Building a real-time music service in Erlang

Ali Sabil

ali.sabil@soundrop.com

SOUNDROP

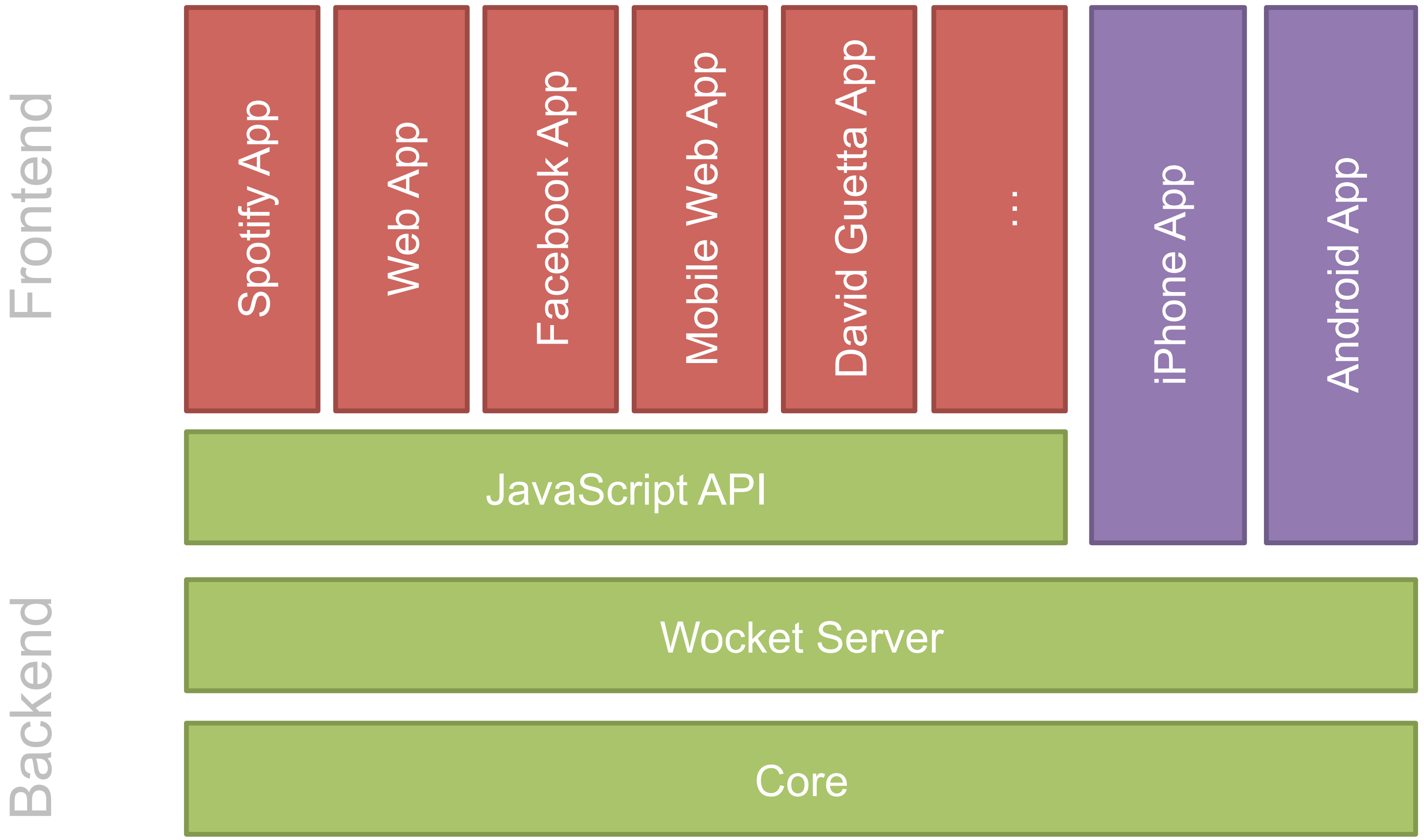
the music service

LET’S TALK ABOUT SOUNDROP

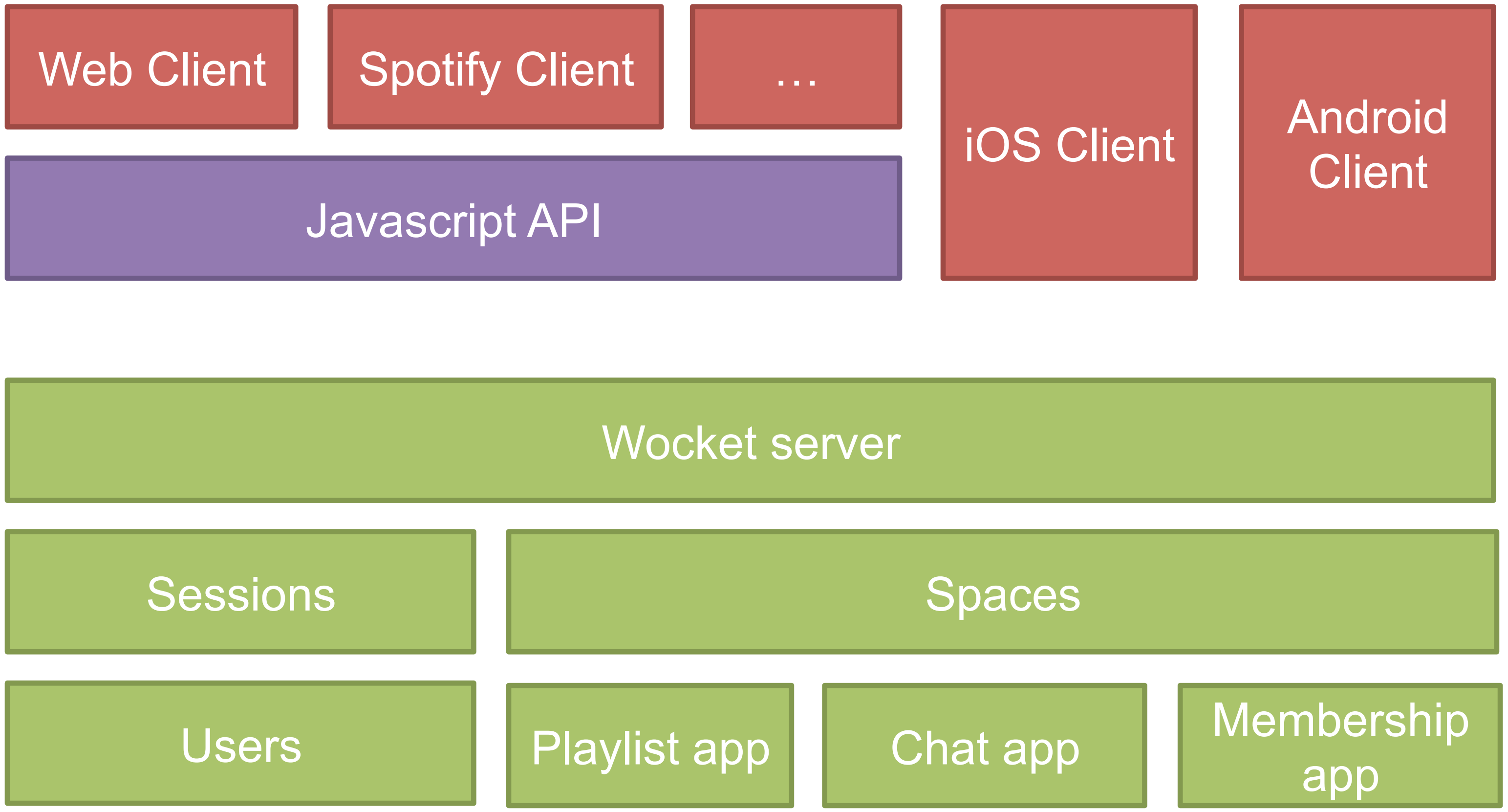
SOUNDROP

Demo

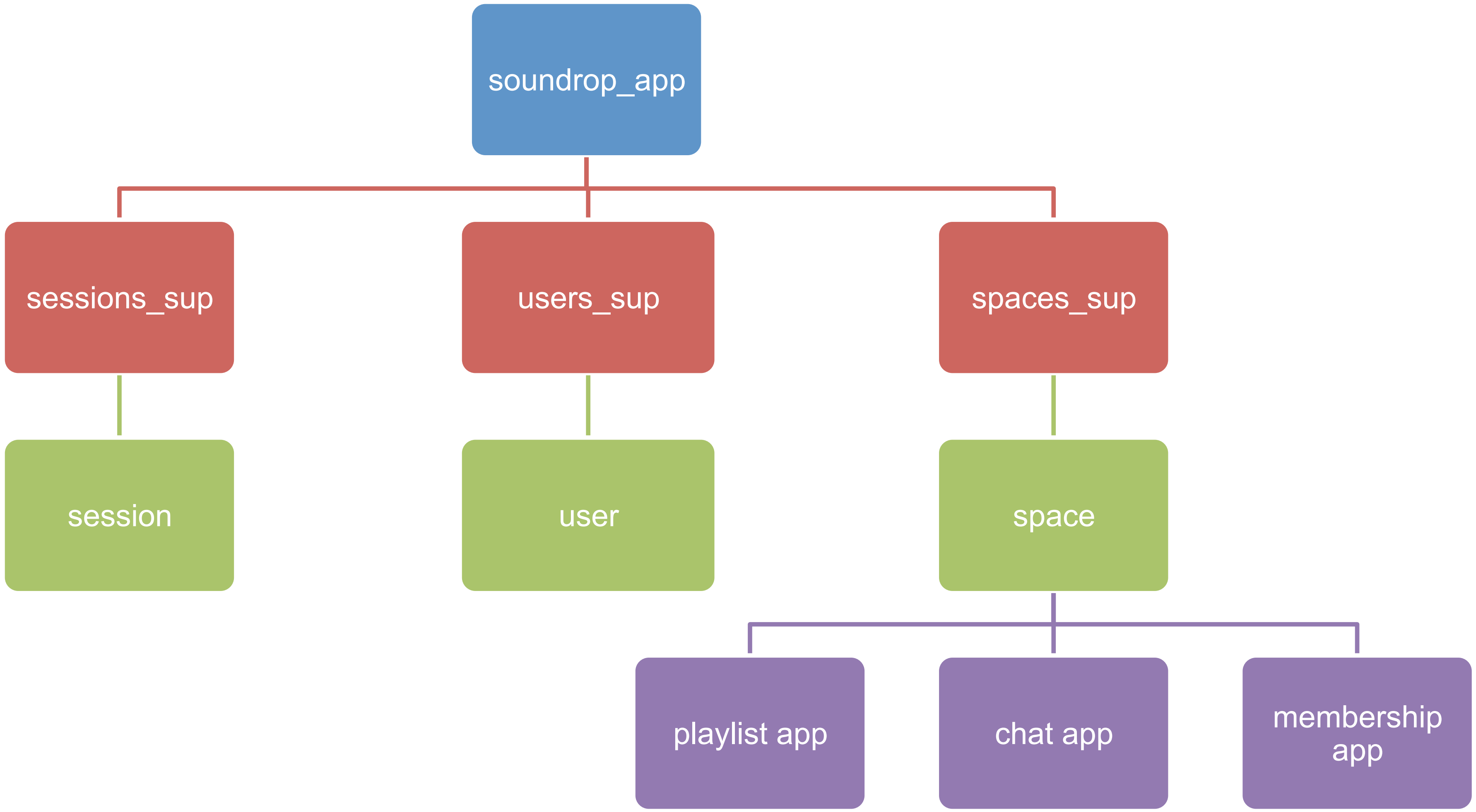
Overview


Frontend
Backend
Spotify App
Web App
Facebook App
Mobile Web App
David Guetta App
…
iPhone App
Android App
JavaScript API
Wocket Server
Core

Overview


Web Client
Spotify Client
…
iOS Client
Android Client
Javascript API
Wocket server
Sessions
Spaces
Users
Playlist app
Chat app
Membership app

Overview



SOUNDROP

the language

LET'S TALK ABOUT ERLANG

Our Journey to Erlang

- Python
- Node.js
- Python + Redis
- Erlang

Why Erlang?

Why Erlang?

- Concurrency

Why Erlang?

- Concurrency
- Fault-tolerance

Why Erlang?

- Concurrency
- Fault-tolerance
- Hot code patching

Why Erlang?

- Concurrency
- Fault-tolerance
- Hot code patching
- Soft real-time

Why Erlang?

- Concurrency
- Fault-tolerance
- Hot code patching
- Soft real-time
- We love it ☺

and the mistakes we made

LET’S TALK ABOUT WHAT WE LEARNED

SOUNDROP

No magic

- Scaling Soundrop is hard
- Erlang doesn’t scale the system for you

VM can crash

- OOM crashes on 32 bits arch
- Overflowing message queues

Erlang as a Linux service is hard

- No standard integration out of the box
- No handling of standard signals
- No .pid file creation

Stuff can hang

- usage run_erl, to_erl and SSH somehow caused shell to hang
- had I/O issues when using sendfile in R15 causing I/O to hang
- HTTPC can sometimes hang forever

Heartbeat timeouts

- We had peaks of inter-node traffic, causing heartbeat timeouts
- Caused split brain issues in gproc in distributed mode

Other hiccups

- Some third party libraries are not really great
- Lack of libraries sometimes, but it's easy to roll your own
- Lack of web based monitoring tools
- You need to rewire your brain to work with Erlang

SOUNDROP

The conclusion

LET’S WRAP UP

SOUNDROP