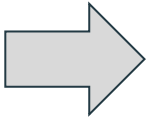
LPaaS-Client

Sistemi distribuiti aa 2020-2021

Introduzione: LPaaS

- Gli standard moderni sono le architetture orientate ai servizi
- La programmazione logica supporta molto bene i sistemi intelligenti



Logical Programming as a Service

Una re-interpretazione della programmazione logica nel distribuito che permette a dispositivi di ogni tipo di sfruttare la programmazione logica

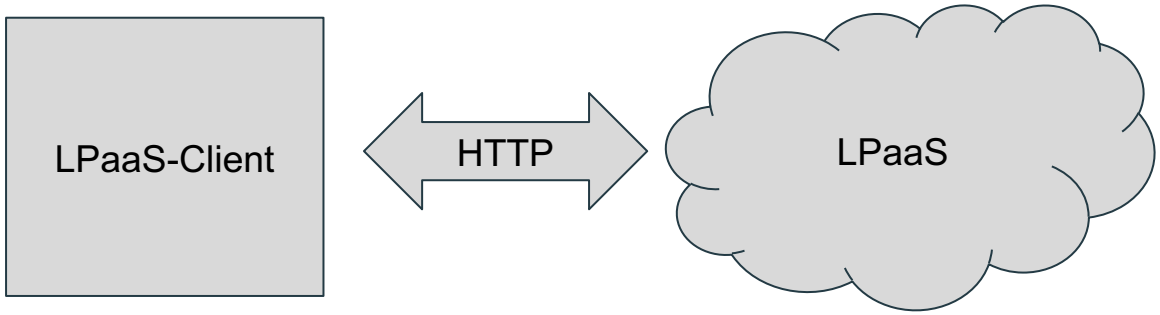
LPaaS vs LP: differenze

- I dati (le teorie) e il motore di esecuzione non è accessibile dall'esterno
- LPaaS introduce la necessità di gestire:
  - Spazio
  - Tempo
  - Contesto
- Non è più possibile utilizzare *assert/retract*
- Comunicazione di base **stateless**
- Possibilità di supportare richieste **stateful**

LPaaS: vantaggi

- Possibilità di portare la programmazione logica ad un insieme eterogeneo di device
- Possibilità di supportare qualsiasi linguaggio di programmazione tramite API RESTful
- Capacità di gestire un flusso di dati in input
    - Ad esempio sensori che aggiornano periodicamente la **knowledge base**
- Possibilità di fornire un flusso di soluzioni

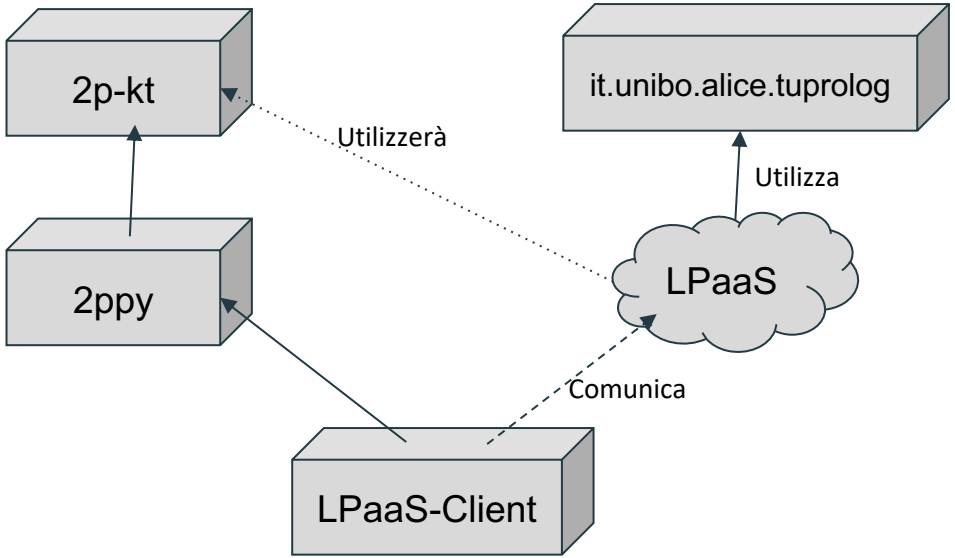
LPaaS-Client: Python



Obiettivi principali

- Abilitare gli sviluppatori Python all'utilizzo di **LPaaS**
- Fornire API sia sincrone che asincrone
- Sfruttare 2ppy per la manipolazione dei termini logici
- Fornire il client come libreria Python
- Validare il prodotto tramite unit-test, system-test e performance-test
- Fornire un esempio concreto di utilizzo

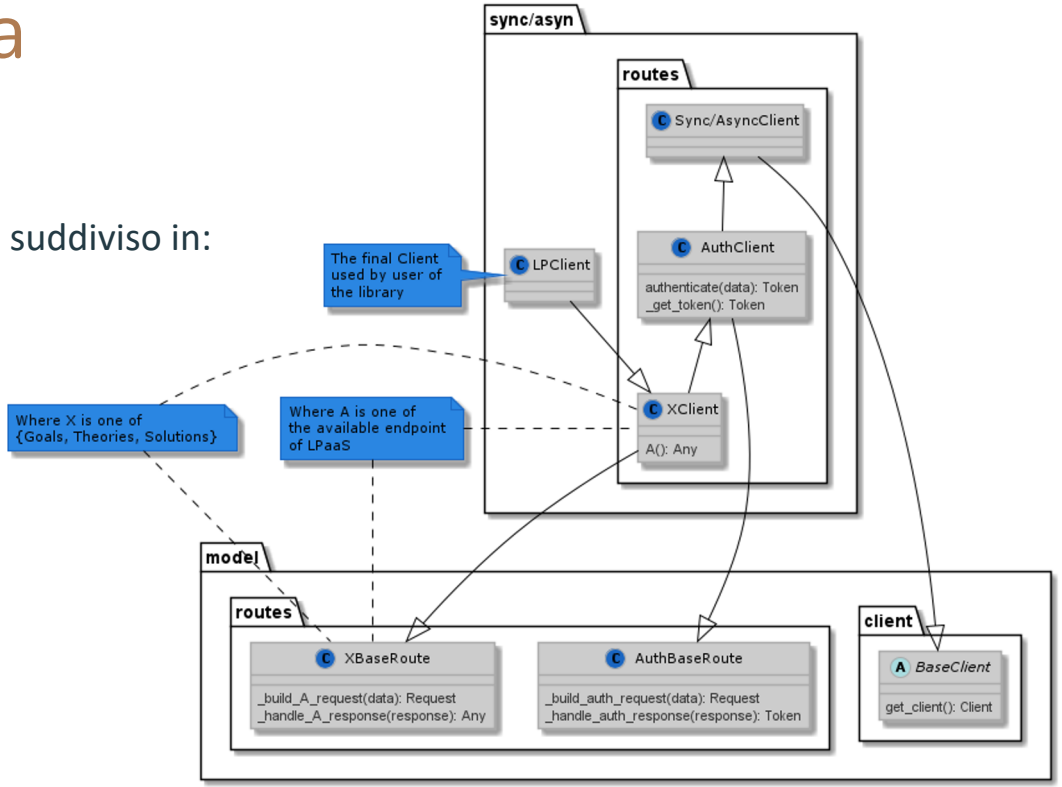
LPaaS-Client: Dipendenze



Architettura

Il client, come il server, è suddiviso in:

- Authentication
- Goals
- Theories
- Solutions



Esempio: Ping-Pong

- Configuratore:
  - Ha il compito di creare la teoria, la goal list e inizializzare una soluzione *S*
- Pinger (source-consumer):
  - Controlla la soluzione *S*
  - Se è il suo turno aggiorna la teoria con il valore *ping*
- Ponger (source-consumer):
  - Controlla la soluzione *S*
  - Se è il suo turno aggiorna la teoria con il valore *pong*

Teoria e Goal-List

Teoria

```
value(ping).
turn(pinger) :- value(pong).
turn(ponger) :- value(ping).
```

Goal-list

```
turn(X)
```