Petting Aavegotchis

GotchiCare

GotchiCare uses Gelato Autoamte to provide an automated Aavegotchi petting service.

Gelato callsexec here which pets the Aavegotchis and settles the fee.

```

Copy functionexec(CareInfocalldata_careInfo)external{ require(msg.sender==executor,"Carer: Only executor");

bytes32_receipt=getReceipt(_careInfo.owner,_careInfo);

require( caringOwners.contains(_careInfo.owner), "Carer: exec: Owner has not started" ); require( _careInfo.owner==ownerOfReceipt[_receipt], "Carer: exec: Receipt does not match" );

operator.pet(_careInfo.gotchis,_careInfo.owner);

payWages(_careInfo.owner,_careInfo.rate);

CareInfomemorynewCareInfo=CareInfo( _careInfo.owner, _careInfo.pets.add(1), _careInfo.rate, _careInfo.gotchis );

updateOwnerInfo(newCareInfo);

}

```

This resolver loops through an array of subscribed users. For each user, it checks if their aavegotchi's petting time has reached.

```

Copy functionchecker() external view returns(boolcanExec,bytesmemoryexecPayload) { address[]memory_caringOwners=careCentre.getCaringOwners(); uint256_length=_caringOwners.length;

for(uint256i=0; i<_length; i++) { ICareCentre.CareInfomemory_careInfo=careCentre.getCareInfoByOwner( _caringOwners[i] );

if(!ownerHasBalance(_caringOwners[i],_careInfo.rate))continue; if(!isApproved(_caringOwners[i]))continue;

uint256[]memory_gotchis=_careInfo.gotchis;

uint256_lastInteracted=gotchiFacet .getAavegotchi(_gotchis[0]) .lastInteracted;

uint256_nextInteract=_lastInteracted+12hours;

if(block.timestamp>=_nextInteract) { canExec=true;

execPayload=abi.encodeWithSelector( ICareCentre.exec.selector, _careInfo );

return(canExec,execPayload); } }

canExec=false; }

```

Previous Rewards Payout Next Code Repositories Last updated1 year ago On this page