiago Azevedo**

**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**

**UC1 - Parking Spot Management**

**BRIEF**

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

**SSD**



**FULLY DRESSED**

**Name:** Parking Spot Management

**Primary Actor:** Admin

**Stakeholders and interests:**

* Admin: 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 Admin.

**Main Success Scenario:**

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

**Extensions (alternative scenarios):**

**\*a.**

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

**\*b.**

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

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

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

**5b. The admin just wants to consult the data**

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

**5c. The admin wants to edit a registered parking spot.**

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

**5d. The admin wants to delete a parking spot**

1. The admin selects Delete.
2. The platform shows the list of all the parking spots that were registered.
3. The admin selects a parking spot.
4. The platform shows the selected parking spot info and asks the admin to confirm the action.
5. The admin 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 admin to that fact.
2. The admin does not change the entered Unique Designation.
3. The use case ends.

**Special Requirements:**

**Variations in Technology and Data:** N/A

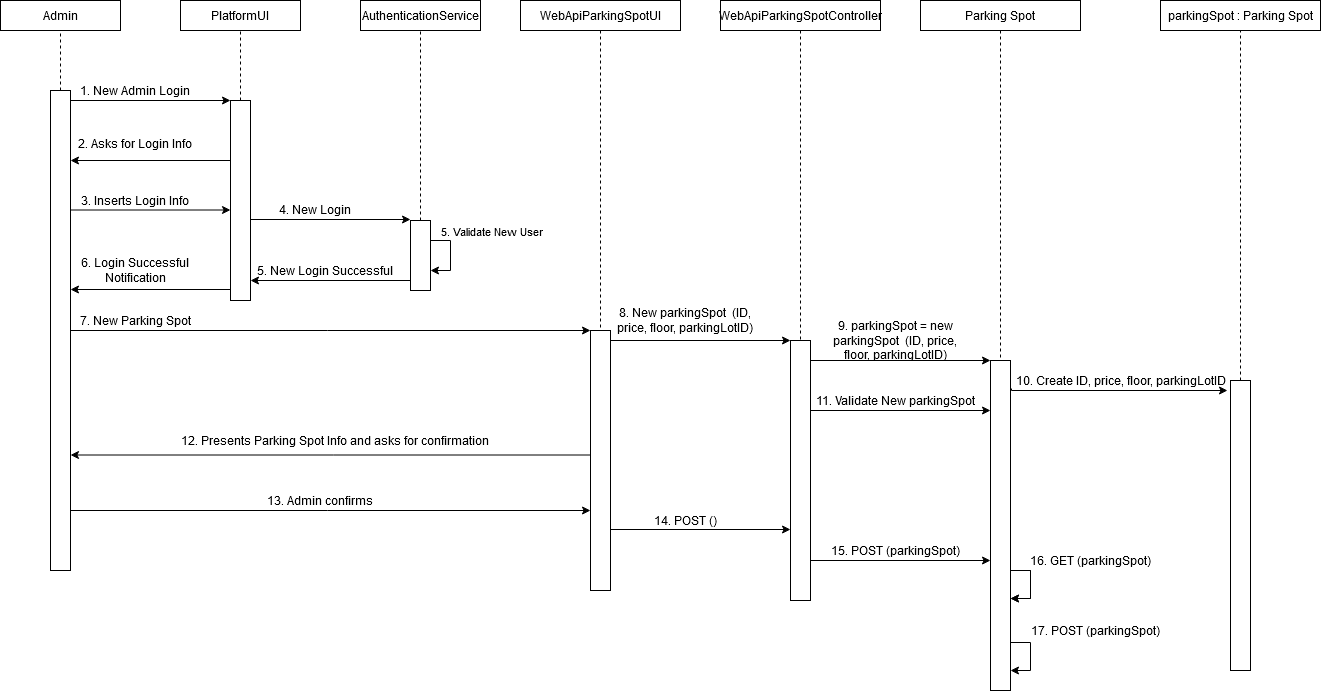
**Frequency of Occurrence:** Whenever a parking place needs to be registered.

**Miscellaneous:**

**Class Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Main Flux** | **Question:**  **Which class(es)...** | **Answer** | **Reason** |
| 1.The admin starts the authentication operation. | ...is responsible for the authentication? | JWToken |  |
| ...interacts with the admin? | WebAppUI |  |
| 2.The Platform allows the admin to be authenticated. | ...shows the admin that the authentication was or not successful? | AuthenticationService (?) / AuthenticationController (?)  WebAppUI |  |
| 3.The admin accesses the platform. | ...interacts with the admin? | WebAppUI |  |
| 4.The platform asks what action the admin intends to perform (Register, Consult, Edit, Delete). | ...is responsible for the Register action? | WebApp |  |
| ...is responsible for the Consult action? | WebApp |  |
| ...is responsible for the Edit action? | WebApp |  |
| ...is responsible for the Delete action? | WebApp |  |
| ...who coordinates the ParkingSpot methods? | ParkingSpotController |  |
| 5.The admin chooses to register a parking space. | ...is responsible for creating ParkingSpot instances? | WebApp |  |
| 6.The platform asks the admin to fill the required data (single designation, floor, price, car park name). | n/a |  |  |
| 7.The admin inputs the requested data. | ...saves the input data? | WebApp  ParkingSpot | IE:ParkingLot has ParkingSpot |
| 8.The platform shows the data to the admin, 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) | WebApp  ParkingSpot | IE: In the DM ParkingSpot is registered in ParkingLot  IE: ParkingLot is registered in WebApp |
| 9.The admin confirms. | n/a |  |  |
| 10.The platform saves the data and informs the admin that the operation was successful. | ...responsible to save the admin input data? | ParkingSpot  WebApp (?)  DataBase |  |

**Sequence Diagram**



**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, 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.

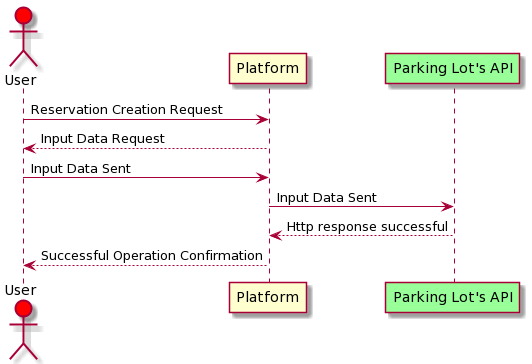
**UC3 - Reservation Management**

**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.

**SSD**



**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).

**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 (ID. priceHours, floor). 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.

**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



**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? | WebAppUI |  |
| 2.The Platform allows the user to be authenticated | ...shows the user that the authentication was or not successful? | AuthenticationService (?) / AuthenticationController (?)  WebAppUI |  |
| 3.The user accesses the platform to check the available parking spots. | ...interacts with the user? | WebAppUI |  |
| 4.The platform asks the start and end date. | n/a |  |  |
| 5.The user inserts the requested data. | ...saves the input data? | WebApp  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? | WebApp  ParkingLot | I |
| 7. The user checks the list. | n/a |  |  |
| 8.The platform informs the user that the operation was successful. |  |  |  |

**Sequence Diagram**  
  


**UC6 - Payment Process**

**UC7 - User Management**