**Spotify**

Spotify is a Swedish audio streaming and media services provider founded on 23 April 2006 by Daniel Ek and Martin Lorentzon. It is one of the largest music streaming service providers, with over 590 million monthly active users, including 226 million paying subscribers, as of September 2023.

**Architecture**

To keep the UI platform agnostic, Spotify engineers built TypeScript Platform APIs that would abstract the different sources of data and different playback stacks, as well as provide helpful information to the user interface about what functionality was available to it.

The final architecture looks like **a layer of Platform APIs** that expose the underlying Spotify ecosystem to clients, with a React-based user interface and the Platform APIs exposed via React Hooks. Thus, the new UI can run on the web, and it can run in our Desktop container, and never know, or care if the data is coming from the C++ stack or web infrastructure.

A screenshot of a computer

Description automatically generated

*The latest architecture of Web Player (left) and Desktop (right) clients. The UI is built as a React application that reaches the backend through the GraphQL and Web API services, and in some cases achieves this through the native Desktop APIs due to their increased performance and capabilities.*

The Spotify backend infrastructure is built up of several layers of hardware and software, ranging from physical machines to messaging and storage solutions.

Any architecture that needs to handle the volume of users that Spotify has needs to partition the problem. The Spotify architecture partitions the problem in several different ways.

One of them is **partitioning by features**. A slightly oversimplified description is that the physical screen area of all the pages and views in our clients is owned by some squad. All the features in the Spotify clients belong to a specific squad. The squad is responsible for that feature across all platforms –from how it appears on an iOS device or a browser via the real-time requests handled by the Spotify backend to the batch-oriented data crunching that takes place in our Hadoop cluster to power features like recommendations, radio, and search.

**Language**

According to Wikipedia, the primary language behind Spotify's development is **Python**. But according to Andreas Blixt who is a 5-year employee of Spotify, the main language used is **C++** and not Python.

Other significant languages used are **Java and C**. Spotify uses the Chromium Embedded Framework (CEF) to display a web interface consisting of HTML/CSS/JavaScript within the desktop application. **JavaScript** is used across the Spotify desktop client, wherever UI is concerned. C++ is used for functionality beneath the UI, with JavaScript sitting on top of it. The languages are connected by an interface called a ‘bridge’.

On the contrary to the desktop application, which is written mostly in C++ with JavaScript utilized for the web-related components, its web version has a little bit of everything. **HTML 5, CSS, and JavaScript are a common combination**. There are no usual frameworks used, but a lot of Java and Ruby on Rails have been implemented for certain functional components. Apart from the web components, there are apps made for different platforms, and different languages are used accordingly for app development.

**Databases**

Overall, Spotify uses a wide variety of technologies to provide its music streaming services. Its microservices architecture, use of containerization, and focus on scalability and availability have helped it become one of the most popular music streaming platforms in the world.

* **Cassandra**: A distributed NoSQL database that provides high availability and scalability. Spotify uses Cassandra to store user data, playlists, metadata, and other information.
* **PostgreSQL**: A relational database that supports SQL queries and transactions. Spotify uses PostgreSQL to store analytics data, such as user behavior and preferences.
* **Google Cloud Storage**: A cloud-based object storage service that offers durability and performance. Spotify uses Google Cloud Storage to store audio files and other media assets.
* **Memcached**: A distributed memory caching system that improves the speed and scalability of web applications. Spotify uses Memcached to cache frequently accessed data, such as track metadata and user sessions.
* **Kafka**: A distributed streaming platform that enables data ingestion, processing, and delivery. Spotify uses Kafka to handle real-time events, such as user actions, logging, and monitoring.

Open-source Apache software platforms Kafka, Storm, Crunch, and Cassandra make up the foundation of Spotify. These software systems seamlessly work together to collect and analyze data in real time and then store it in user profiles as personalized suggestions according to the user’s preference.

A diagram of a pipeline

Description automatically generated

*A diagram of data flow in Spotify*[*(Spotify)*](https://engineering.atspotify.com/2015/01/09/personalization-at-spotify-using-cassandra/#:~:text=We%20help%20our%20users%20discover,and%20distastes%20in%20different%20contexts.)*.*

Spotify utilizes Apache Cassandra in its infrastructure to serve data behind a personalization algorithm, which recommends songs to its users. Spotify’s data infrastructure is built on open-source Apache software platforms like Kafka, Storm, and Cassandra. These systems work together to collect and analyze data in real time and store it in user profiles as personalized suggestions.

**Tricks for better performance**

There are many ways to optimize the Spotify app for better performance, such as sound quality, battery life, and data usage.

* **Adjust the streaming and download quality.** You can choose from Low (24 kbps), Medium (96 kbps), High (160 kbps), or Very High (320 kbps). Higher quality means better sound, but also more data and storage consumption.
* **Enable data saver mode.** This will automatically lower the streaming quality when you are using mobile data, saving you bandwidth and battery.
* **Use Spotify Connect to control playback on other devices.** This feature lets you play music and control it from any of your devices, such as your laptop, speaker, or smart TV.
* **Use advanced search operators to find exactly what you want.** You can use keywords like year, genre, label, and more to narrow down your search results.
* **Try out the crossfade feature to blend songs smoothly.** This will make the transitions between tracks more seamless and enjoyable.
* **Start using a private session to listen to music anonymously.** This will prevent Spotify from sharing your listening activity with your followers or on social media.
* **Add local files to Spotify.** You can import your music files from your device to Spotify and listen to them alongside your streaming tracks.