



# Sapien Soft

**#SPRINT3**

**iParkPro - Requisitos, Análise e Design**

## Índice:

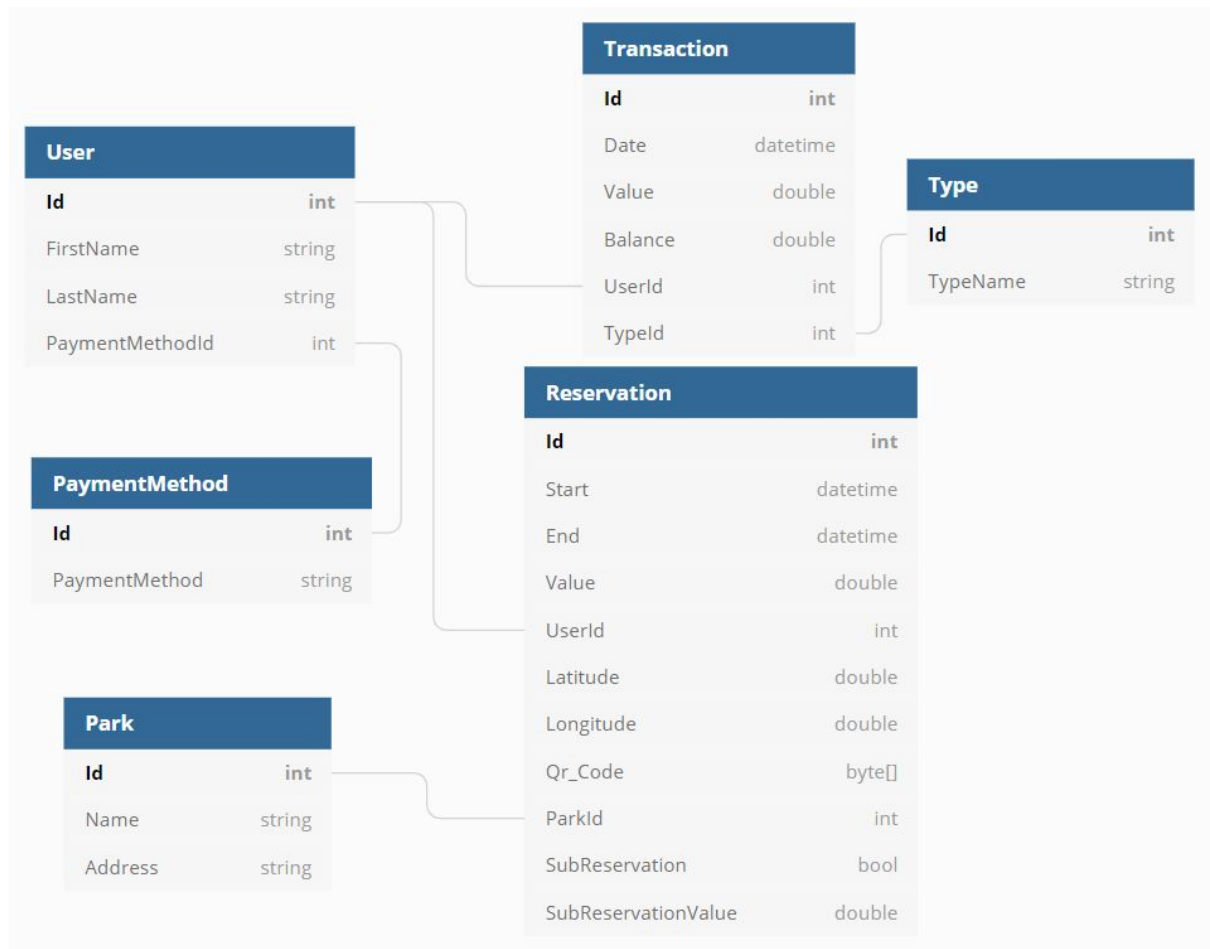
- Estrutura e Associação entre classes (pág 1-3);
- User Stories (pág 4);
- UC1 - Registo de User (pág 5-7);
- UC2 - Utilizador carrega a carteira (pág 8-10);
- UC3 - Utilizador pesquisa por lugares livres (pág 11-13);
- UC4 - Utilizador faz uma reserva (pág 14-16);
- UC5 - Utilizador cancela a sua reserva (pág 17-19);
- UC6 - Utilizador altera a sua password (pág 20-22).
- UC7 - Utilizador sub-aluga uma reserva (pág 23-25).

## Estrutura da API:

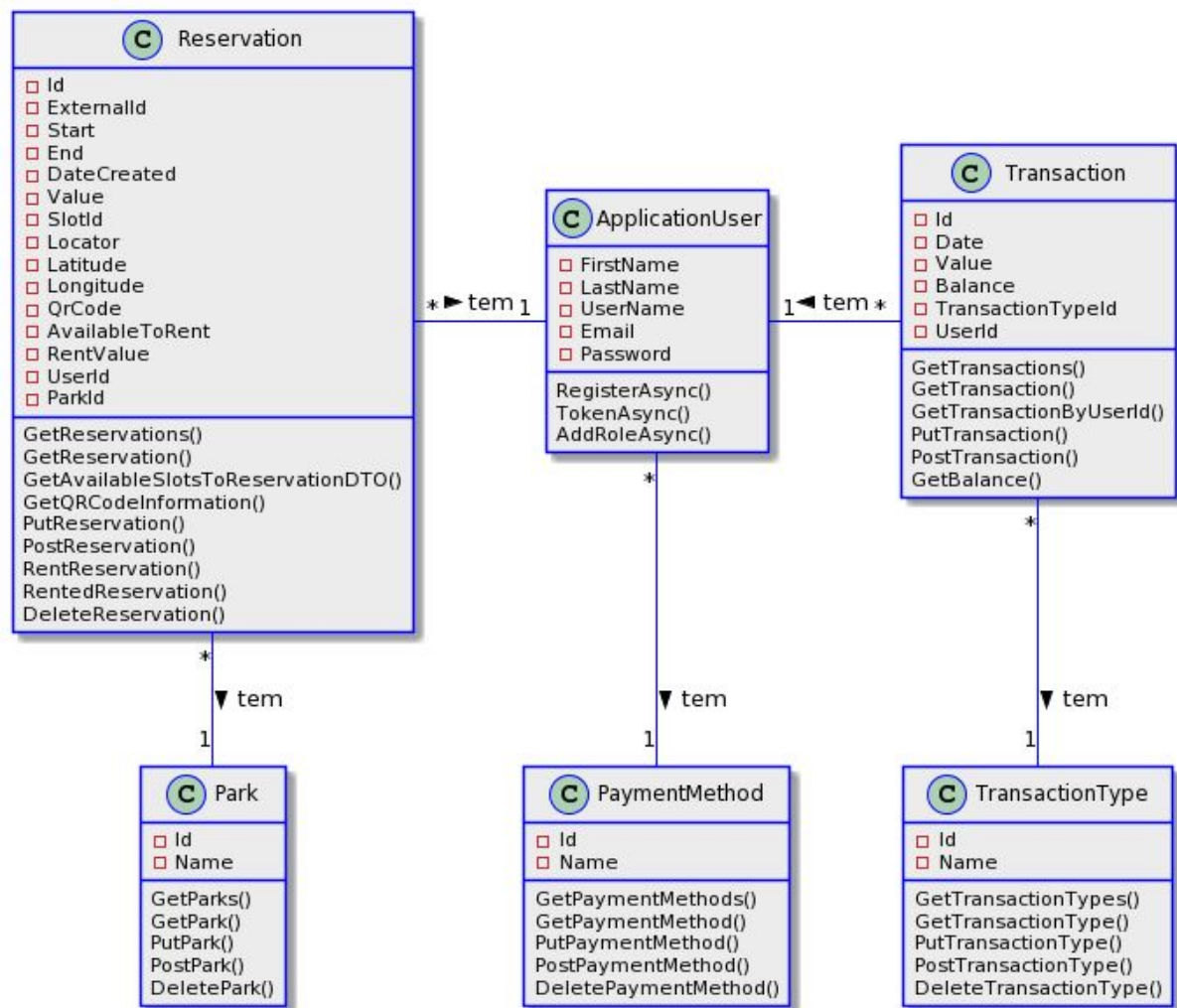
### Classes Candidatas:

User - Id[PK](int), FirstName(string), LastName(string),  
AcceptedPolicy(bool) PaymentMethodId[FK](int),  
LoginMethodId[FK](int);  
PaymentMethod - Id[PK](int), PaymentMethod(string);  
LoginMethod - Id[PK](int), LoginMethod(string);  
Park - Id[PK](int), Name(string), Address(string);  
Transaction - Id[PK](int), Date(DateTime), Value(double),  
Balance(double), UserId[FK](int), TypeID[FK](int);  
Type - Id[PK](int), TypeName(string);  
Reservation - Id[PK](int), Start(DateTime), End(DateTime),  
Value(double), UserId[FK](int), Latitude(double), Longitude(double),  
QR-Code(byte[]), ParkId[FK];  
RentProposal - Id[PK](int), ReservationId[FK], Value(double),  
AvailableStart(DateTime), AvailableEnd(DateTime);  
SubReservation - Id[PK](int), RentProposalId[FK](int),  
Renter(UserId2)[FK](int), StartDate(DateTime), EndDate(DateTime);

## Modelo Dominio:



## Diagrama de classes:



## Associação entre classes

Conceito A	Associação	Conceito B
ApplicationUser	define (CRUD)	<ul style="list-style-type: none"> <li>- Discount</li> <li>- Status</li> <li>- Slots</li> <li>- Reservation</li> <li>- User</li> </ul>
API	possui	<ul style="list-style-type: none"> <li>- Reservation</li> <li>- Discount</li> </ul>
Reservation	possui	<ul style="list-style-type: none"> <li>- ApplicationUser</li> </ul>
Transaction	possui	<ul style="list-style-type: none"> <li>- TransactionType</li> </ul>

## User Stories

User Story	As a <type of user>	I want to <perform some task>	So that I can <achieve some goal>
1	User	Registo na Plataforma.	Utilizar os serviços da Plataforma.
2	User	Obter lugares disponíveis para determinada data/hora de início e fim.	Conseguir encontrar a melhor oferta (lugar ao melhor preço/localização).
3	ParkManager (Administrator)	Proceder à Gestão interna do Parque através de operações CRUD sobre os seus principais elementos.	Atualizar registos da Plataforma.
4	User	Fazer uma reserva.	Garantir um lugar de estacionamento.
5	ParkManager (Administrator), User	Cancelar uma reserva previamente registada.	Evitar custos com um serviço que não vou utilizar.
6	ParkManager (Administrator), Use	User altera password	Garantir segurança da minha conta.

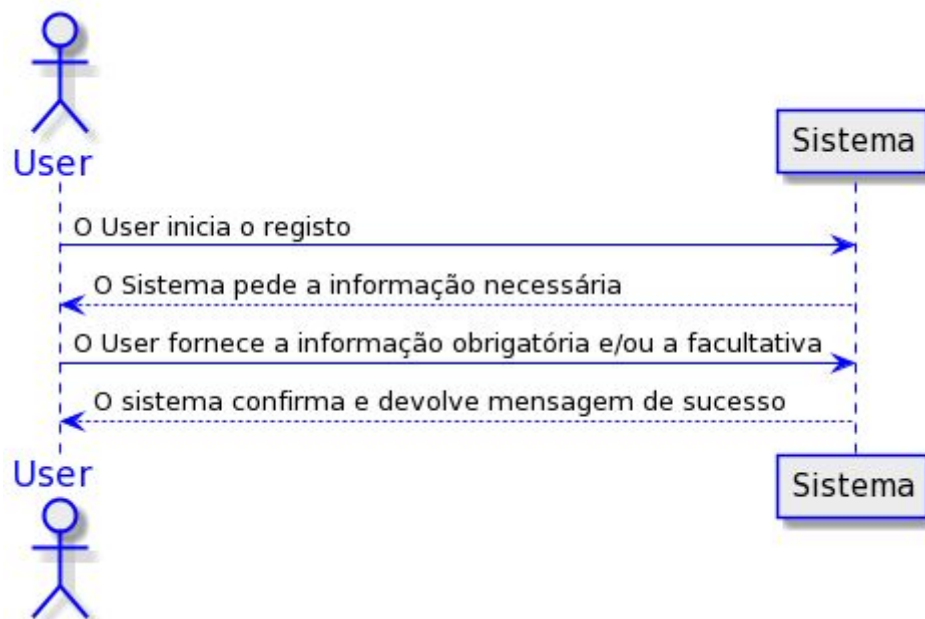
# UC1 - Registo de User

## 1. Requisitos

### 1.1 Breve Descrição

O User não registado inicia o registo de um novo User. O sistema solicita os dados necessários sobre o User (i.e. UserName, Email, PhoneNumber) e pede para ler a informação legal. O User introduz os dados solicitados e confirma que aceita as políticas em vigor. O Sistema valida e apresenta os dados pedindo confirmação. O User confirma. O Sistema regista os dados do novo User e informa o novo utilizador do sucesso da operação.

### 1.2 SSD



### 1.3 Descrição Completa

#### 1.3.1 Ator Principal

User

#### 1.3.2 Partes Interessadas

- **User não-registado** - Ficarà registado e poderá usufruir dos serviços da API dos Parques parceiros.
- **Proprietários da Plataforma** - Mais Users a usar os seus serviços.

#### 1.3.3 Pré-condições

----

#### 1.3.4 Pós-condições

A informação do registo é armazenada no sistema e um e-mail é enviado para o User.

#### 1.3.5 Cenário Principal de Sucesso

1. O User inicia o registo através da Interface;
2. O Sistema recebe um pedido e solicita a informação necessária para o concluir;
3. O User fornece a informação obrigatória e a facultativa;
4. O sistema confirma e devolve mensagem de sucesso;

#### 1.3.6 Extensões (ou fluxos alternativos)

- O potencial novo User solicita o cancelamento do registo.
  - O caso de uso termina.
- O sistema informa que dados mínimos obrigatórios estão em falta.
  - O sistema informa que dados obrigatórios estão em falta.
  - O sistema permite a introdução dos dados em falta.
    - O User não altera os dados. O caso de uso termina.
- O sistema informa que detetou que os dados (ou algum subconjunto dos dados) introduzidos devem ser únicos e que já existem no sistema.
  - O sistema alerta o User para o facto.
  - O sistema permite a sua alteração.
    - O User não altera os dados. O caso de uso termina.

### 1.3.7 Requisitos Especiais

-----

### 1.3.8 Variações de Tecnologias de dados

-----

### 1.3.9 Frequência de ocorrência

Frequência irregular e invulgar.

### 1.3.10 Questões em Aberto

- Sem informação

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso

User	
<b>Id</b>	<b>int</b>
FirstName	string
LastName	string
PaymentMethodId	int

## 3. Design

### 3.1. Racional

Fluxo principal	Questão: Que classe...	Resposta	Justificação
1. O User faz um pedido para um novo registo	... interage com o User?	UserController	Controller
	... coordena o UC?	UserController	Controller
2. A classe UserController passa o pedido à Classe UserService	... valida o modelo?	UserService	Business Logic
3. O RegisterModel cria	... é responsável pela	RegisterModel	Creator

o objeto 'model' e devolve o modelo à classe UserService	persistência da informação?		
4. A classe UserService devolve o modelo à classe UserController.			
6. A classe UserController informa o Client (User) do sucesso da operação.	... É responsável pela troca de informação com o Client (User)?	UserController	IE: A gestão das Roles de Users é responsabilidade do UserService

### 3.2. Sistematização

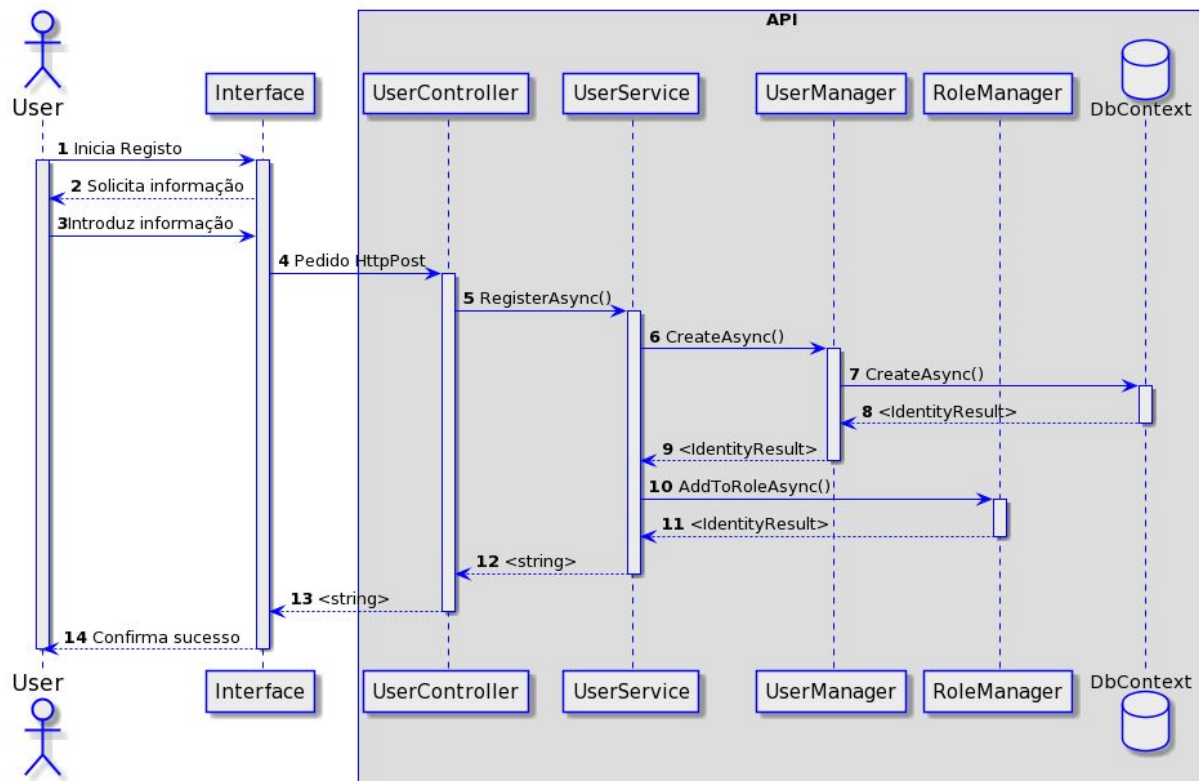
Do racional resulta que as classes conceituais promovidas a classes de software são:

- ApplicationUser

Outras classes de software identificadas:

- UserService
- UserManager
- RoleManager
- RegisterModel

### 3.3. Diagrama de Sequência





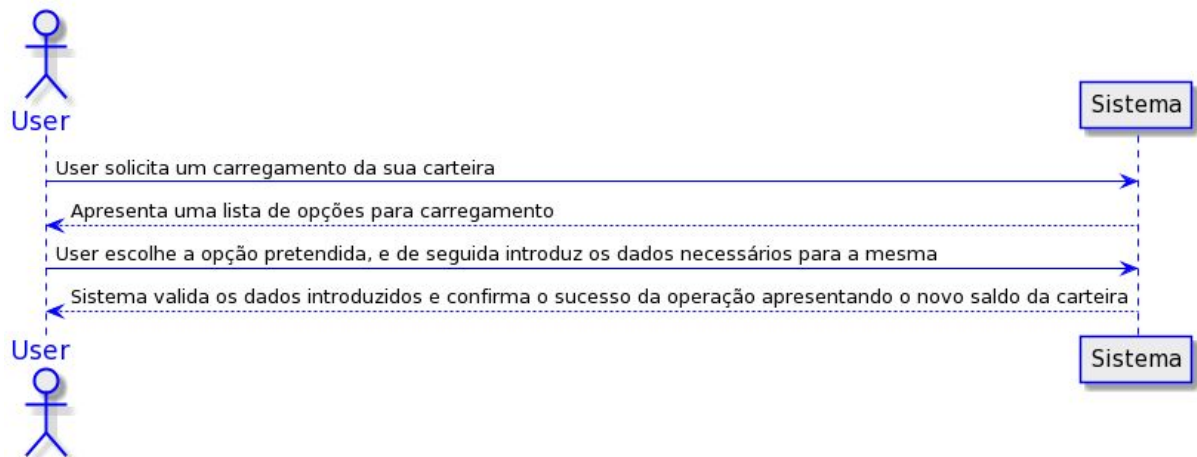
## UC2 - Utilizador carrega a carteira

### 1. Requisitos

#### 1.1 Breve Descrição

O User (utilizador final) pretende adicionar fundos à sua carteira virtual de forma a poder realizar mais transações. Ele solicita o mesmo da aplicação, a mesma encaminha-o para uma escolha de métodos de pagamento disponíveis e o User escolhe o método e conclui o mesmo.

#### 1.2 SSD



#### 1.3 Descrição Completa

##### 1.3.1 Ator Principal

User(Utilizador final)

##### 1.3.2 Partes Interessadas

- **User**- Ter mais saldo na sua carteira para efetuar reservas;
- **Sistema(Plataforma central)** - Deixar o utilizador apto a realizar mais reservas.

##### 1.3.3 Pré-condições

User tem que se encontrar registado na plataforma central.

##### 1.3.4 Pós-condições

O User(Utilizador final) consegue carregar saldo na sua carteira.

##### 1.3.5 Cenário Principal de Sucesso

1. O User(Utilizador final) acede ao sistema e solicita o carregamento da sua carteira
2. O Sistema apresenta os métodos de carregamento
3. O User seleciona o método pretendido
4. O Sistema procede o carregamento utilizando os dados introduzidos pelo User

##### 1.3.6 Extensões (ou fluxos alternativos)

- O User não insere os dados corretos;
  - O sistema informa que os dados introduzidos estão errados;
  - O sistema pergunta ao utilizador se pretende voltar a introduzir os dados ou selecionar um método de carregamento diferente.
- O User não possui fundos para carregar através do método de carregamento pretendido;
  - O Sistema informa o User do ocorrido;
  - O Sistema apresenta a opção de selecionar outro método de carregamento.

### 1.3.7 Requisitos Especiais

-----

### 1.3.8 Variações de Tecnologias de dados

-----

### 1.3.9 Frequência de ocorrência

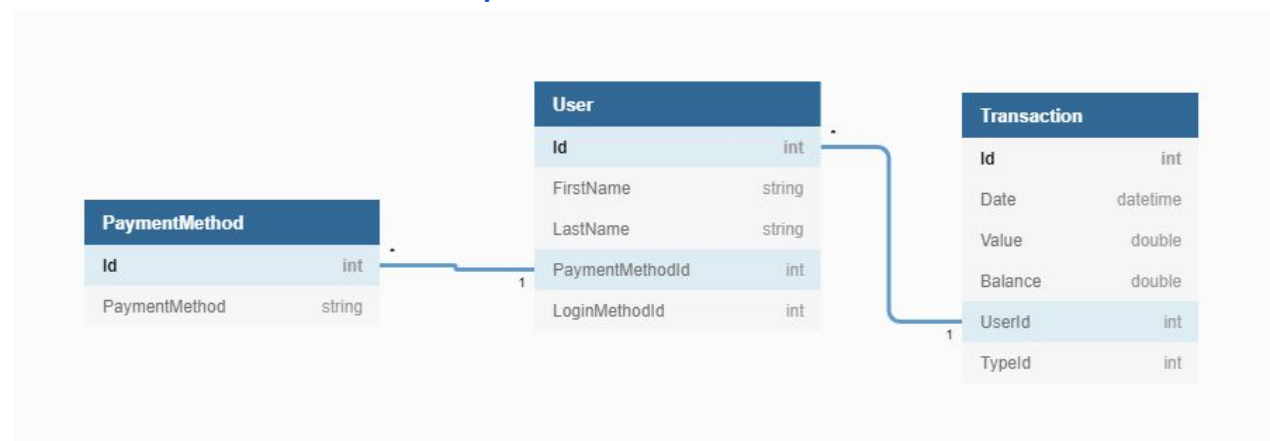
Diária.

### 1.3.10 Questões em Aberto

-----

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso



## 3. Design

### 3.1. Racional

Fluxo principal	Questão: Que classe...	Resposta	Justificação
O User acede ao sistema e solicita o carregamento da sua carteira.	... interage com o User?	UserController	Controller
	... lista os métodos de pagamento?	PaymentMethods	Class
	... realiza o carregamento?	UserServices	Class
	... dá persistência a um histórico de transações	Transactions	Class

### 3.2. Sistematização

Do racional resulta que as classes conceituais promovidas a classes de software são:

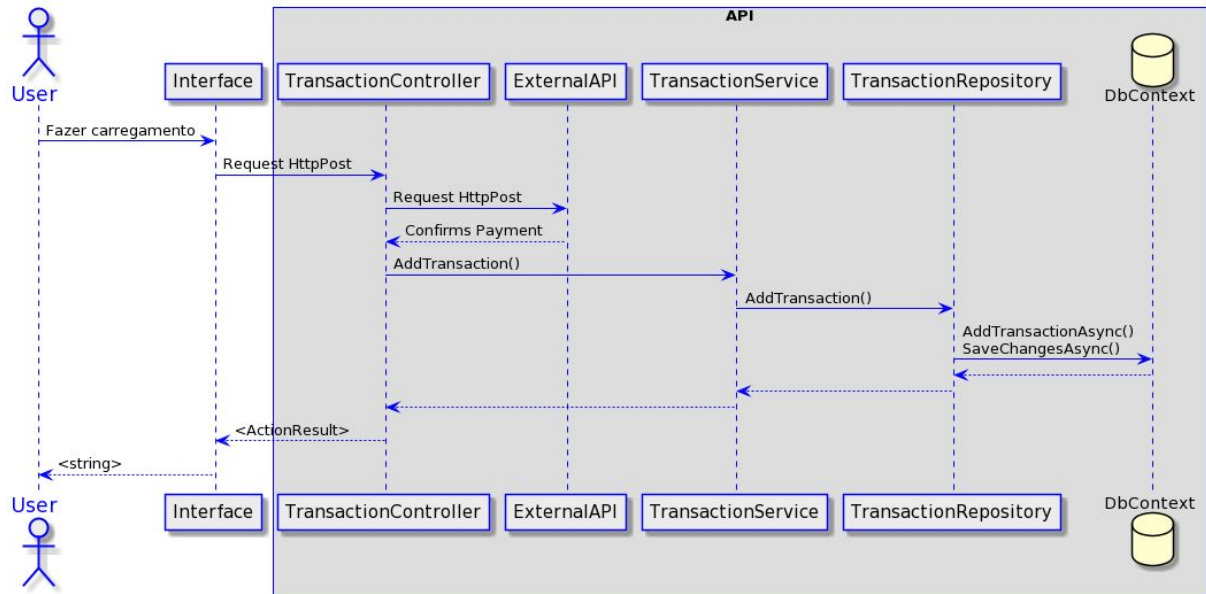
- User
- PaymentMethods
- Transactions
- UserController

- UserServices

Outras classes de software identificadas:

- 

### 3.3. Diagrama de Sequência



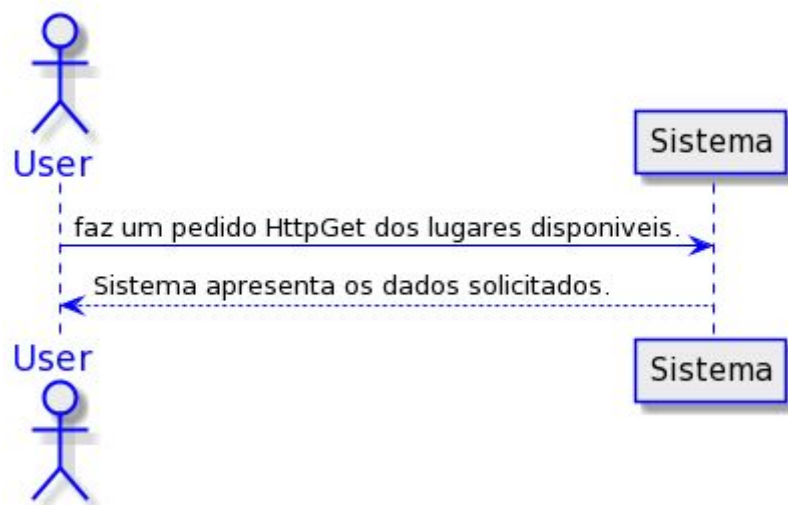
## UC3 - Utilizador pesquisa por lugares livres

### 1. Requisitos

#### 1.1 Breve Descrição

O User Registrado acede à plataforma para visualizar os lugares livres nos diferentes parques, podendo ou não prosseguir para uma reserva. Interpela o sistema nesse sentido, sendo que este recorre a todos os endereços dos diversos API para devolver esta informação ao User, mediante a introdução de alguns parâmetros.

#### 1.2 SSD



#### 1.3 Descrição Completa

##### 1.3.1 Ator Principal

User (Utilizador final)

##### 1.3.2 Partes Interessadas

- **Sapiensoft** - Utilização dos seus serviços;
- **User** - Leitura da informação detalhada de lugares disponíveis de acordo com a sua utilização.

##### 1.3.3 Pré-condições

User tem de estar registado na Plataforma.

##### 1.3.4 Pós-condições

User acede à informação necessária.

##### 1.3.5 Cenário Principal de Sucesso

1. O User acede ao sistema e solicita os lugares disponíveis, dentro de uma determinada data de início e fim.
2. O Sistema apresenta os dados solicitados.

##### 1.3.6 Extensões (ou fluxos alternativos)

- O User não insere data de início e/ou data de fim;
  - O sistema informa que dados mínimos obrigatórios estão em falta;
  - O sistema permite a reinserção dos dados;
- Durante o período de disponibilização dos dados, um dos lugares apresentados ficou reservado.

- Ao selecionar o lugar e avançando para usa reserva, o User deve ser alertado pelo Sistema que o lugar em questão foi reservado aquando a reserva;
- O caso de uso termina.

### 1.3.7 Requisitos Especiais

----

### 1.3.8 Variações de Tecnologias de dados

----

### 1.3.9 Frequência de ocorrência

Horária

### 1.3.10 Questões em Aberto

- O user vai querer consultar a disponibilidade de um parque em específico?

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso



## 3. Design

### 3.1. Racional

Fluxo principal	Questão: Que classe...	Resposta	Justificação
O User acede ao sistema e solicita os lugares vagos	... interage com o User?	ReservationsInterface	Interface
	... coordena o UC?	ReservationsController	Controller
O ReservationsController faz um pedido HttpGet para obter as slots desse parque assim como as reservas	... obtém esta informação?	ReservationService	Service
	... responsável por fornecer os dados?	ReservationRepository	Repository

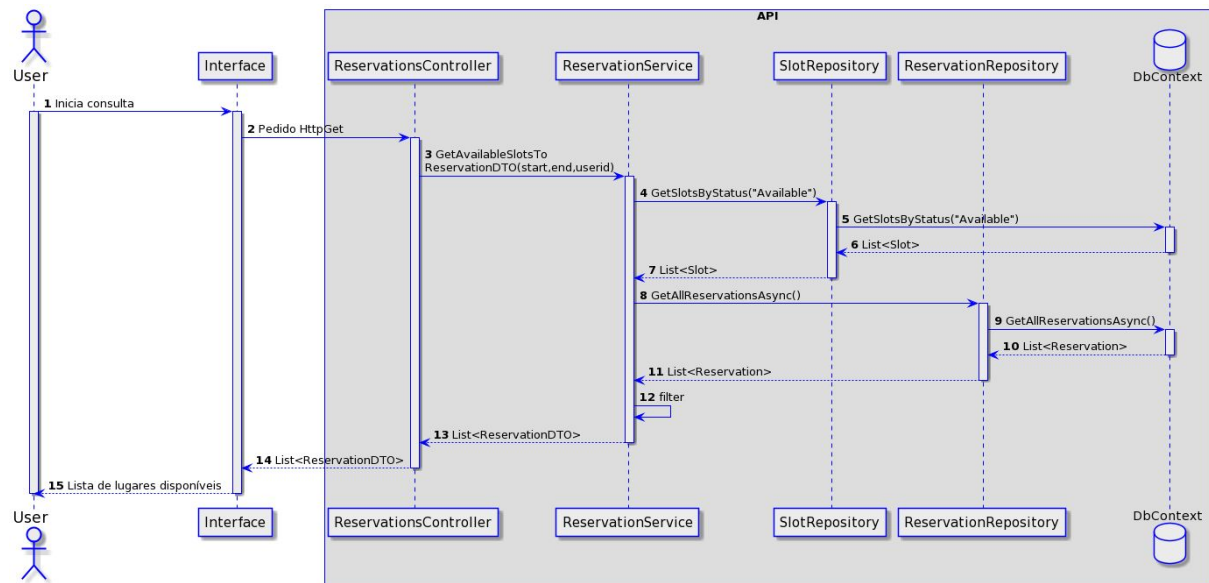
ativas, compilando uma lista de slots livres.	... responsável por armazenar os dados?	DbContext	Base de Dados
---	---	-----------	---------------

### 3.2. Sistematização

Do racional resulta que as classes conceituais promovidas a classes de software são:

- ReservationController
- ReservationsInterface
- ReservationService
- ReservationRepository
- DbContext

### 3.3. Diagrama de Sequência



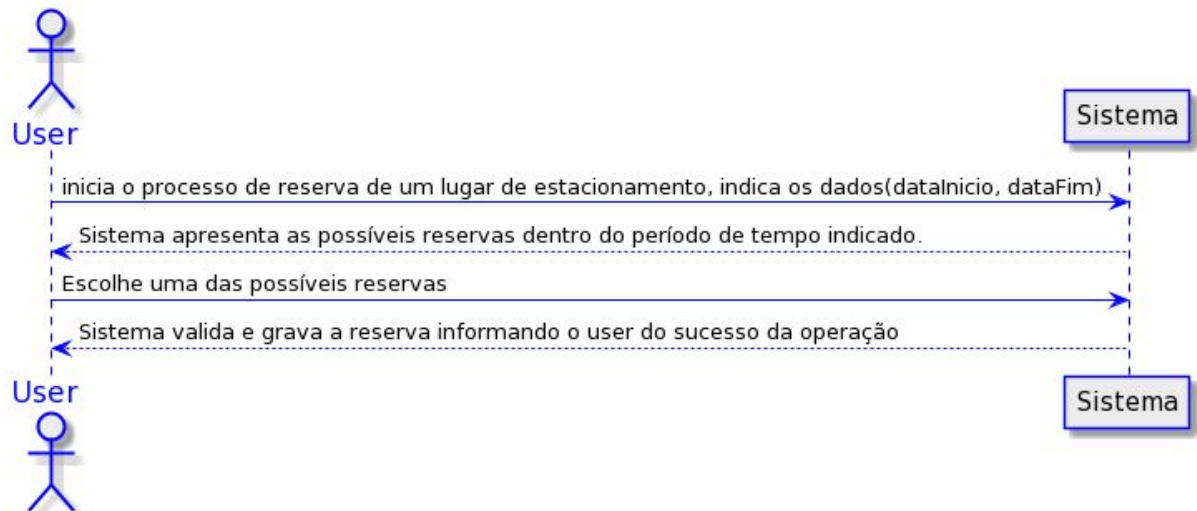
## UC4 - User faz uma Reserva

### 1. Requisitos

#### 1.1 Breve Descrição

O User (User final), depois de fazer um pedido de todos os lugares livres, dentro de um período de tempo indicado, seleciona uma das reservas e inicia um pedido Post(reserva). O Sistema valida e grava a reserva escolhida. Informa o utilizador do sucesso da operação.

#### 1.2 SSD



#### 1.3 Descrição Completa

##### 1.3.1 Ator Principal

User(User Final)

##### 1.3.2 Partes Interessadas

- User (user final): Possibilidade de fazer reservas.
- Api Central: Persistência das reservas e atualização de disponibilidade.
- Plataforma Central: Utilização dos seus serviços.

##### 1.3.3 Pré-condições

- User tem de estar registado.

##### 1.3.4 Pós-condições

- A reserva fica gravada tanto na Api Central como no sistema de origem.

##### 1.3.5 Cenário Principal de Sucesso

O User realiza a reserva do seu lugar, fica registada e poderá usufruir do lugar de estacionamento.

##### 1.3.6 Extensões (ou fluxos alternativos)

- User desiste da reserva a meio de realizar a mesma
- O caso de uso termina.

##### 1.3.7 Requisitos Especiais

-----

##### 1.3.8 Variações de Tecnologias de dados

-----

##### 1.3.9 Frequência de ocorrência

Constante

##### 1.3.10 Questões em Aberto

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso

Reservation	
Id	int
Start	datetime
End	datetime
Value	double
UserId	int
Latitude	double
Longitude	double
Qr_Code	byte[]
ParkId	int
SubReservation	bool
SubReservationValue	double

## 3. Design

### 3.1. Racional

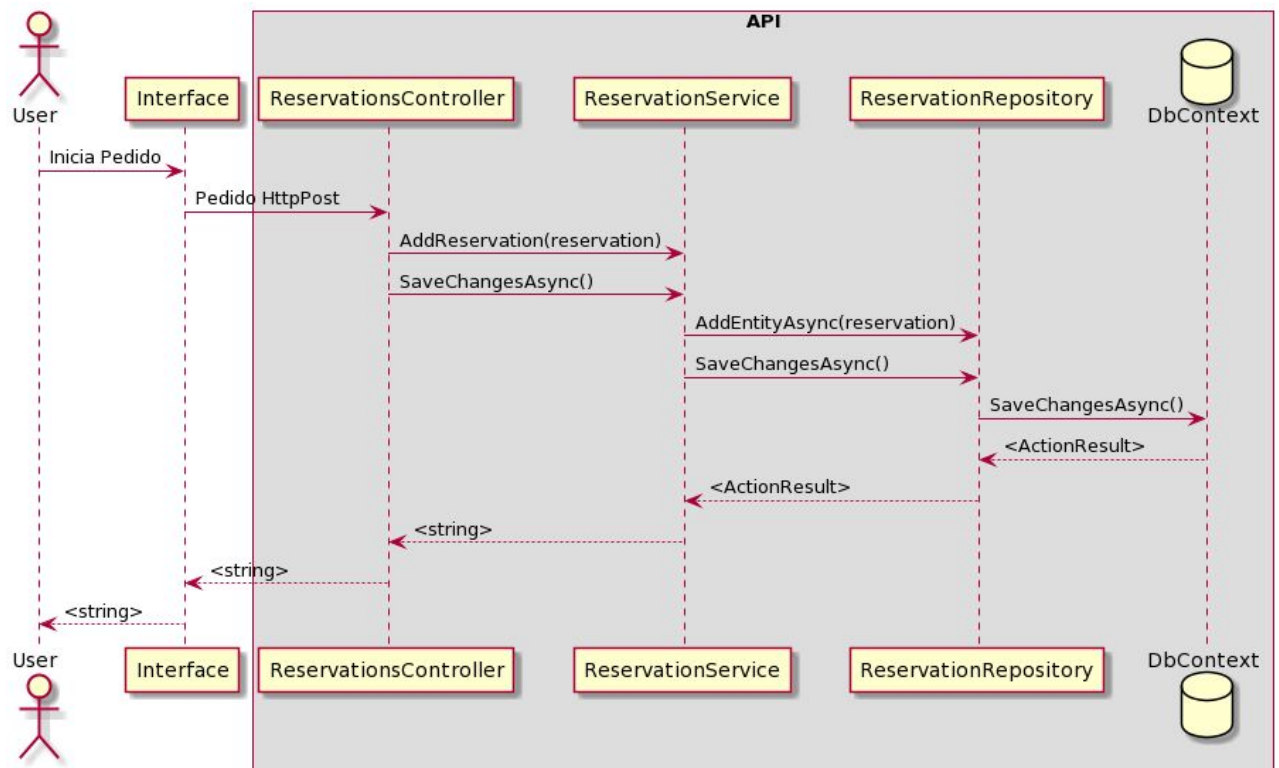
Fluxo principal	Questão: Que classe...	Resposta	Justificação
O User inicia o processo de reserva	... apresenta as reservas disponíveis	ReservationsController	Controller
O User escolhe o lugar	... cria a nova reserva	ReservationsController	Controller

### 3.2. Sistematização

-



### 3.3. Diagrama de Sequência



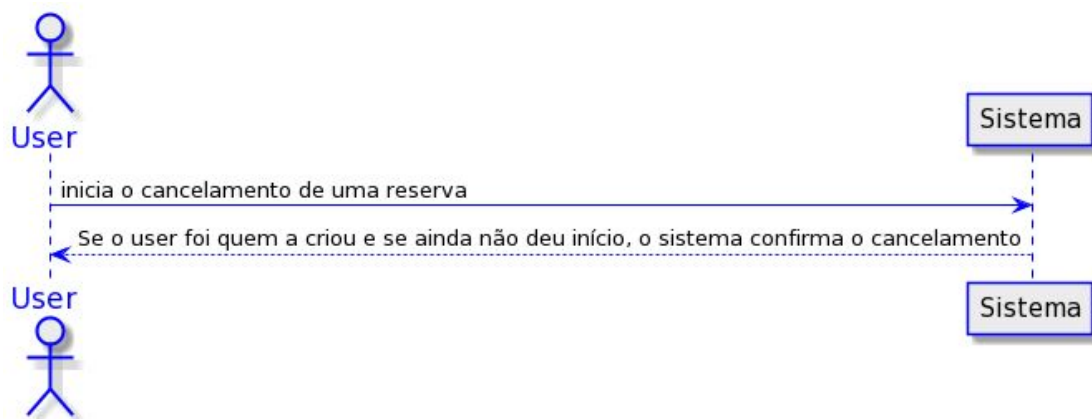
## UC5 - User cancela Reserva

### 1. Requisitos

#### 1.1 Breve Descrição

O User (User final), inicia um pedido HttpDelete para cancelar uma reserva efetuada. O Sistema verifica se o user é o criador da reserva que pretende cancelar e se está dentro do período permitido para cancelamento estipulado. Após isso, efetua o cancelamento e apaga a reserva efetuada.

#### 1.2 SSD



#### 1.3 Descrição Completa

##### 1.3.1 Ator Principal

User(User Final)

##### 1.3.2 Partes Interessadas

- User (user final): Possibilidade de cancelar reservas.
- Api Central: Atualização de disponibilidade.
- Plataforma Central: Utilização dos seus serviços.

##### 1.3.3 Pré-condições

---

##### 1.3.4 Pós-condições

A Reserva tem de ser removida do sistema de origem de forma a manter atualizada a disponibilidade dos lugares de estacionamento.

##### 1.3.5 Cenário Principal de Sucesso

1. O User(User Final) acede ao sistema e inicia um processo de cancelamento de reserva

2. O Sistema confirma se o User está autorizado (Se foi o criador da reserva e se está dentro do período permitido para cancelamento estipulado ) e confirma o sucesso do processo..

##### 1.3.6 Extensões (ou fluxos alternativos)

- O User(Plataforma Central) não é o criador e/ou o período permitido para o cancelamento foi ultrapassado.
  - (Não é o criador:) O sistema informa que o User não tem autorização para fazer aquele pedido.
  - O caso de uso termina.

- (O período de cancelamento foi ultrapassado:) O sistema informa o User do razão da impossibilidade de proceder ao cancelamento da reserva.
- O caso de uso termina.
- Durante o tempo de escolha, um dos lugares apresentados ficou reservado naquele horário escolhido por outro User.
  - Ao selecionar a possível reserva, o Sistema terá de informar a Plataforma Central que a reserva escolhida foi reservada durante o processo.
  - O caso de uso termina.

### 1.3.7 Requisitos Especiais

- Devolução Monetária

### 1.3.8 Variações de Tecnologias de dados

- Método de pagamento; Devolução monetária.

### 1.3.9 Frequência de ocorrência

Possivelmente Diária.

### 1.3.10 Questões em Aberto

-----

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso

Reservation	
<b>Id</b>	<b>int</b>
Start	datetime
End	datetime
Value	double
UserId	int
Latitude	double
Longitude	double
Qr_Code	byte[]
ParkId	int
SubReservation	bool
SubReservationValue	double

## 3. Design

### 3.1. Racional

Fluxo principal	Questão: Que classe...	Resposta	Justificação
O User(Plataforma Externa) acede ao sistema e inicia um processo de	... interage com o User?	ReservationsController	Controller
	... coordena o UC?	ReservationsController	Controller(Delete)

cancelamento de reserva			
-------------------------	--	--	--

### 3.2. Sistematização

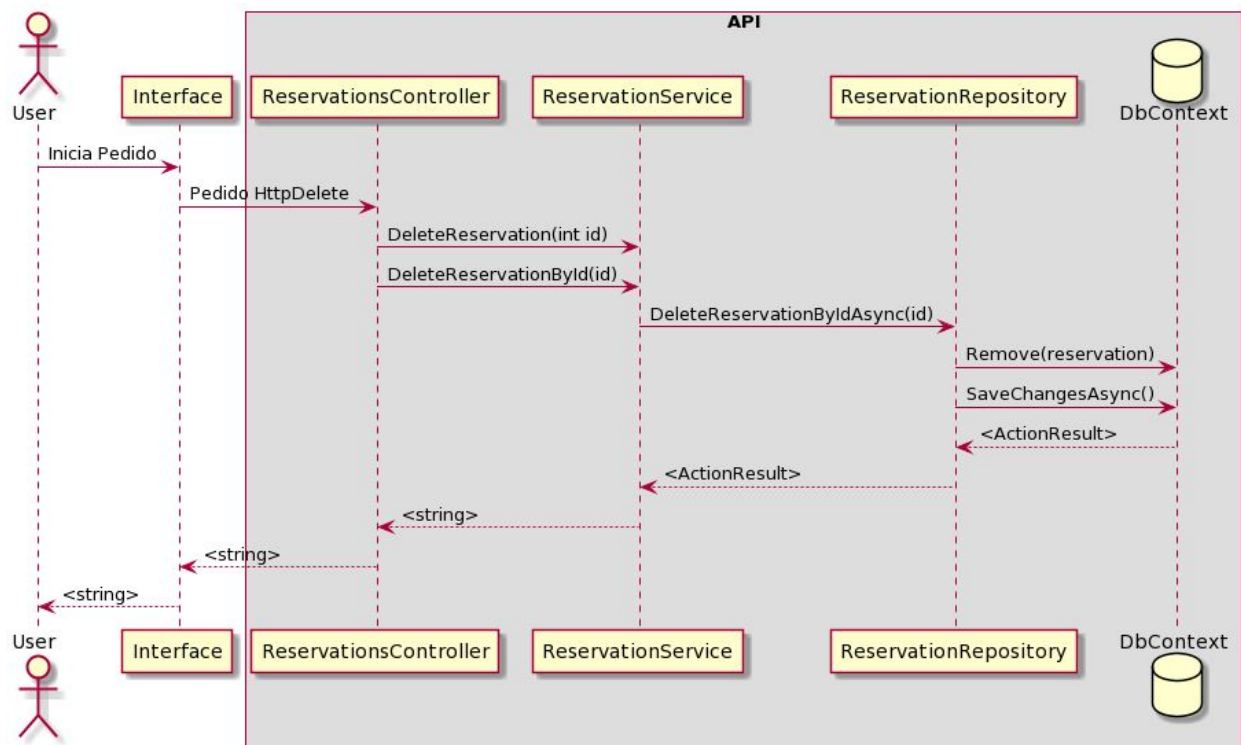
Do racional resulta que as classes conceptuais promovidas a classes de software são:

- Reservation

Outras classes de software identificadas:

- ReservationsController
- ReservationService
- IReservationService - classe interface.
- ReservationRepository
- IReservationRepository - classe interface.

### 3.3. Diagrama de Sequência



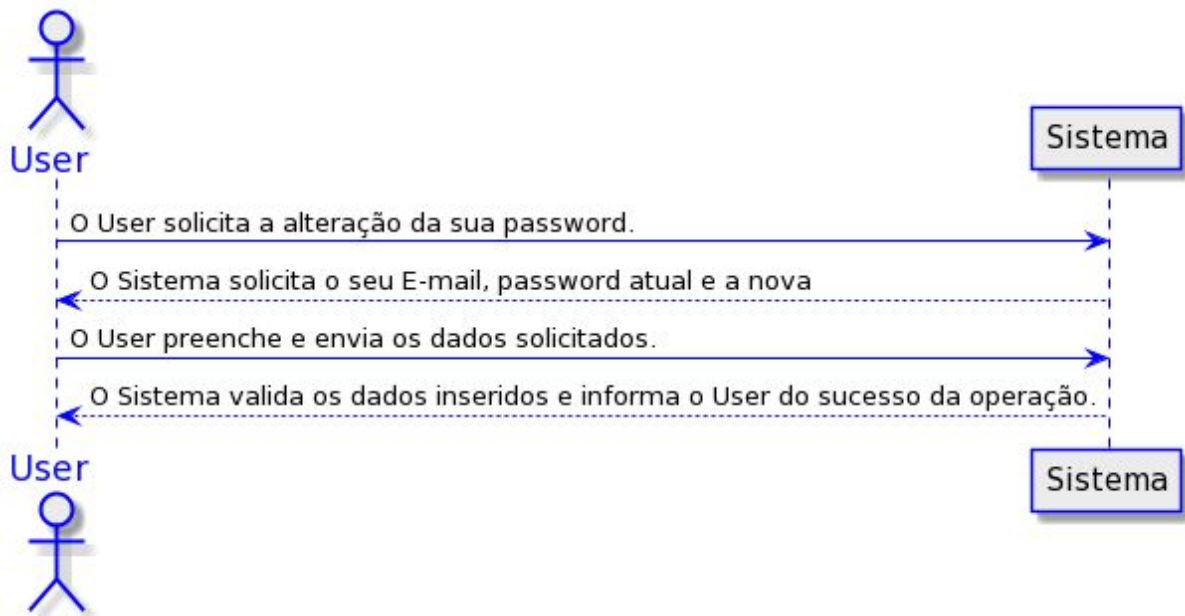
## UC6 - User altera a sua password

### 1. Requisitos

#### 1.1 Breve Descrição

O User inicia o processo para alterar a sua password. O sistema solicita o E-mail do User assim como a password atual e a nova. O Sistema valida os dados e depois informa o User do sucesso da operação.

#### 1.2 SSD



### 1.3 Descrição Completa

#### 1.3.1 Ator Principal

User

#### 1.3.2 Partes Interessadas

- User: Alterar a sua password.

#### 1.3.3 Pré-condições

- O user já se encontra registrado

#### 1.3.4 Pós-condições

-----

#### 1.3.5 Cenário Principal de Sucesso

O User solicita uma alteração de password, fornecendo os dados necessários. Conclui com a password alterada.

#### 1.3.6 Extensões (ou fluxos alternativos)

- User insere um E-mail não registado
  - O Sistema informa o User, e reinicia o processo.
- User não insere a password atual correctamente
  - O Sistema informa o User, e reinicia o processo.
- Password nova não preenche os pré-requisitos
  - O Sistema informa o User, e reinicia o processo.

#### 1.3.7 Requisitos Especiais

-----

### 1.3.8 Variações de Tecnologias de dados

-----

### 1.3.9 Frequência de ocorrência

- Ocasional

### 1.3.10 Questões em Aberto

-----

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso

User	
<b>Id</b>	<b>int</b>
FirstName	string
LastName	string
PaymentMethodId	int

## 3. Design

### 3.1. Racional

Fluxo principal	Questão: Que classe...	Resposta	Justificação

### 3.2. Sistematização

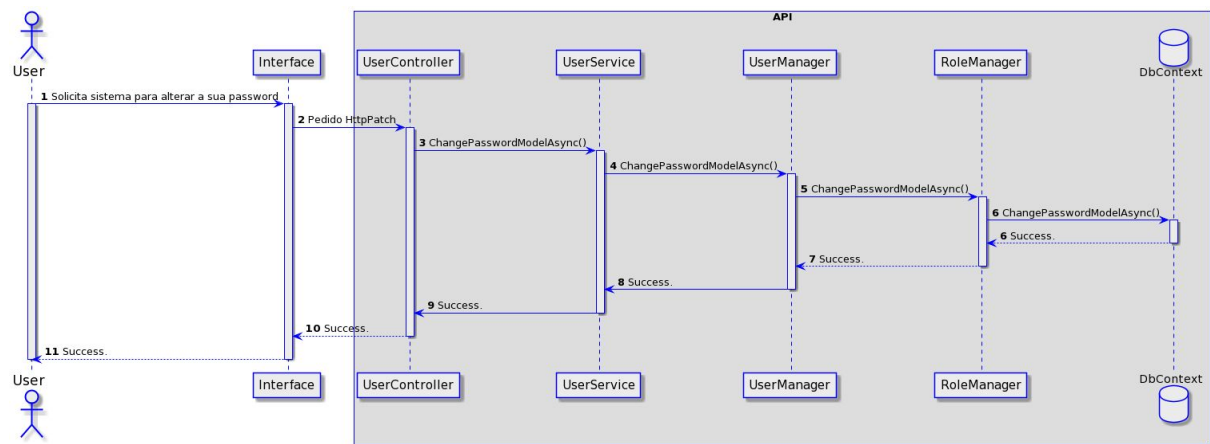
Do racional resulta que as classes conceituais promovidas a classes de software são:

- User

Outras classes de software identificadas:

- UserController

### 3.3. Diagrama de Sequência



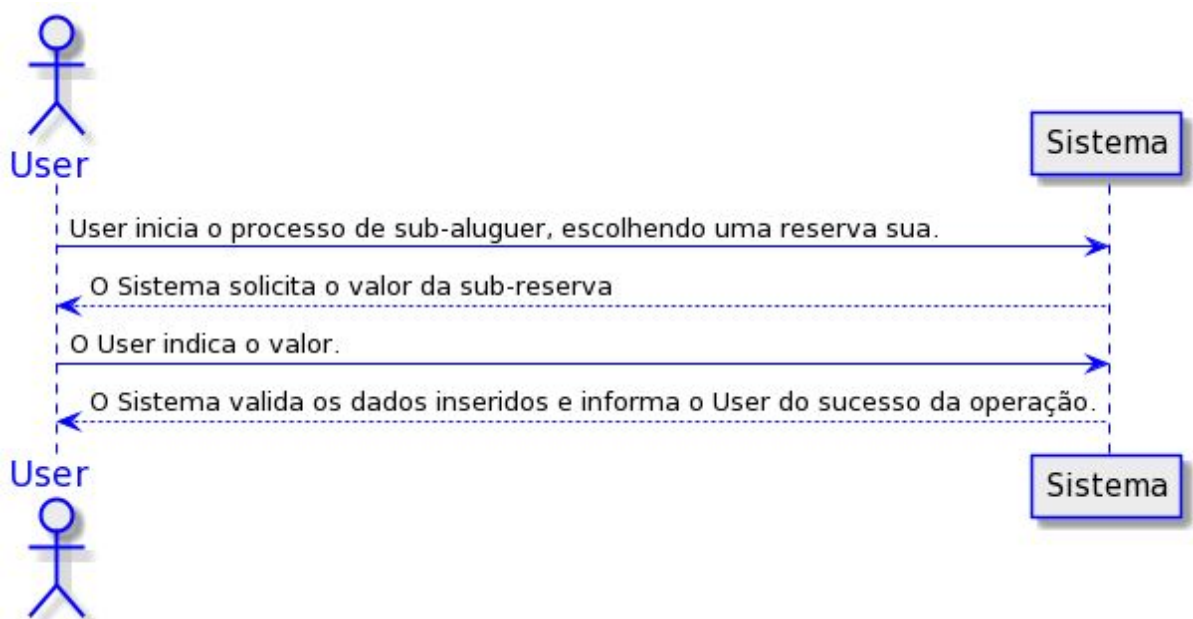
## UC7 - Utilizador sub-aluga uma reserva

### 1. Requisitos

#### 1.1 Breve Descrição

O User registado inicia o processo de sub-aluguer de uma reserva previamente finalizada. O sistema solicita informação sobre qual a reserva que pretende sub-alugar. O User introduz os dados solicitados (valor). O Sistema valida e apresenta os dados pedindo confirmação. O User confirma. O Sistema regista os dados do sub-aluguer e informa sobre o sucesso da operação.

#### 1.2 SSD



#### 1.3 Descrição Completa

##### 1.3.1 Ator Principal

User

##### 1.3.2 Partes Interessadas

- **User** - Continua a ser o detentor da reserva e pode ganhar algum rendimento adicional quando impossibilitado de usar a mesma..
- **Proprietários da Plataforma** - Menor número de cancelamentos e uma maior dinâmica entre utilizadores.

##### 1.3.3 Pré-condições

O User tem de ter reservas.

##### 1.3.4 Pós-condições

A informação do registo é armazenada no sistema e um e-mail é enviado para o User.

##### 1.3.5 Cenário Principal de Sucesso

1. O User registado inicia o processo de sub-aluguer de uma reserva previamente finalizada.
2. O sistema solicita informação sobre qual a reserva que pretende sub-alugar.
3. O User introduz os dados solicitados (valor).



4. O Sistema valida e apresenta os dados pedindo confirmação.
5. O User confirma.
6. O Sistema regista os dados do sub-aluguer e informa sobre o sucesso da operação.

### 1.3.6 Extensões (ou fluxos alternativos)

- O User solicita o cancelamento do registo da sub-reserva.
  - O caso de uso termina.
- O sistema informa que dados mínimos obrigatórios estão em falta.
  - O sistema informa que dados obrigatórios estão em falta.
  - O sistema permite a introdução dos dados em falta.
    - O User não altera os dados. O caso de uso termina.

### 1.3.7 Requisitos Especiais

-----

### 1.3.8 Variações de Tecnologias de dados

-----

### 1.3.9 Frequência de ocorrência

Poucas vezes.

### 1.3.10 Questões em Aberto

- Sem informação

## 2. Análise OO

### 2.1. Modelo de Domínio relevante para o Caso de Uso

Reservation	
<b>Id</b>	<b>int</b>
Start	datetime
End	datetime
Value	double
UserId	int
Latitude	double
Longitude	double
Qr_Code	byte[]
ParkId	int
SubReservation	bool
SubReservationValue	double

### 3. Design

#### 3.1. Racional


#### 3.2. Sistematização

Do racional resulta que as classes conceituais promovidas a classes de software são:

- Reservation

Outras classes de software identificadas:

- ReservationsController
- ReservationService
- ReservationRepository
- IReservationRepository

#### 3.3. Diagrama de Sequência

