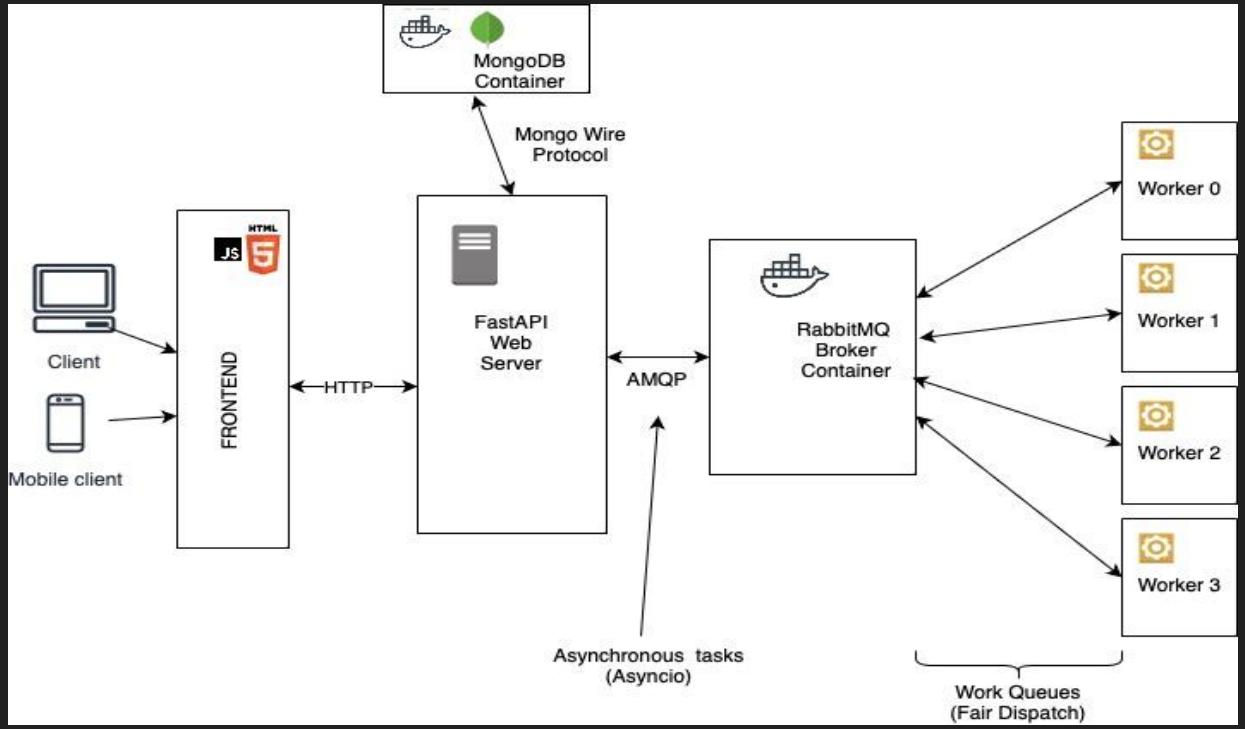
Computação Distribuída

Distributed Music Editor

Bernardo Falé 93331
André Silva 98651

Architecture Diagram



Protocol

Comunicação Server -> Worker

```
data = {
    "job_id": job_id,
    "song_id": str(music_id),
    "block_id": counter,
    "body": base64.b64encode(chunk).decode('utf-8'),
    "tracks": tracks.tracks,
    "group_id" : rand_hash
}
```

Comunicação Worker -> Server

```
data = {
    "job_id": job_id,
    "song_id": str(song_id),
    "block_id": block_id,
    "body": base64.b64encode(chunk).decode('utf-8'),
    "track_id": index,
    "processing_time": processing_time,
    "group_id": group_id
}
```

Timeline

1. The client makes a request and submits a music file
2. The server stores that data into a GridFS database
3. The client makes a request for a song to be processed
4. The request is processed and the song is sliced and distributed among 4 workers
5. The messages are exchanged through rabbitMQ with the protocol defined in the last slide
6. After processing, the workers send back the chunks to the server
7. Upon reception, the server stores the decoded chunk data of each track
8. The client gets the processing state and downloads the merged file

Tests and Results

1. Testing with test.mp3
    a. Optimal chunk size?
    b. How much time gained by distributing work?

2. Testing with music.mp3
    a. Optimal chunk size?
    b. How much time gained by distributing work?