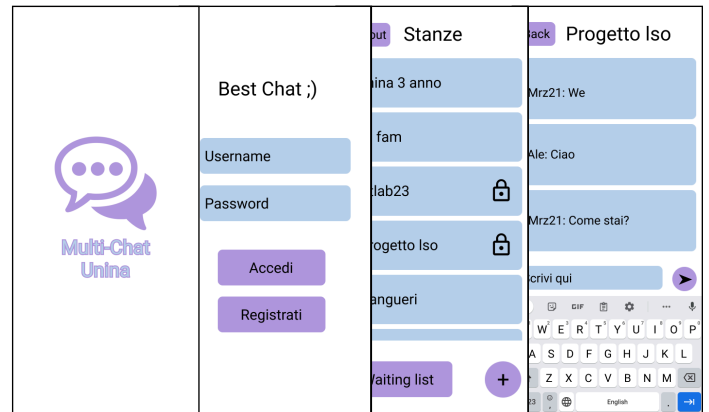


Marzia Pirozzi N86003545
Alessia Ascolese N86003704
Progetto per "Laboratorio Di Sistemi Operativi"
prof. Alessandra Rossi



Multi-Chat Unina

Android Client/C Server structure

Requisiti Richiesti

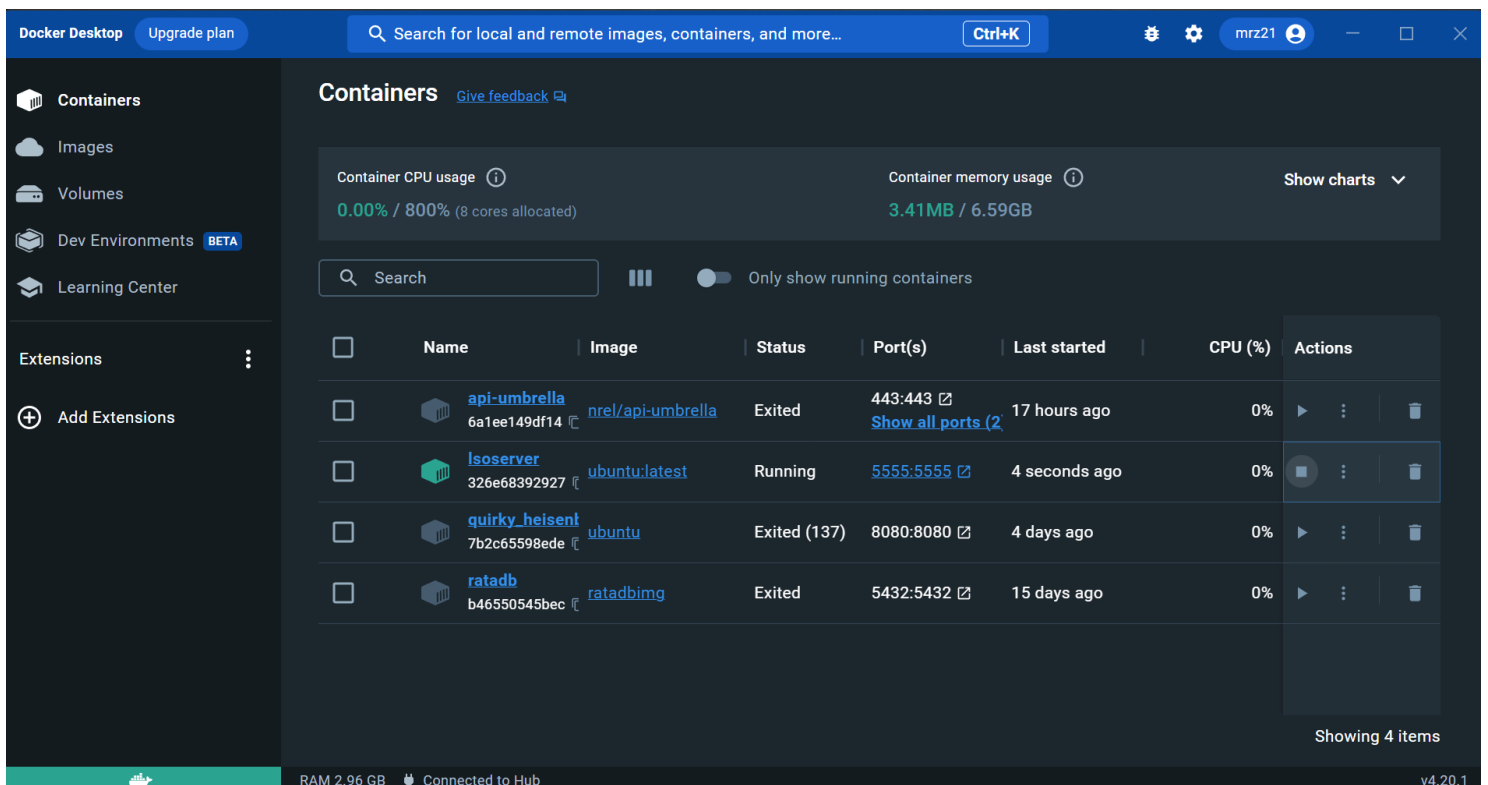
- ✓ N utenti (client) possono accedere ad un sistema di chat
- ✓ Gli utenti devono potersi registrare e loggare al server
- ✓ Registrazione può essere sviluppata tramite database sql o file
- ✓ Gli utenti possono creare stanze e possono chattare solo con gli utenti presenti nella stanza
- ✓ Gli utenti possono accedere a stanze già create
- ✓ Il creatore della stanza deve poter accettare un utente che si vuole aggiungere alla stanza

Funzionalità Extra

- ✓ Applicazione Multidispositivo
- ✓ Dati mantenuti su database POSTGRESS
- ✓ Servere e Database su Cloud (AWS)

Scelte Backend

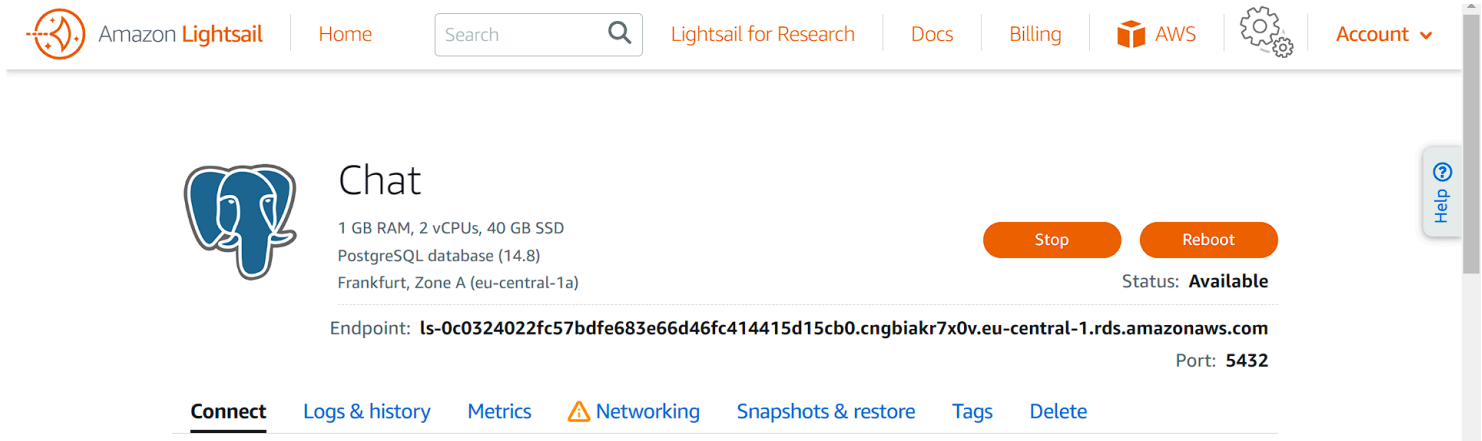
Siamo partiti con l'utilizzare un container Docker, con immagine ubuntu, poiché esso ci consentiva di effettuare delle operazioni in modo più efficiente dato l'utilizzo successivo di Android Studio presente su Windows.



Una volta avviato, abbiamo fatto l'update e abbiamo installato da terminale tutti i servizi necessari sulla nostra macchina: **nano**, **compilatore C**, **libreria postgres (libpq-dev)**.

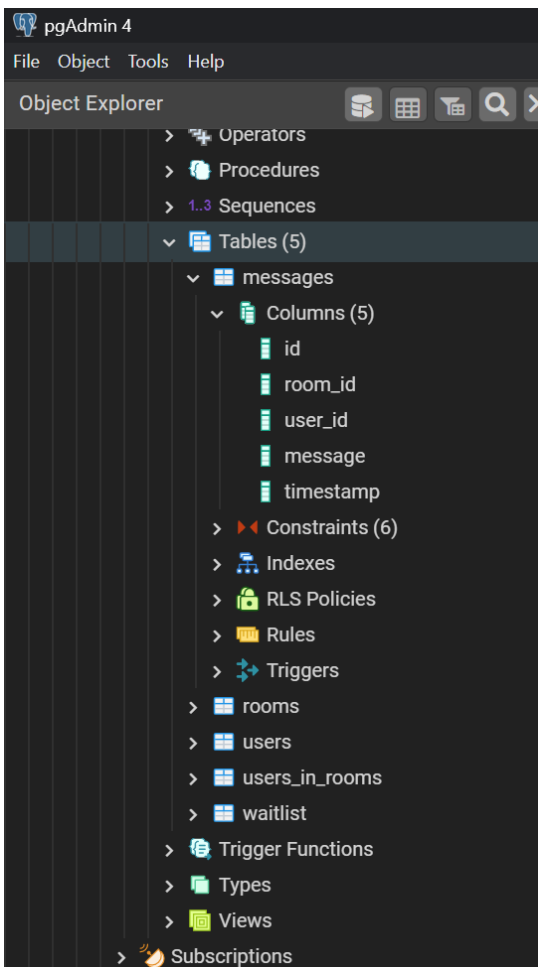
Successivamente una volta testato il funzionamento in locale del server raggiungibile quando connessi alla stessa rete, abbiamo deciso di spostare il server su cloud attraverso il servizio **AWS EC2** aprendo le porte in entrata e in uscita per comunicare con il database e con il client android studio da qualsiasi indirizzo IP.

Database



The screenshot shows the Amazon Lightsail console for a PostgreSQL database instance named 'Chat'. The instance is located in Frankfurt, Zone A (eu-central-1a) and has 1 GB RAM, 2 vCPUs, and 40 GB SSD. It is a PostgreSQL database (14.8). The status is 'Available'. The endpoint is `ls-0c0324022fc57bdf683e66d46fc414415d15cb0.cngbiakr7x0v.eu-central-1.rds.amazonaws.com` and the port is 5432. The console includes navigation links for Connect, Logs & history, Metrics, Networking, Snapshots & restore, Tags, and Delete. There are also buttons for Stop and Reboot.

Abbiamo deciso di usare come database uno di tipo relazionale : **PostgreSQL**. Abbiamo analizzato la traccia e individuato le seguenti entità:



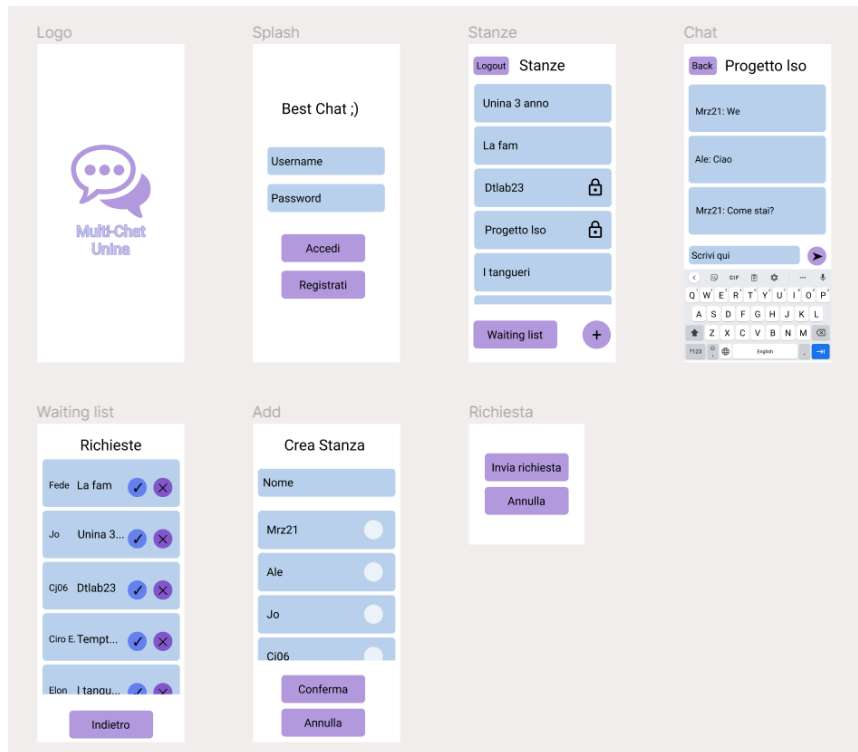
- Utenti
- Stanze
- Messaggi
- Wait-List
- Relazione: Users_In_Rooms

Abbiamo aggiunto dei constraint quali :

- Username degli utenti (Unique)
- Nome delle stanze (Unique)
- vari Primary keys e Foreign Keys dove necessario

Scelte Frontend

Per il FRONTEND abbiamo scelto il classico Android Studio



Tramite Figma abbiamo effettuato uno studio grafico pensando a quali fossero le funzionalità principali che la chat dovesse possedere.

Abbiamo poi controllato l'usabilità in particolare usando strumenti come Adobe Colors per testare il contrasto e la daltonia

Struttura Backend

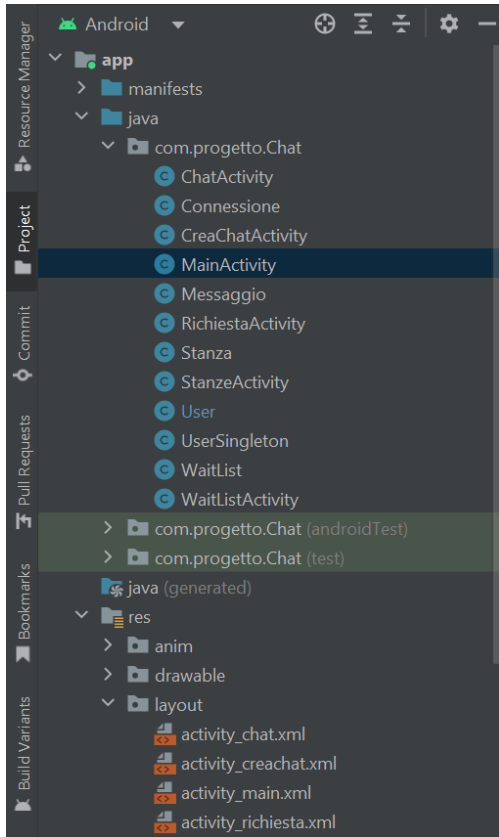
Abbiamo scritto il server in C aprendo la porta 5555, separando il main dalle funzioni che abbiamo inserito in un file ".h" rendendo il nostro codice modulare.

```
aws Servizi Cerca [Alt+S]

ubuntu@ip-172-31-19-165:~$ ls
mylibrary.h  server  server.c
ubuntu@ip-172-31-19-165:~$ gcc -o server server.c -lpq
ubuntu@ip-172-31-19-165:~$ ./server

Connessione al db riuscita
Server in ascolto sulla porta 5555...
```

Struttura Frontend



Abbiamo diviso la business logic dalla presentation logic creando delle classi:

- **Model** che rappresentano le entità che abbiamo trovato nella analisi iniziale
- **View** che rappresentano le activity con cui interagiranno gli utenti
- **Controller** che rappresenta la connessione con il server

In particolare abbiamo utilizzato come controller la classe UserSingleton in quanto abbiamo deciso di adottare questo pattern perché dal momento in cui viene lanciata l'applicazione un unico utente viene stanziato ed esiste fino alla chiusura di esso. Quindi UserSingleton contiene tutte le funzioni di comunicazione col server

Funzionamento:

Il server per poter gestire più utenti riceve la richiesta di una connessione da parte del client, viene quindi eseguita una fork e il figlio chiama la funzione handleClient che riceve una stringa strutturata nel seguente modo: **"n%argomento1%argomento2..."**.

- n rappresenta il numero dell'operazione da effettuare
- i successivi sono gli argomenti da passare alle funzioni

Il server divide il messaggio in un array di stringhe e passa quest'ultimo come argomento alla funzione **switchCase** che in base al primo argomento esegue la funzione associata a quel numero. Dopo aver eseguito la funzione viene inviato al client un messaggio contenente o la risposta del database codificata allo stesso modo o un messaggio di successo/errore.

Test effettuati

- Gli username, i nomi delle stanze e i messaggi che contengono i caratteri '%' e '@ ' possono dare effetti indesiderati in quanto sono proprio i caratteri separatori dei messaggi tra client e server
- L'applicazione risulta essere poco veloce in quanto ogni volta che si avvia una activity bisogna aspettare che il server interroghi il database riceva risposta e invii il messaggio finale al client
- Al momento abbiamo provato con un massimo di tre utenti connessi senza riscontrare problemi.
- L'applicazione potrebbe contenere eccezioni non gestite. Ad esempio quando proviamo a registrarci o fare login mentre il server è spento l'app si chiuderà.

Walkthrough

Splash

Best Chat ;)

Username

Password

Accedi

Registrati

Schermata Home

Per poter utilizzare la nostra app , è necessario inserire le credenziali. Se non si ha già un account si preme sul pulsante **Registrati** per poterne creare uno altrimenti si preme il pulsante **Accedi**

Schermata Stanze

una volta entrati abbiamo una lista di tutte le stanze create da tutti gli utenti. Premendo su una stanza in cui non abbiamo accesso verrà aperta una schermata che ci permetterà di inviare una richiesta di accesso, altrimenti si aprirà la chat.

Da questa schermata abbiamo anche la possibilità di fare logout, di accedere alla **Waiting list** o alla schermata per creare una nuova stanza.

Stanze

Logout

Stanze

Unina 3 anno

La fam

Dtlab23



Progetto Iso



I tangueri

Waiting list



Add

Crea Stanza

Nome

Mrz21 ☐

Ale ☐

Jo ☐

Ci06 ☐

Conferma

Annulla

Creare una stanza

Se l'utente vuole creare una stanza può farlo cliccando sul pulsante "+".

Basta inserire un nome e selezionare gli utenti che vogliamo aggiungere alla stanza. Da questo elenco manca il nome dell'utente che sta creando la stanza perché verrà aggiunto automaticamente. Se non viene selezionato nessun utente verrà creata una stanza a cui solo l'utente ha l'accesso.

Gestione Richieste

Un utente oltre a inviare richieste a stanze in cui non ha accesso, può anche gestire le richieste a stanze di cui lui è amministratore. Può accettarle o rifiutarle e la richiesta in entrambi i casi sparirà dall'elenco.

In caso di accettazione della richiesta l'utente viene aggiunto alla stanza.

Richiesta

Invia richiesta

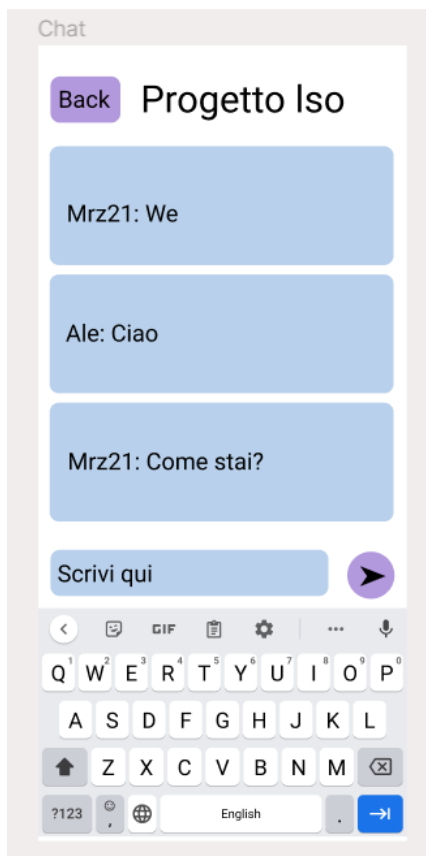
Annulla

Waiting list

Richieste

Fede	La fam	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Jo	Unina 3...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Cj06	Dtlab23	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Ciro E. Tempt...		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Elon	I tanqu...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Indietro



La Chat

La chat corrisponde ad una lista di messaggi ordinati in ordine cronologico. Quando si scrive un messaggio questo viene aggiunto alla lista, inoltre quando si preme sul pulsante invia il text input viene resettato automaticamente e viene anche abbassata la tastiera che era comparsa nello scrivere il messaggio.



Multi-Chat
Unina

Grazie per l'attenzione

Ascolese Alessia, Pirozzi Marzia