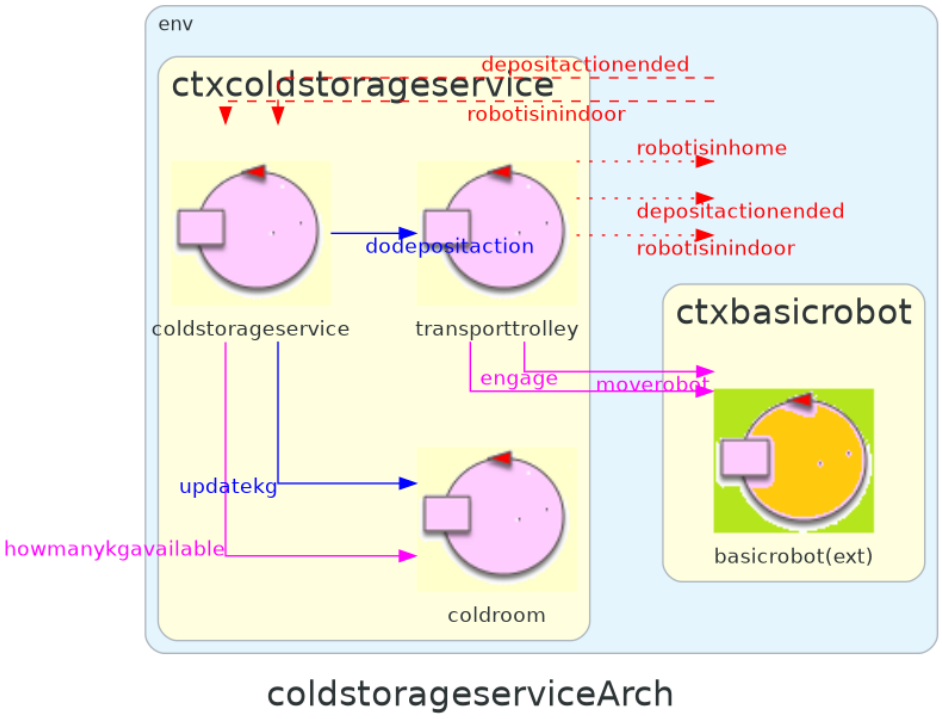
ColdStorageSprint2V0

Introduzione

Goal conseguiti nello sprint1

- sviluppo del core business ColdStorageService + TransportTrolley

architettura logica sprint1



coldstorageserviceArch

Goal dello sprint2

- introduzione della GUI nel sistema ottenuto nello sprint1 e testing
  Si noti che lo sviluppo del progetto dello sprint2 è da intendersi come estensione del progetto dello sprint1

Requirements

a ServiceAccessGUI that allows an human being to see the current weight of the material stored in the ColdRoom and to send to the ColdStorageService a request to store new **FW** kg of food. If the request is accepted, the services return a ticket that expires after a prefixed amount of time (**TICKETTIME** secs) and provides a field to enter the ticket number when a Fridge truck is at the INDOOR of the service.

SERVICE USER STORY

- A Fridge truck driver uses the ServiceAccessGUI to send a request to store its load of FW kg. If the request is accepted, the driver drives its truck to the INDOOR of the service, before the ticket expiration time TICKETTIME.
- When the truck is at the INDOOR of the service, the driver uses the ServiceAccessGUI to enter the ticket number and waits until the message charge taken (sent by the ColdStorageService) appears on the ServiceAccessGUI. At this point, the truck should leave the INDOOR.
- When the service accepts a ticket, the transport trolley reaches the INDOOR, picks up the food, sends the charge taken message and then goes to the ColdRoom to store the food.
- When the deposit action is terminated, the transport trolley accepts another ticket (if any) or returns to HOME.

Requirements analysis

- Per **current weight** si intende il numero di kg contenuti nella ColdRoom effettivi. Viene quindi formalizzato con un numero intero positivo.
- Dal punto 1 della service user story, si evince che ci sarà un campo in input dove l'utente può inserire il numero di kg FW per richiedere il ticket.
- Una volta che l'utente ha inserito il ticket number, non deve essere possibile per lui inserire un altro ticket prima che ottenga alcuna risposta dalla ServiceAccessGui
- In qualunque momento deve essere possibile richiedere ticket.

Dal precedente sprint si ricorda che:

- Non deve succedere che un camion, ricevuto il proprio ticket si veda rifiutata l'operazione di scarico una volta arrivato in INDOOR a patto che il ticket non sia scaduto.
- la richiesta di un ticket può avvenire mentre sono ancora in corso operazioni di scarico precedenti

Problem Analysis

comunicazione con il sistema

da requisiti si prevede la comunicazione tramite i seguenti messaggi:

- Request storefood : storefood(KG)
- Reply ticketaccepted : ticketaccepted(TICKETCODE, TICKETSECRET, TIMESTAMP)
- Reply ticketdenied : ticketdenied(ARG)
- Request sendticket: sendticket(TICKETCODE, TICKETSECRET)
- Reply chargetaken : chargetaken(ARG)
- Reply ticketexpired: ticketexpired(ARG)
- Reply ticketrejected: ticketrejected(ARG)

Il problema della disponibilità della GUI

da requisiti si evince che la gui deve essere sempre disponibile ad accettare richieste, le comunicazioni intraprese devono essere di natura **NON BLOCCANTE**

Trasparenza della gui

per il principio di clean architecture la GUI deve operare in maniera trasparente al sistema

L'attore service ACCESSGUI

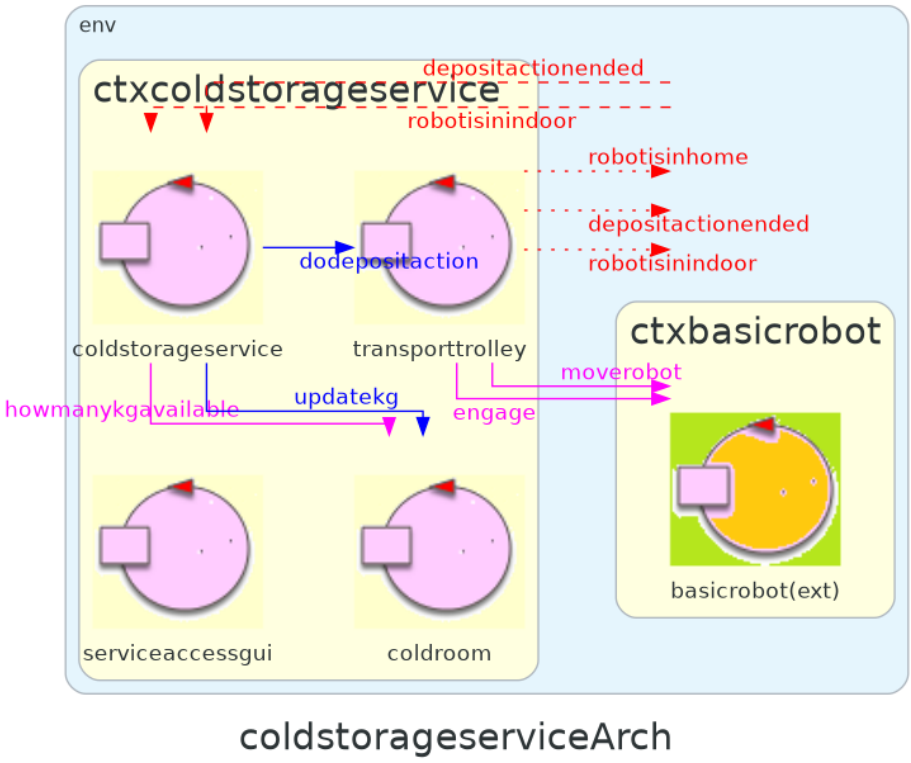
come introdotto nella architettura dello sprint0 si aggiunge al sistema un attore in grado di comunicare con coldstorageservice.

La scelta del contesto

data l'assenza di particolari requisiti che richiedano di inserire la serviceaccessgui in un contesto diverso si decide di inserirla nello stesso contesto del sistema

Architettura logica dopo analisi del problema

Al sistema precedente si aggiunge un attore *ServiceAccessGui* che si interfaccia con ColdStorageService per la richiesa di ticket



coldstorageserviceArch

PROGETTAZIONE

Tutto il codice della parte di progettazione è consultabile al seguente link github

L'alieno WEBGUI

si decide di implementare un sistema alieno in grado di fornire una interfaccia web per consentire l'interazione utente, dopo una breve analisi dei tool disponibili si presentano le seguenti possibilità di implementazione:

- **NODEJS**
- **SPRING**
  per rapidita di implementazione si decide di sviluppare un primo prototipo con NODEJS e il framework EXPRESS per andare alla ricerca di problemi di natura implementativa

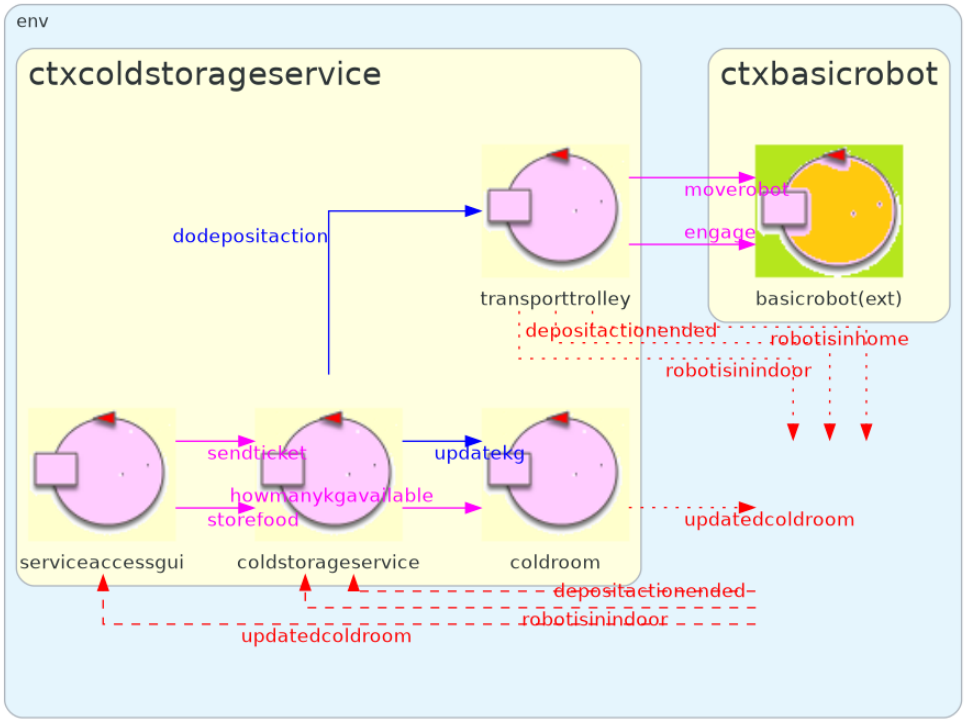
Il problema delle chiamate bloccanti e della gestione della sessione

da requisiti si ha che la comunicazione del messaggio CHARGETAKEN deve essere una reply a una request precedentemente fatta dalla SERVICEACCESSGUI, da requisiti si ha che la gui deve essere sempre disponibile, questo porta a due soluzioni implementative:

- ### chiamate bloccanti
  - l'attore **SERVICEACCESSGUI**, ricevuta una richiesta da parte della **WEBGUI** esegue a sua volta la richiesta a **COLDSTORAGESERVICE** e si mette in attesa della risposta che poi fornirà alla **WEBGUI**
  - questo approccio però non soddisfa il requisito che la **WEBGUI** sia sempre disponibile in quanto anche se la **WEBGUI** fosse in grado di inviare piu richieste alla **SERVICEACCESSGUI** questa **richiesta non sarebbe servita fino al termine del servizio delle richieste precedenti**
- ### chiamate non bloccanti con gestione della sessione
  - in questa seconda versione dell'implementazione si ha che **SERVICEACCESSGUI** dopo aver inviato una richiesta a **COLDSTORAGESERVICE** non attende la risposta ma si mette in attesa di altre richieste
  - questo porta un problema legato alla gestione della sessione in quanto **non si è piu in grado di determinare a quale richiesta fa riferimento una determinata risposta** quando questa giunge a **SERVICEACCESSGUI**

di seguito si presenta l'architettura di progettazione realizzata per testare le problematiche sopra riportate, per il codice fare riferimento al file coldstorageservice-progettazione.qak

Architettura finale progettazione



coldstorageserviceArch

By Caterina Leonelli email: caterina.leonelli2@studio.unibo.it, GIT repo: https://github.com/RootLeo00/sw-eng.git

By Matteo Longhi email: matteo.longhi5@studio.unibo.it GIT repo: https://github.com/carnivuth/iss_2023_matteo_longhi.git