

Beccaccino / Marafone

Elaborato Sistemi Distribuiti – AA 2021/2022

Alpi Davide
Parrinello Angelo
Penazzi Paolo

Outline

Il progetto si pone come obiettivo lo sviluppo di una versione digitale multigiocatore del celebre gioco di carte "Beccaccino"

Deliverables

01

Server che gestirà le partite di Beccaccino

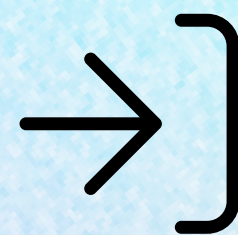
02

Applicativo client mobile (Android) per giocare

Funzionalità Principali



Creazione di una lobby

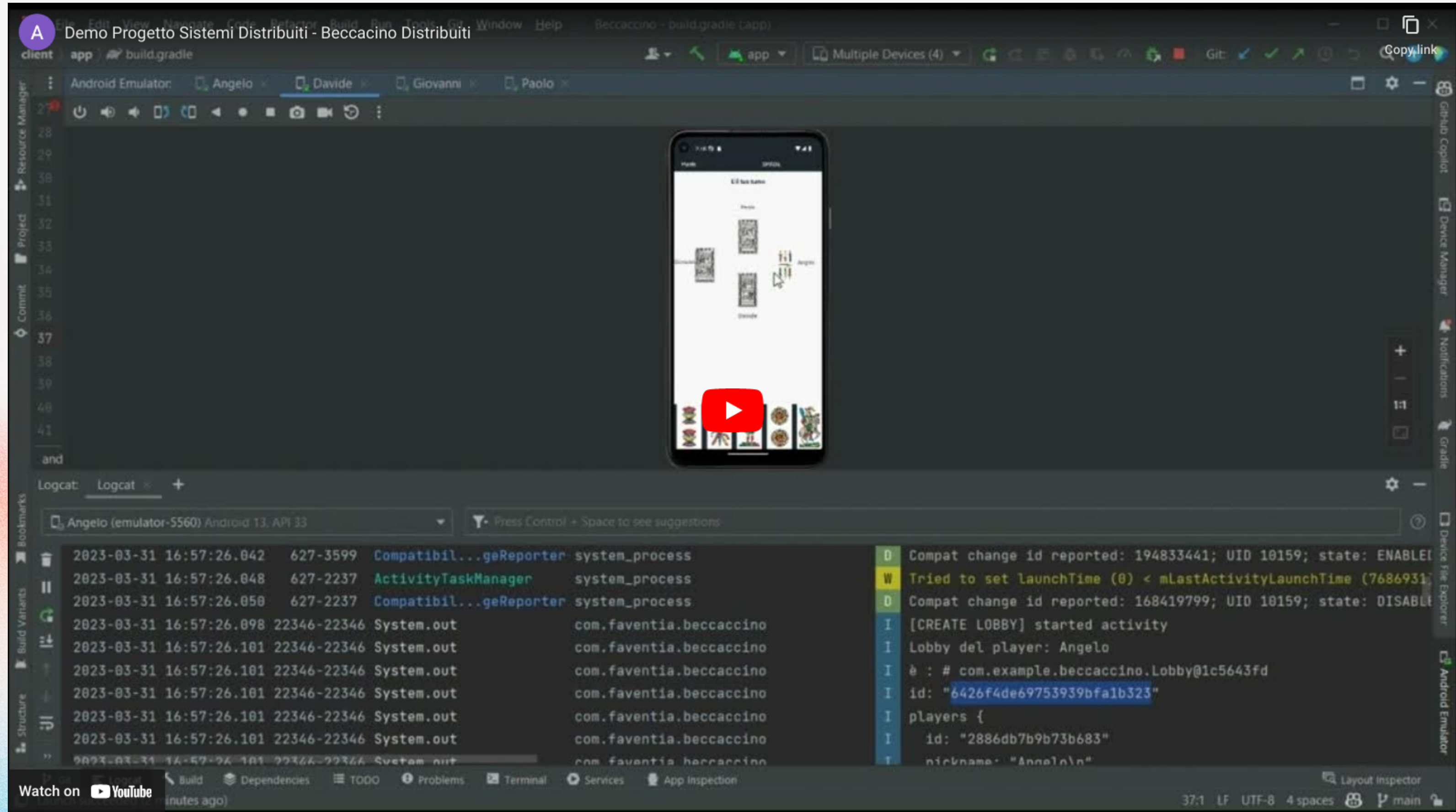


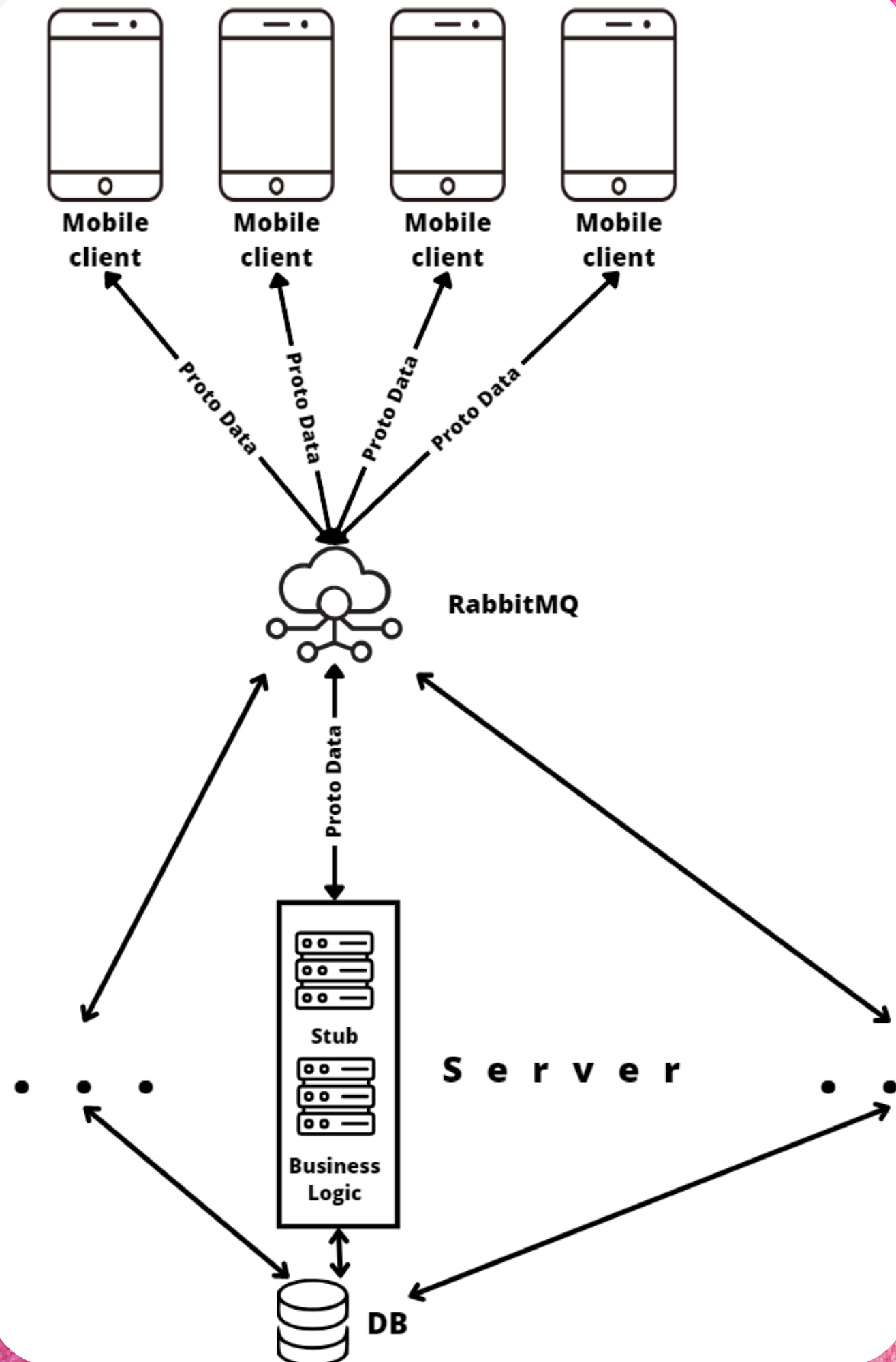
Unione ad una lobby



Avvio e svolgimento della partita

Demo





Architettura

Attori principali

- Client: fornisce la visualizzazione del gioco lato utente
- Broker: middleware in grado di distribuire i messaggi tra i componenti del sistema
- Server: implementa la logica del gioco

Scalabilità

- Il server gestisce le request dei client e non mantiene uno stato, quindi più server possono essere deployati per gestire picchi di utenza.
- MongoDB scala tramite sharding

Queues Management

Lobby

- Messaggi di richiesta inviati su coda unica **todoQueueLobbies**
- Messaggi di risposta inviati su coda univoca per giocatore **resultsQueueLobbies + playerdID**
- **N.B.** Un player non può essere presente in più lobby/partite

Game

- Messaggi di richiesta inviati su coda unica **todoQueueGames**
- Messaggi di risposta inviati su coda univoca per giocatore **resultsQueueLobbies + gameID + playerdID**
- Un attore Guardiano si occuperà di rimuovere la partita e i player nel caso in cui uno di essi sia inattivo

Handling Unexpected Behavior

Gestione di casistiche inattese e
possibili failures

- 1** Un giocatore disconnesso (per qualsiasi motivo) può riconnettersi e recuperare i dati della partita.
- 2** Un giocatore inattivo renderebbe impossibile il proseguimento della partita: è stato implementato un meccanismo per far fronte a tale comportamento.
- 3** Se il creatore di una lobby esce, la lobby viene eliminata. Non esiste un meccanismo di eliminazione automatica delle lobby come per i game, ma ogni giocatore può essere in una sola lobby per volta.
- 4** Tutti i controlli sul corretto svolgimento della partita sono effettuati sia lato client che lato server.
- 5** Per aumentare la resilienza in caso di errore, i servizi lato server sono impostati per riavviarsi in automatico. In caso di fallimento lato server (durante una partita), lo stato di gioco rimane comunque salvato nel database in modo da poterlo ricostruire ad un successivo avvio del server.

Learning Agenda

Che cosa abbiamo
imparato

- Familiarità con la logica distribuita e gestione di tutti gli aspetti che ne derivano.
- Comprensione del protocollo AMQP e del message broker RabbitMQ.
- Gestione di database documentali (MongoDB).
- Deployment e build tramite Docker compose e Gradle.
- De/serializzazione con tecnologie standard (Protobuf).

Grazie dell'attenzione!

Siamo aperti a domande e ad
approfondimenti



Angelo Parrinello
Software Engineer



Paolo Penazzi
Software Engineer



Davide Alpi
Software Engineer