BARBOT

DOCUMENTAZIONE PDDL APP

* Bonomi Andrea matricola 731763
* Freschi Luca matricola 731609

Indice

Indice

1. Obiettivo del progetto pag. 2
2. Dominio pag. 4
   1. Tipi
   2. Predicati
   3. Funzioni
   4. Azioni
3. Problema pag. 7
   1. Inizializzazione
   2. Mappa
   3. Livello delle bottiglie
   4. Ingredienti richiesti
4. Planner pag. 8

1. OBIETTIVO DEL PROGETTO

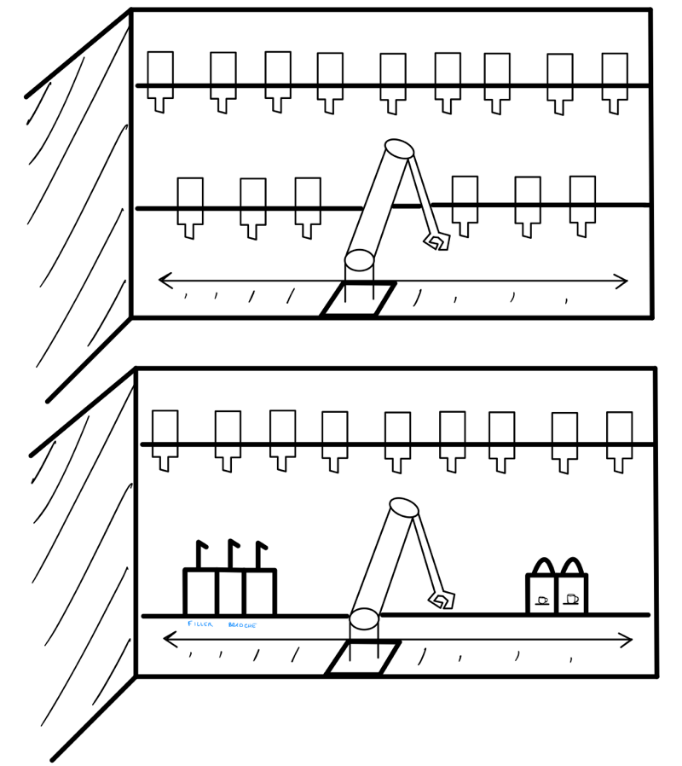
L’applicazione PDDL “BarBot” ha come obiettivo quello di formulare un piano di azioni per un robot collaborativo all’interno di un bar. Il suo compito principale è la composizione dei drink: arrivato un ordine, prende il bicchiere, si reca in tutte le posizioni necessarie per ottenere gli ingredienti richiesti e consegna lo shaker in una posizione nota, comunicando la fine dell’ordine. Tiene inoltre conto dello stato degli ingredienti, avvisando quando è necessario effettuare il cambio bottiglia, secondo parametri impostati.

Dall’apposita applicazione, è possibile sia ordinare cocktail classici, tipici di ogni menù, sia drink personalizzabili, avendo la possibilità di aggiungere manualmente ogni ingrediente e creare quindi la propria ricetta.

Gli ordini verranno caricati nel problema PDDL, che attraverso un planner compatibile pianificherà l’ordine delle azioni, così da risolvere tutti gli ordini.

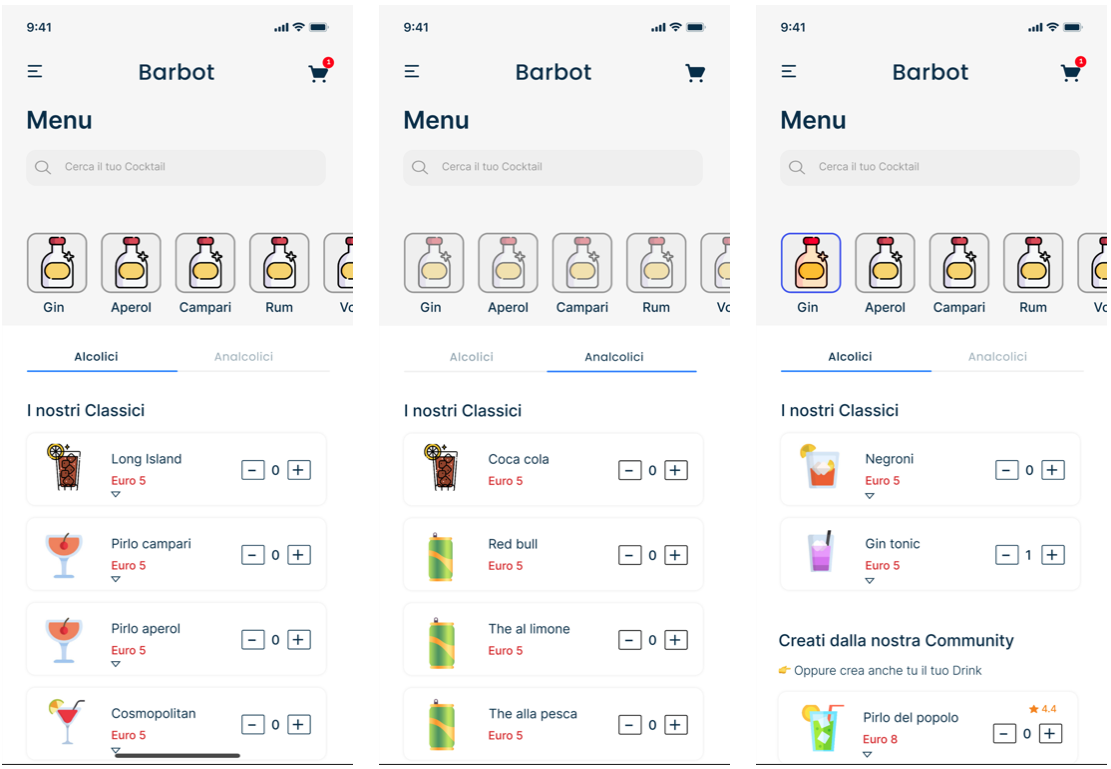
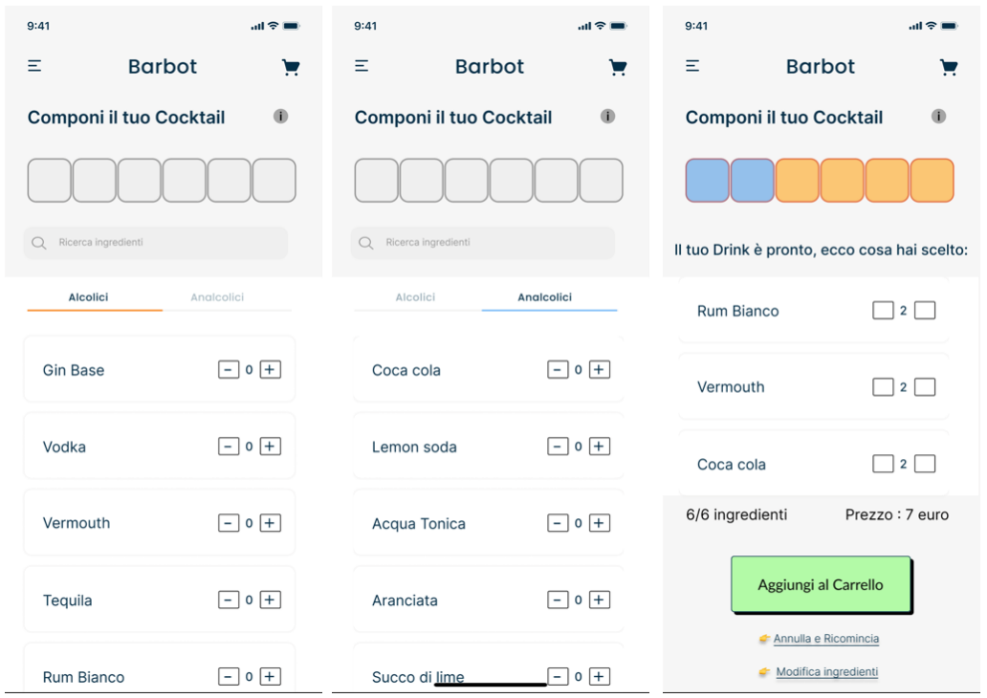
Il robot, essendo collaborativo, fornirà al barman il drink composto dei suoi ingredienti, così che possa eseguire l’eventuale miscelazione. Il punto di partenza in cui il robot afferra lo shaker e il punto di arrivo in cui lo deposita sono fissi e si ipotizza, data la presenza del barman, la continua disponibilità degli shaker e la possibilità di depositare lo shaker pieno.

Tutti i drink sono composti da 6 slot ingrediente, implicazione che permette di creare tutti i drink classici seguendo la relativa ricetta, ma anche un’ampia possibilità di personalizzazione, sia modificando le dosi classiche, sia sperimentando nuove composizioni.



Bozza grafica dell’implementazione del robot collaborativo

Alcune schermate dell’applicazione (mock-up), tra cui la home e la sezione drink personalizzati.



Applicazione completa:

[https://www.figma.com/Barbot](https://www.figma.com/proto/MxIhQvA7TVTMJ7VtCXGYkS/BarBot-Prototype?node-id=519-2856&scaling=scale-down&page-id=0%3A1&starting-point-node-id=518%3A7960&show-proto-sidebar=1)



2. DOMINIO

L’applicazione necessita di un planner numerico, che supporti i fluents.

## 2.1 tipi

Il programma definisce quattro tipi, o oggetti:

* *Drink* à Parte centrale del programma, ne costituisce l’obiettivo;
* *Bottle* (extends Agent)à Rappresenta le bottiglie da cui il robot prende gli ingredienti, costituiscono una mappa
* *Station* (extends Agent) à Rappresentano il punto di inizio da cui il robot deve partire per eseguire l’ordine e il punto in cui deve lasciare lo shaker e quindi chiudere l’ordine.
* *Agent* à superclasse che permette di creare una mappa, costituita dalle bottiglie e dalle stazioni, così che il robot possa muoversi e ordinare le azioni.

## 2.2 predicati

(located ?x - agent)

Indica la posizione corrente. La definizione sul tipo Agent permette di passare dalle stazioni Start e Stop alle bottiglie.

(isnear ?x ?y - agent )

Definisce la mappa, indicando la vicinanza tra gli agenti.

(beStart ?x - station)

Predicato usato per distinguere lo Start dallo Stop.

(order-in-progress ?d - drink)

Definisce l’inizio della composizione per il drink selezionato.

        (served ?x - drink)

Definisce la conclusione del drink e la sua consegna.

        (warningLevel ?b - bottle)

Avvisa la necessità di effettuare un cambio bottiglia

(handFree)

Definisce lo stato del robot, impegnato durante la composizione del drink, libero altrimenti.

## 2.3 funzioni

(required-ingredient ?d - drink ?b - bottle)

Attraverso questa funzione si definiscono gli ingredienti necessari per ogni drink, indicando la quantità di ognuno (da 1 a 6).

(level ?b - bottle)

Definisce il livello della bottiglia, si decrementa ad ogni composizione

(used-ingredient ?d - drink)

Tiene il conto degli ingredienti utilizzati; arrivato a 6 indica il completamento del drink

## 2.4 azioni

 (:action moveToAgent

        :parameters (?from ?to - agent)

        :precondition (and (located ?from) (isnear ?from ?to))

        :effect (and (located ?to) (not(located ?from))  )

    )

Azione di movimento.

Parametri: due agenti, *?from* in cui è posizionato, *?to* dove vuole arrivare

Precondizioni: essere posizionato nell’agente *?from*, i due agenti devono essere vicini

Effetto: sposta il posizionamento da *?from* a *?to*

(:action start-order

        :parameters (?d - drink ?s - station)

        :precondition (and (located ?s) (beStart ?s) (not(served ?d)) (handFree) )

        :effect (and (order-in-progress ?d) (not(handFree)) )

    )

Azione di inizio composizione.

Parametri: Drink scelto da comporre, stazione di partenza (Start)

Precondizioni: essere posizionato nella stazione *?s*, indicato come Start (*beStart*), il drink non deve essere già servito e il robot deve essere libero.

Effetto: il drink diventa in composizione, il robot non è libero

(:action compose

        :parameters (?d - drink  ?b - bottle)

        :precondition (and (order-in-progress ?d) (located ?b)

(> (required-ingredient ?d ?b) 0) (> (level ?b) 6) )

        :effect (and (decrease (required-ingredient ?d ?b)1) (decrease (level ?b) 1)

(increase (used-ingredient ?d) 1)) )

)

Azione di composizione.

Parametri: drink e ingrediente

Precondizioni: drink in composizione, posizionato sulla bottiglia, il drink deve richiedere l’ingrediente, il livello della bottiglia deve essere sufficiente (settato a 6 per permettere di effettuare il cambio bottiglia tra un ordine e l’altro).

Effetto: decrementa la funzione (required ingredient), decrementa il livello della bottiglia, incrementa gli ingredienti utilizzati per il drink.

(:action warning

        :parameters (?b - bottle)

        :precondition (and (< (level ?b) 9) (handFree) )

        :effect (and (warningLevel ?b) )

)

Azione di avviso per il cambio bottiglia

Parametri: Bottiglia

Precondizioni: livello della bottiglia sotto una soglia, robot libero (permette di effettuare il cambio bottiglia tra un ordine el’altro).

Effetto: rende vero il predicato di warning.

(:action change

        :parameters (?b - bottle)

        :precondition (and   (warningLevel ?b) (handFree))

        :effect (and  (not(warningLevel ?b)) (assign (level ?b) 12))

)

Azione che indica l’avvenuto cambio bottiglia

Parametri: Bottiglia

Precondizioni: predicato di warning attivo, robot libero

Effetto: assegna un nuovo valore al livello bottiglia cambiata, disattiva il predicato di warning

(:action finish-order

        :parameters (?d - drink ?s - station)

        :precondition (and  (located ?s) (not(beStart ?s)) (not(served ?d))

                            (= (used-ingredient ?d) 6) )

        :effect (and (served ?d) (handFree) (not(order-in-progress ?d)))

    )

Azione di termine composizione.

Parametri: Drink composto, stazione di arrivo (Stop)

Precondizioni: essere posizionato nella stazione *?s*, indicato come Stop (not(*beStart*)), il drink non deve essere già servito, gli ingredienti usati sono uguali a 6.

Effetto: drink servito e non più in composizione, robot libero.

3 problema

All’interno del problema, ci sono tre parti principali da definire.

## 3.a inizializzazione

Predicati di stato iniziale: definire le stazioni di Start, punto di partenza, e Stop; robot libero.

(located Start)

(beStart Start)

(not(beStart Stop))

(handFree)

## 3.b mappa

Definire i rapporti di vicinanza tra le bottiglie e le stazioni, creando i percorsi che il robot può seguire. Serve definire i predicati in entrambi i versi.

Es.

(isnear Gin Vermuth )

(isnear Gin Campari )

(isnear Vermuth Gin )

(isnear Campari Gin  )

(isnear Start Gin )

(isnear Campari Stop )

(isnear Stop Campari )

(isnear Stop Start)

## 3.c livello delle bottiglie

Definire il livello delle bottiglie. A livello logico, nello stato iniziale dovrebbero avere tutte uno stesso livello, ma nulla vieta di inizializzarle a livelli diversi.

Es.

(=(level Campari) 10)

(=(level Gin) 66)

(=(level Vermuth) 66)

## 3.d ingredienti richiesti

Per ogni drink vanno specificati gli ingredienti che necessita e la relativa quantità. A seguito dell’inizializzazione degli ingredienti usati a zero, indicare drink, ingrediente e quantità.

Es.

 (= (used-ingredient Negroni) 0)

(= (required-ingredient Negroni Campari) 2)

(=(required-ingredient Negroni Gin)2)

(=(required-ingredient Negroni Vermuth)2)

4. planner

Come già specificato lungo la documentazione, l’applicazione definisce funzioni e si basa in generale sui fluents, definiti a partire dalla versione 2.1 di PDDL. Perciò, è necessario utilizzare un planner numerico, con specifico supporto ai fluents.