QuizTune

Quizlet style App for recognizing music

Challenge Task FS 2026 — Distributed Systems

Severin Nauer

OST — Spring Semester 2026

How It Works

The user flow:

- Log in with Spotify (OAuth PKCE flow)
- Create a **learning environment** from Spotify playlists
- Start a quiz — a random song plays via Spotify Web Playback SDK
- Identify the **song title**, **artist**, and **release year**
- Score based on accuracy

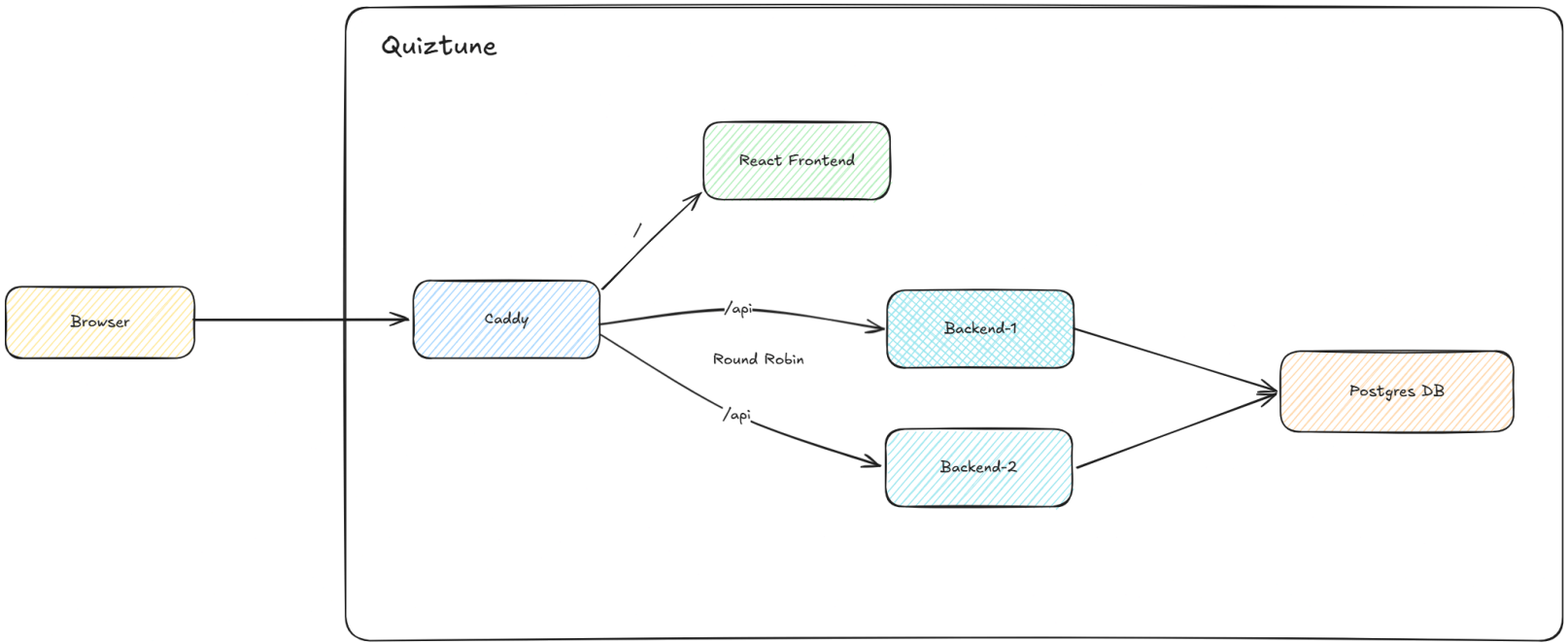
Key design choice:

- Playback happens in-browser via the Spotify SDK. The backend handles auth, data storage, and scoring.

Tech Stack

Frontend	React 19 · Vite 7 · TypeScript · Tailwind v4 · shadcn/ui
Backend	.NET 10 Minimal API · EF Core 10 · Npgsql
Database	PostgreSQL 18 Alpine · named volume
Load Balancer	Caddy · round-robin · health checks 5s
Auth	Spotify Login · JWT HS256 · 15min · refresh rotation · HttpOnly
Infrastructure	Docker Compose · 5 containers · `docker compose up`

Architecture



Load Test Setup

Tool: “hey” — HTTP load generator

Endpoint: `GET /api/environments`

- Queries `LearningEnvironments` for the authenticated user
- JWT Bearer token in Authorization header

Environment: Docker on local machine, all 5 containers on same host

Baseline

3,990 req/s · 12ms avg · 0% errors

Test: 5,000 requests, 50 concurrent (moderate load)

Metric	Req/s	Avg	p50	p95	p99
Value	3,990	12ms	11ms	22ms	37ms

Latency distribution:

“p10” “7ms”
“p25” “9ms”
“p50” “11ms”
“p75” “14ms”
“p90” “18ms”
“p95” “22ms”
“p99” “37ms”

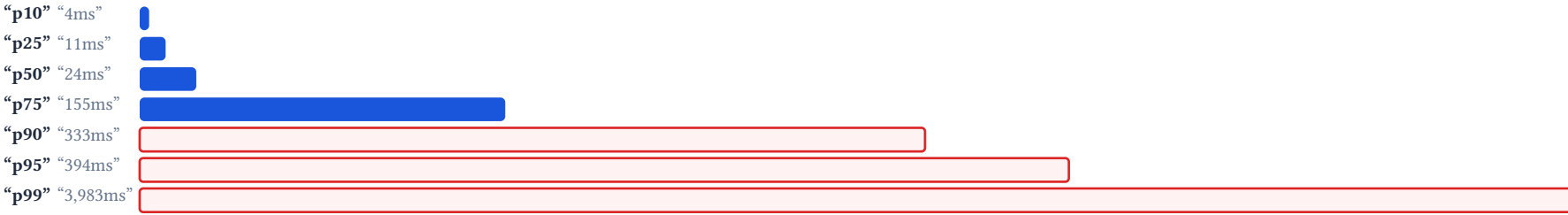
Breaking Point — Under Stress

System collapses under load. 20%+ error rate at 100 concurrent.

Test: 10,000 requests, 100 concurrent

Metric	Req/s	Avg	p50	p95	p99
Value	188	135ms	24ms	394ms	3.98s

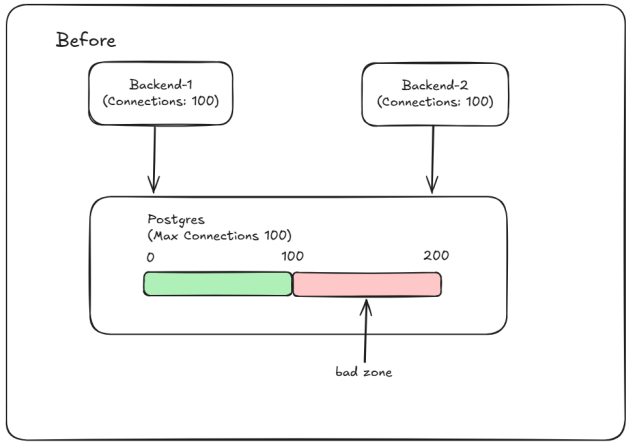
Latency distribution:

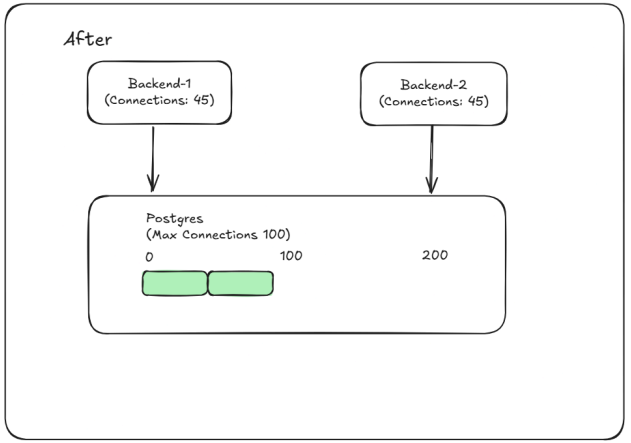


Error breakdown:

500	503	Timeout	Total
464	1,593	183	2,240 (~22%)

Root Cause + The Fix





The Problem
MaxPoolSize=100 × 2 = 200
PostgreSQL limit: 100
→ “too many clients” ✗

The Fix
MaxPoolSize=45 × 2 = 90
Under PostgreSQL’s 100
→ safe ✓

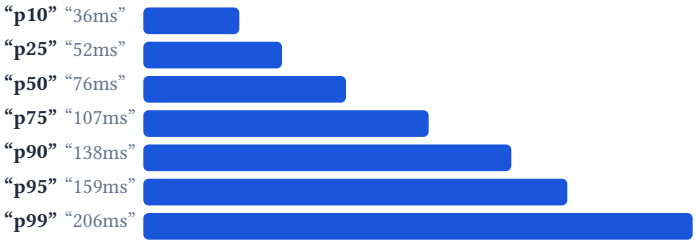
Results — After the Fix

2,343 req/s · 83ms avg · 0% errors

Test: 20,000 requests, 200 concurrent

Metric	Req/s	Avg	p50	p95	p99
Value	2,343	83ms	76ms	159ms	206ms

Latency distribution:



All 20,000 requests returned `[200]` — zero failures.
p10-p99 all under 210ms. Clean, predictable latency. No tail explosion.

Key Takeaways

Spotify API	“Restrictive for non-commercial apps. Rate limits and OAuth add overhead.”
Connection Pools	“Multiple backends must cap pool size to match the database limit. Defaults assume a single instance.”
Docker Overhead	“`docker compose up` attached adds TTY and logging overhead. Detach with `-d` for load tests.”

Next steps

PgBouncer	“Centralized connection pool proxy between backends and PostgreSQL. Eliminates per-instance pool coordination.”
Auto Scaling	“Dynamic backend scaling with service discovery. Add or remove instances without manual config changes.”
Audio Service	“Replace Spotify with a more accessible audio provider. Less restrictive, lower barrier for users.”
Quiz Modes	“More learning and quizzing modes beyond timed recognition. Adaptable to different skill levels.”