Specifica requisiti software

“Piazzale azienda ZP”
versione 1.0

Valentino Pisi | 218115@studenti.unimore.it

1.0. Introduzione

1.1. Requisiti

All’esame deve essere mostrata l’applicazione in esecuzione e deve essere presentata una relazione che descrive il progetto e deve comprendere:

- Descrizione dei requisiti, ed in particolare delle funzionalità messe a disposizione (es es. tramite **SRS**);
- Descrizione dell’architettura (ad es. tramite diagramma a blocchi o **UML**);
- Descrizione dei **protocolli** usati (client-server o peer-to-peer, ad es. tramite diagrammi UML).

1.2. Scopo

Lo scopo di questo progetto è sviluppare un sistema distribuito utilizzando il framework Pyro per creare un magazzino chiamato "Piazzale". Il sistema consente agli utenti autorizzati di inserire nuovi materiali nel magazzino e memorizzarne le informazioni, tra cui il nome del materiale, il nome dell'utente che lo ha inserito e la data/ora di inserimento. L'obiettivo principale del progetto è fornire un'interfaccia intuitiva e sicura per la gestione dei materiali all'interno di un ambiente distribuito.

1.3. Finalità del progetto

La finalità del progetto è dimostrare l'efficacia dell'utilizzo del file system distribuito tramite l'implementazione del magazzino distribuito chiamato "Piazzale". Utilizzando il framework Pyro, il sistema si basa su una comunicazione client-server per consentire l'interazione tra i componenti. L'obiettivo generale è fornire una soluzione scalabile ed efficiente per la memorizzazione e la gestione dei materiali nel contesto distribuito del magazzino. Il progetto mira a soddisfare le esigenze di archiviazione distribuita, garantendo nel contempo un accesso sicuro e controllato alle funzionalità offerte ai soli utenti autorizzati.

1.4. Glossario

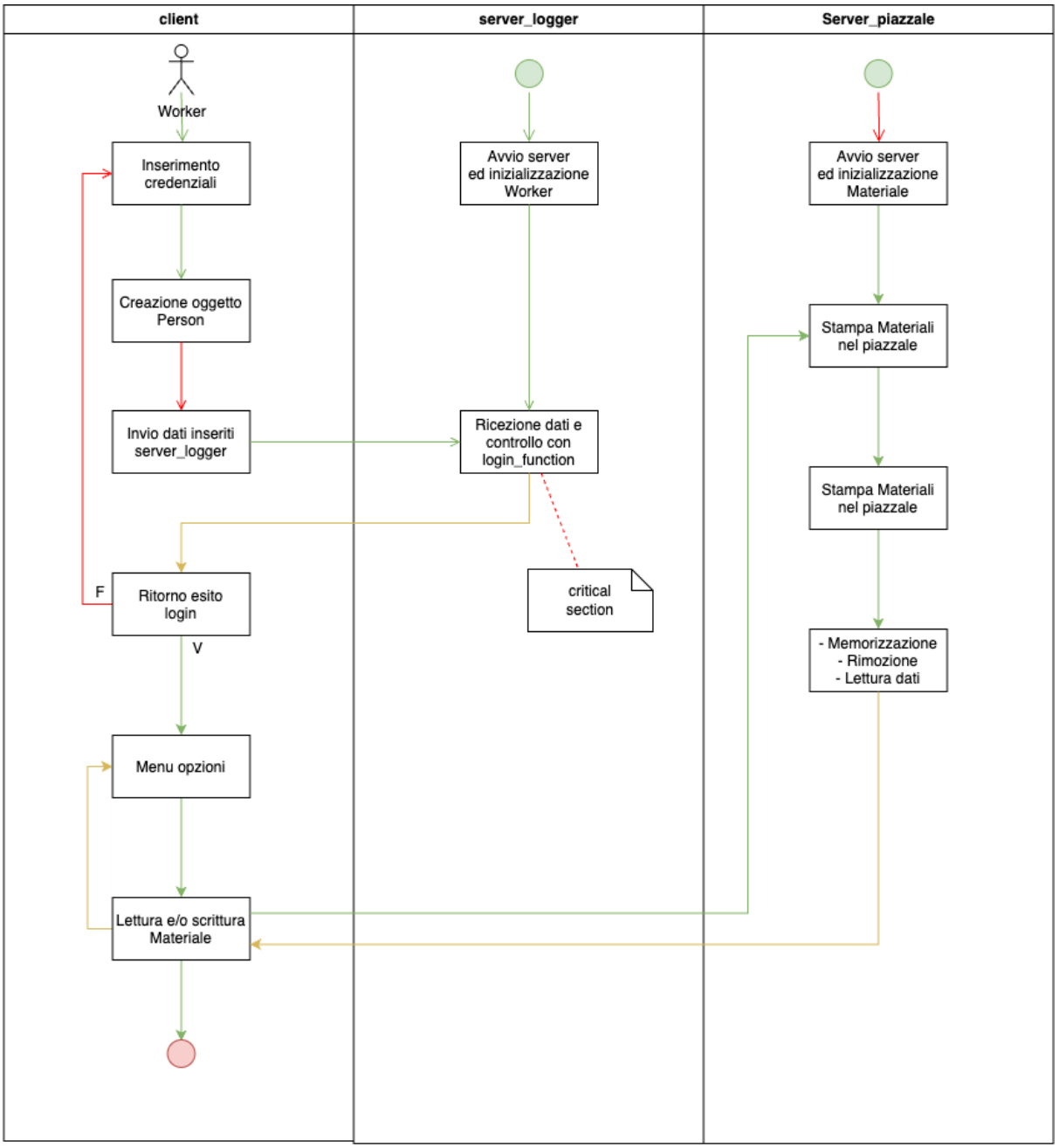
Termine	Definizione
Lavoratore/Utente	Persona che effettua manipolazioni o letture relativamente al materiale memorizzato
Materiale	Oggetto depositato nel piazzale. Contiene le informazioni inerenti a tale entità e viene acceduto e creato dagli Utenti

1.5. Referenze

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, 1988.

2.0. Descrizione generale

2.1. Funzionamento sistema



2.2. Requisiti funzionali

Il sistema si basa su una struttura client - server organizzata come segue.

2.2.1. Programma client

È uno script python denominato “client.py” che:

- Contiene l'inizializzazione dei due proxy Pyro: Piazzale e Logger.
- Gestisce il login dell'utente.
- Fornisce un menu per le interazioni con il Piazzale, consentendo all'utente di selezionare le azioni desiderate.

Per elargire quelli che per l’utente dell’azienda saranno i servizi il client utilizza i remote objects contattando i proxy Pyro per manipolarli. Rende i remote objects trasparenti all’utente.

Il client all’avvio dovrà effettuare alcuni setup:

- Individuare la location dell’identificatore dell’oggetto necessario utilizzando il Pyro Name Server (es. `"PYRONAME:zp.piazzale"`);
- Creare un oggetto speciale “Proxy” che effettua le chiamate all’oggetto remoto. Una volta ottenuto il “Proxy” il client potrà utilizzare tali oggetti remoti come se fossero oggetti locali:

```
piazzale = Proxy("PYRONAME:zp.piazzale")

logger = Proxy("PYRONAME:zp.logger")
```

2.2.2. Oggetti distribuiti

2.2.2.1. Person

Oggetto definito nel file “person.py” ed è utilizzato per immagazzinare i dati di un utente autorizzato. Viene generato per autenticare un utente ed in caso di esito positivo verrà utilizzato per registrare, lato server, le sue interazioni con il materiale del piazzale. Riassumendo quindi l’oggetto “Person” svolge le seguenti funzioni:

- Definisce l'oggetto "Person" che rappresenta un utente autorizzato.
- Fornisce metodi per eseguire le azioni consentite sull'oggetto Piazzale, utilizzando Pyro per la comunicazione.
- Utilizza il proxy Logger per controllare le credenziali dell'utente.

2.2.2.2. Material

Oggetto definito nel file “material.py” ed è utilizzato per immagazzinare e successivamente rappresentare il materiale che verrà depositato nel piazzale. In esso vengono memorizzate le informazioni relative al materiale memorizzato. I campi sono:

- **Id**: identificatore generato in fase di creazione dell’oggetto, ne permette un riconoscimento più semplice in caso di omonimi;
- **Name**: nome del materiale depositato;
- **Arrive_date**: viene registrata la data e l’orario di arrivo in maniera autonoma dal sistema;
- **Inserted_by**: contiene il nome dell’utente che ha inserito il prodotto. Ciò permette una più semplice fase di controllo qualità.

Riassumendo l’oggetto distribuito “**Material**” svolge le seguenti funzioni:

- Definisce l'oggetto "**Material**" e le relative inizializzazioni.
- Memorizza le informazioni del materiale, inclusi il nome, il nome dell'utente che lo ha inserito e la data/ora di inserimento.

2.2.3. Programmi server

2.2.3.1. Server piazzale:

Il server del piazzale, definito nel file "**server_piazzale.py**", memorizza gli oggetti remoti della classe "**Material**" che verranno acceduti dagli utenti. Il server effettuerà:

- Inizializzazione di alcuni materiali sample;
- Esporrà metodi per l'interrogazione dello stesso. Tali metodi sono:
    - **list_contents**: Per stampare tutto il materiale contenuto nel piazzale;
    - **take**: Per prelevare un materiale dal piazzale e quindi rimuoverlo da esso;
    - **store**: Per memorizzare un materiale arrivato nel piazzale;
    - **get_material_info**: Per recuperare le informazioni di uno, o più in caso di nomi uguali, materiali memorizzati.
- Infine viene definito il nome con il quale il server verrà esposto "**zp.piazzale**".

Riassumendo il "**server_piazzale**" svolge le seguenti funzioni:

- Contiene l'inizializzazione del Piazzale con alcuni oggetti.
- Fornisce i metodi per interagire con il Piazzale, che vengono solitamente chiamati dai metodi della classe Person.

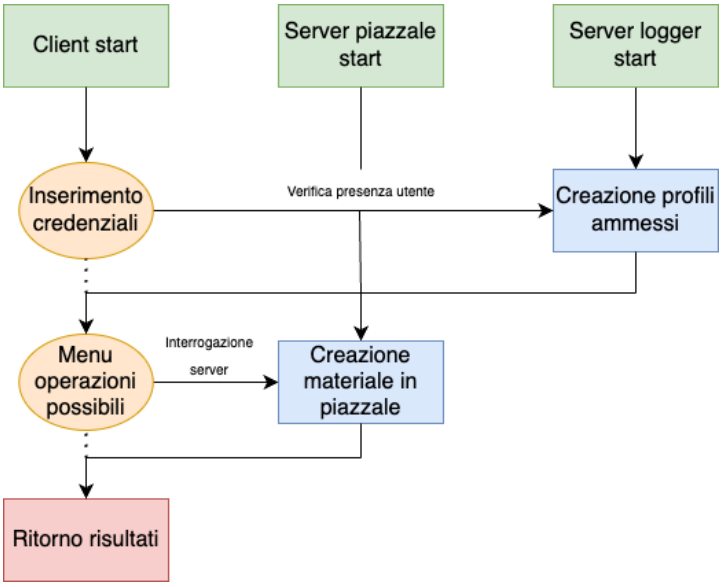
2.2.3.2. Server logger:

Il server per l'autenticazione, definito nel file "**server_logger.py**", come suggerisce il nome si occupa di memorizzare gli oggetti remoti della classe "**Person**" e di gestire l'inizializzazione degli utenti di sample e di fornire il metodo di autenticazione di questi.

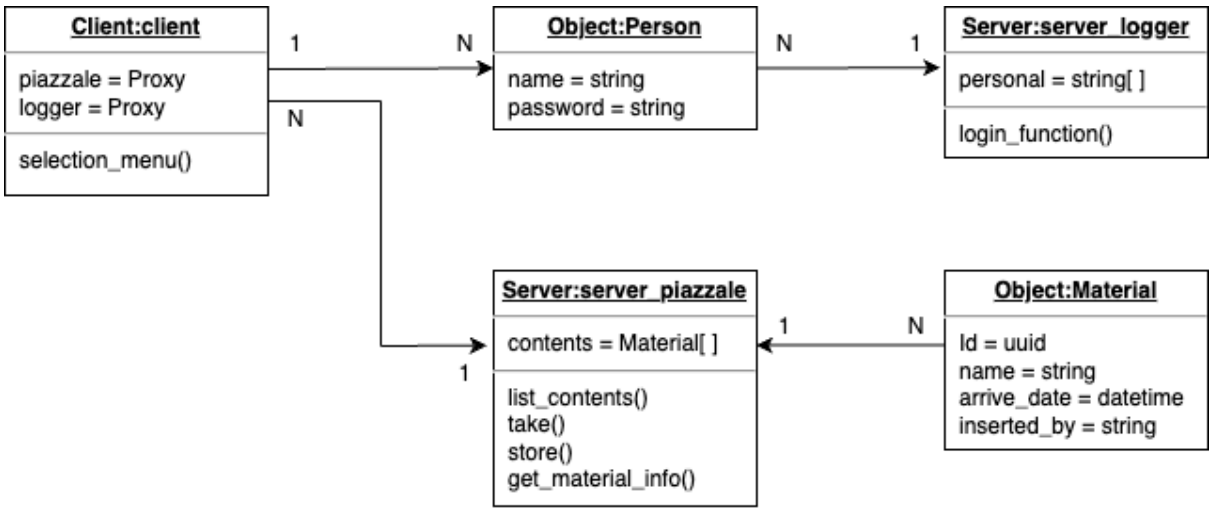
- **login_function**: prende in input il nome utente e la password. In caso di match fornisce il via libera alle funzioni di interrogazione fornite dal client;
- Definizione del nome con il quale il server verrà esposto "**zp.logger**"-

Riassumendo il "**server_logger**" svolge le seguenti funzioni:

- Contiene il codice del proxy Logger;
- Implementa una funzione per controllare le credenziali dell'utente che si vuole autenticare.



2.2.4. UML



3.0. Requisiti software

Sistema operativo MacOS con Python e Pip. Versioni raccomandate

- python 3.10 o superiori;
- pip 22.3 o superiori.

Di default Pyro utilizza il protocollo Pyro che si basa sulle pickle facility di Python per creare i suoi messaggi.

Per creare il virtual environment contenente le librerie necessarie al programma eseguire i seguenti comandi:

- *virtualenv venvzp*
- *venvzp\bin\activate*

Per installare le librerie necessarie utilizzare il file “requirements.txt” contenente l’elenco delle librerie e delle relative versioni utilizzate in fase di sviluppo mediante il comando:

- *pip3 install –r requirements.txt*

3.1. Bootstrap programma

Per eseguire il sopra descritto programma saranno necessari almeno 4 terminali:

1. Avvio del pyro name server: è un componente fondamentale che consente la registrazione e la ricerca degli oggetti remoti. Permette di:
    a. **Registrare degli oggetti remoti**: gli oggetti che vogliamo rendere accessibili tramite Pyro devono essere registrati presso il name server. Durante la registrazione, viene assegnato un nome univoco all'oggetto remoto. L'oggetto viene in sostanza associato a un identificatore unico all'interno del name server;
    b. **Pubblicare degli oggetti remoti**: una volta registrato, l'oggetto remoto viene pubblicato nel name server. questo significa che il name server tiene traccia dell'oggetto e fornisce un'interfaccia per accedervi in modo remoto:
    c. **Ricercare degli oggetti remoti**: il client effettua una richiesta al name server per ottenere un riferimento all'oggetto desiderato;

    d. **Risoluzione delle richieste**: permette di cercare l'oggetto corrispondente nella sua lista di oggetti registrati. una volta trovato, il name server restituisce al client il riferimento all'oggetto remoto, consentendo al client di chiamare i metodi sull'oggetto come se fosse locale;
    e. **Gestione delle disconnessioni**: per rispondere alla disconnessione di oggetti remoti;
    f. Il comando da eseguire nel primo terminale è il seguente: *python3 -m Pyro5.nameserver.*
2. Avvio del Proxy "**server_piazzale**" nel secondo terminale e del Proxy "**server_logger**" nel terzo. I proxy Pyro semplificano l'accesso agli oggetti remoti, astraggono la comunicazione di rete (pickle), gestiscono la trasparenza di località e supportano la distribuzione degli oggetti nelle applicazioni distribuite basate su Pyro. I comandi per avviare i due server sono:
    a. *python3 src/server/server_piazzale.py*
    b. *python3 src/server/server_logger.py*
3. Avvio dello script client per accedere al programma descritto in questo report. Il comando per il quarto terminale è:
    a. *python3 src/client.py*

4.0. Riferimenti

GitHub: github.com/TinoPisi
Contatti: 218115@studenti.unimore.it