

## Ejercicio 6

### Análisis y definición del escenario

Un contrato que permite a los inquilinos reservar y acceder a una propiedad de alquiler por un período determinado después de realizar un pago en criptomonedas.

### Diseño

#### Caso de uso

Identificador	Alquiler de una propiedad
Descripción	El sistema deberá permitir al inquilino solicitar el alquiler de una propiedad
Secuencia Normal	1.- El usuario realiza una petición al contrato
	2.- El sistema comprueba si la cantidad de criptomoneda es suficiente para la realización del contrato
	2a.- Si el usuario posee suficiente crédito, se le asigna la propiedad durante el tiempo especificado y se cobra el precio de su alquiler al usuario
	2.b- Si el usuario no posee suficiente crédito, se deniega la transacción
Importancia	De alta importancia para el funcionamiento de la aplicación

### Contenido del contrato inteligente

#### Datos

- Propiedad**
- Estructura de la propiedad a alquilar. Contiene:
  - nombre** - Nombre de la propiedad.
  - dirección** - Dirección física de la propiedad.
  - estado** - Estado físico de la propiedad.
  - precioDia** - Precio de la propiedad por día.
- Alquiler**
- Estructura para almacenar información sobre cada alquiler.
  - inquilino** - Dirección del inquilino.
  - inicio** - Fecha de inicio del alquiler.
  - fin** - Fecha de finalización del alquiler.
- propiedadADuenho** - Mapping que relaciona una propiedad con su dueño.
- duenhoAPropiedad** - Mapping que relaciona a un usuario con el número de propiedades que posee.

**propiedadAAquiler** - Mapping que relaciona una propiedad con un array de alquileres.

## Funciones

**reservar(uint \_idPropiedad, uint \_inicioAlquiler, uint \_finAlquiler) public payable {}**

Permite a un inquilino alquilar una propiedad.

**renovar(uint \_idPropiedad, uint \_inicioAlquiler, uint \_finAlquiler) public payable {}**

Permite a un inquilino volver a alquilar una propiedad que ya ha alquilado anteriormente.

**cancelarReserva(uint \_idPropiedad, uint \_inicioAlquiler, uint \_finAlquiler) public {}**

Permite a un inquilino cancelar un alquiler.

**anadirPropiedad(string memory \_nombre, string memory \_direccion, string memory \_estado, uint \_precioDia) public {}**

Permite a un propietario añadir una propiedad.

**eliminarPropiedad(uint \_idPropiedad) public onlyOwnerOf(\_idPropiedad) {}**

Permite a un propietario eliminar una de sus propiedades.

**modificarPropiedad(uint \_idPropiedad, string memory \_nombre, string memory \_direccion, string memory \_estado) public onlyOwnerOf(\_idPropiedad) {}**

Permite a un propietario modificar una de sus propiedades.

**cancelarReservaInquilino(uint \_idPropiedad, uint \_inicioAlquiler, uint \_finAlquiler, address \_inquilino) public onlyOwnerOf(\_idPropiedad) {}**

Permite a un propietario cancelar el alquiler de un inquilino.

**cancelarReservasInquilino(uint \_idPropiedad, address \_inquilino) public onlyOwnerOf(\_idPropiedad) {}**

Permite a un propietario cancelar todos los alquileres de un inquilino.

**cancelarReservasPropiedad(uint \_idPropiedad) public onlyOwnerOf(\_idPropiedad) {}**

Permite a un propietario cancelar todos los alquileres de una propiedad.

**\_terminoElAlquiler(uint \_idPropiedad, address \_inquilino) internal view returns (bool) {}**

Comprueba si el alquiler de un inquilino ya ha terminado.

**\_rangoAlquileres(uint \_idPropiedad, uint \_inicioAlquiler, uint \_finAlquiler) internal view returns (bool) {}**

Comprueba si una propiedad está alquilada durante un periodo de tiempo.

**\_esInquilino(uint \_idPropiedad, address \_inquilino) internal view returns (bool) {}**

Comprueba si un inquilino ha alquilado una propiedad.

**\_eliminarAlquiler(uint \_idPropiedad, uint \_id) internal {}**

Función interna para eliminar alquileres.

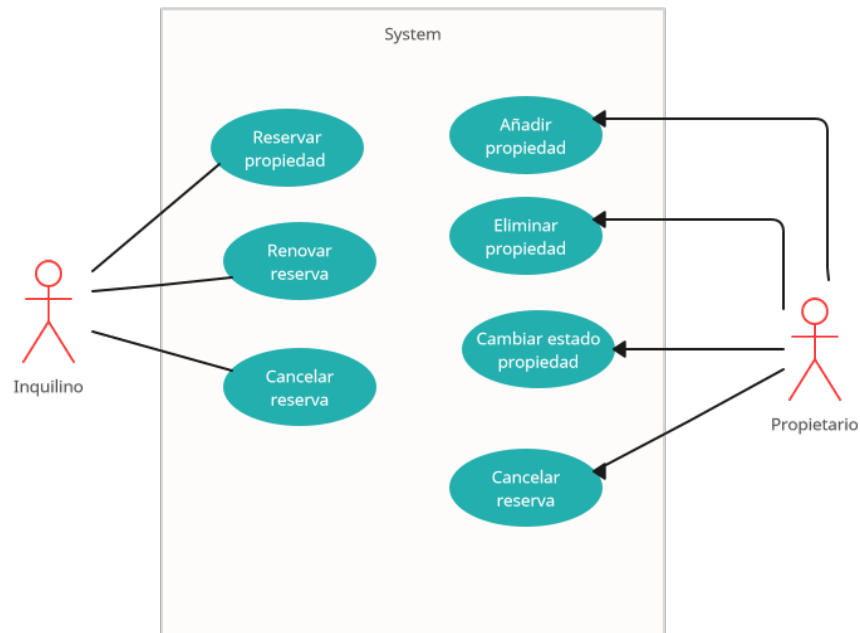
**transferirFondosAlDuenho() public {}**

Función para acceder a los fondos del contrato.

## Usuarios del sistema

Dueños de propiedades: Usuarios que ofrezcan propiedades a alquilar.

Inquilinos: Usuarios que deseen alquilar propiedades.



## Implementación

Disponible en [github](#).

# Pruebas

## Prueba reservar

Aquí se puede observar cómo la transacción ha sido cancelada porque no tengo suficiente ETH cómo para alquilar la propiedad.

The screenshot shows the Remix IDE interface. On the left, the 'Deployed Contracts' panel displays the 'ALQUILER AT 0xD8B...33F8 (MEMORY)' contract. The 'reservar' function is selected, with parameters: `_idPropiedad: 0`, `_inicioAlquiler: 1638352800`, and `_finalAlquiler: 1638871200`. The 'transaction' button is highlighted in red. The main editor shows the Solidity code for the contract, including functions like `propiedades[i] = propiedades[i+1]`, `propiedades.pop()`, `modificarPropiedad`, and `cancelarReservaInquilino`. The console on the right shows the following messages:

```
transaction to alquiler.reservar pending ...
transaction to alquiler.reservar errored: Error occurred: revert.
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "La cantidad pagada es insuficiente para el alquiler.".
Debug the transaction to get more information.
```

Below the messages, the debug information is displayed:

```
[vm] From: 0x583...addC4 to: alquiler.reservar(uint256,uint256,uint256) 0x8B...33F8 values: 0 wei
data: 0x0fa...730a8 logs: 0 hash: 0xab6...599d1
```

## Prueba renovar

Lo mismo se puede observar con renovar.

The screenshot shows the Remix IDE interface. On the left, the 'Deployed Contracts' panel displays the 'ALQUILER AT 0xD8FC...9AB36 (MEMORY)' contract. The 'renovar' function is selected, with parameters: `_idPropiedad: 0`, `_inicioAlquiler: 1638352800`, and `_finalAlquiler: 1638352800`. The 'transaction' button is highlighted in red. The main editor shows the Solidity code for the contract, including the `contract alquiler` definition, `constructor`, and `modifier onlyOwnerOf`. The console on the right shows the following messages:

```
transaction to alquiler.renovar pending ...
transaction to alquiler.renovar errored: Error occurred: revert.
revert
The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.
```

Below the messages, the debug information is displayed:

```
[vm] From: 0x583...addC4 to: alquiler.renovar(uint256,uint256,uint256) 0x8FC...9AB36 values: 0 wei
data: 0x2c9...747a8 logs: 0 hash: 0x86a...7a0b6
```

## Prueba anhadirPropiedad

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying the 'anhadirPropiedad' function with its parameters: `_nombre` (Casa blanca), `_direccion` (RIO JAMARA 36, CUAUHTEMOC, 91069), `_estado` (habitable), and `_precioDla` (25). Below the function, there are several transaction records. The main editor shows the Solidity code for the 'alquiler' contract, including the constructor and the `onlyOwnerOf` modifier. The bottom panel shows the console output, indicating that the transaction to `alquiler.anhadirPropiedad` is pending.

## Prueba eliminarPropiedad

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying the 'eliminarPropiedad' function with its parameter: `_idPropiedad` (1). Below the function, there are several transaction records. The main editor shows the Solidity code for the 'alquiler' contract, including the `eliminarPropiedad` function. The bottom panel shows the console output, indicating that the transaction to `alquiler.eliminarPropiedad` is pending.

## Prueba modificarPropiedad

### DEPLOY & RUN TRANSACTIONS

Transactions recorded 1 1

#### Deployed Contracts

▼ ALQUILER AT 0xD8B...33F8B (MEMORY)

Balance: 0 ETH

Contract Name	Address
anhadPropiedad	string_nombre, string_direccion, string_estado, uint256_precioDia
cancelarReserva	uint256_idPropiedad, uint256_inicioAlquiler, uint256_finAlquiler
cancelarReserva	uint256_idPropiedad, uint256_inicioAlquiler, uint256_finAlquiler, address_inquilino
cancelarReserva	uint256_idPropiedad, address_inquilino
cancelarReserva	uint256_idPropiedad
cancelarReserva	uint256_idPropiedad

#### modificarPropiedad

\_idPropiedad: 1

\_nombre: Casa roja

\_direccion: RIO JAMAPA 36 CUALTEMOC 91069

\_estado: habitable

\_precioDia: 25

Caldata
Parameters
transact

```

85     propiedades[i] = propiedades[i+1];
86   }
87   propiedades.pop();
88 }
89
90 function modificarPropiedad(uint_idPropiedad, string memory_nombre, string memory_direccion,
91   string memory_estado, uint256_precioDia, uint256_idPropiedad) public {
92   PropiedadModificar propiedadModificar = propiedades[_idPropiedad];
93   propiedadModificar.nombre = _nombre;
94   propiedadModificar.direccion = _direccion;
95   propiedadModificar.estado = _estado;
96   propiedadModificar.precioDia = _precioDia;
97 }
98
99 function cancelarReservaInquilino(uint_idPropiedad, uint_inicioAlquiler, uint_finAlquiler,
100   address_inquilino) public {
101   require(_esInquilino(_idPropiedad, _inquilino));
102   for(uint j = 0; j < propiedadAAlquilar[_idPropiedad].length; j++){

```

☒ listen on all transactions
 ☐ Search with transaction hash or address

transact to alquiler.modificarPropiedad pending ...

✓ [ev] from: 0x583...e5dc4 to: alquiler.modificarPropiedad(uint256,string,string,string,uint256) 0xd8b...33f8b  
 value: 0 wei data: 0x199...00000 logs: 0 hash: 0xa61...b021f