

Logotipo

Descripción generada automáticamente

**TECHNICAL PROPOSE Document**

**Technical propose Document**

**Introduction**

**Purpose**

M2S implements a global platform whose number of users and processing load can be very high. Therefore, a technological proposal capable of offering high performance and availability is made, while having the flexibility to scale the necessary resources dynamically.

**Architecture and technological proposal.**

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 1 Diagram M2S Infrastructure Proposal

The implementation of a **microservices-based architecture** is proposed to enhance the scalability and flexibility of the system in a granular manner. This architectural approach allows for the development and deployment of independent services that can scale individually, enabling better resource management and faster feature development. The architecture is organized into several key components:

**Frontend**

* The user interface will be designed as a **responsive web application** to ensure a seamless experience across devices.
* **Framework Choice**: The frontend will be built using either **React** or **Angular**, both of which are powerful JavaScript frameworks that facilitate the creation of dynamic, interactive user interfaces. This choice will enable the team to leverage reusable components, ensuring a maintainable and scalable codebase. The responsive design will cater to various screen sizes, improving user engagement.

**Backend**

* The backend will consist of a **RESTful API** composed of multiple microservices, each responsible for specific business functionalities. This modular approach will allow for independent deployment and scaling of services.
* **Framework and Technologies**:
  + **Spring Boot** will be used for API creation, enabling rapid development of RESTful services. Spring Boot’s configuration management and dependency injection features streamline the process of building and deploying microservices.
  + **Spring Data JPA** will facilitate interaction with the database, simplifying database operations and reducing boilerplate code.
  + **Hibernate** will be utilized as the ORM (Object-Relational Mapping) framework implementing JPA, enabling efficient data handling and relationships within the relational database.
  + **Spring Security** will manage authentication and authorization processes across microservices, ensuring secure access to APIs and protecting sensitive user data.

**Identity Provider and Authentication Server**

* The system will utilize the **OpenID Connect/OAuth2 protocol** as the authentication and authorization model. This will be implemented over secure SSL connections, ensuring that all data transmissions are encrypted and secure.
* This model will facilitate integration with third-party clients, allowing them to access the platform's data with user consent. It ensures a smooth user experience while maintaining robust security measures.
* A cloud service, either from the infrastructure provider or a dedicated third-party authentication service, will be leveraged to manage user identities and authentication processes efficiently.

**Database**

* A **relational database** such as **PostgreSQL** will be used for storing structured information related to schools, students, songs, and other entities. PostgreSQL is known for its reliability, robustness, and support for complex queries.
* Additionally, a **specialized time-series database** like **InfluxDB** will be implemented to store statistical data and user activity metrics. This database is optimized for handling time-stamped data efficiently, making it ideal for monitoring and analysis.

**Storage**

* The architecture will include integration with a **cloud storage service** to handle file storage needs. This will allow for the efficient management of files, images, and other media, ensuring that storage solutions can scale with the platform's needs.

**AI Services**

* The platform will leverage available **cloud services** for artificial intelligence or develop custom algorithms focused on **Natural Language Processing (NLP)** to implement the validation of song lyrics. This functionality will enhance user engagement by ensuring the quality and appropriateness of submitted content.

**External Integrations**

* The architecture will include **APIs for exoplanet information**, providing users with access to external data sources and enhancing the platform’s content offerings.
* **Partnerships with streaming service providers** will be established through APIs, enabling direct publishing of songs and facilitating seamless streaming to customers. This integration will expand the platform's reach and improve user experience.

**Infrastructure**

To support high availability and performance requirements, as well as to minimize the installation, configuration, and maintenance overhead of the platform, a reputable cloud service provider will be leveraged to host the platform. Among the leading options in the market, **Amazon Web Services (AWS)**, **Microsoft Azure**, and **Google Cloud Platform (GCP)** stand out as ideal candidates due to their extensive service offerings, global infrastructure, and robust performance capabilities.

Given that a microservices-based architecture will be employed, **Kubernetes** will serve as the orchestrator for the platform. This decision provides a solid foundation for dynamically scaling resources as needed, ensuring that the application can efficiently handle varying workloads while maintaining optimal performance. Furthermore, utilizing Kubernetes enables a more agile and flexible approach to evolving the platform in the future, allowing for easier deployment of new features and services. To facilitate this, it is proposed to utilize one of the fully managed Kubernetes services provided by these cloud providers, such as **Amazon EKS**, **Azure Kubernetes Service (AKS)**, or **Google Kubernetes Engine (GKE)**. These services will handle much of the complexity involved in managing Kubernetes clusters, allowing the development team to focus on building and deploying applications rather than infrastructure management.

In addition to Kubernetes, several key services will be necessary to enhance the platform's functionality and security:

* **API Gateway**: An API Gateway will be implemented to control and monitor API usage across the platform. This service will manage traffic, enforce security policies, and provide insights through analytics, ensuring that API calls are efficient and secure.
* **Web Application Firewall (WAF)**: To protect the platform from common web-based attacks, a Web Application Firewall will be integrated. The WAF will act as a protective barrier, filtering and monitoring HTTP traffic between the web application and the internet, thus safeguarding against threats such as SQL injection, cross-site scripting (XSS), and other vulnerabilities.
* **Content Delivery Network (CDN)**: To enhance the speed and reliability of content distribution, a CDN will be employed. This service will cache content at various edge locations around the globe, enabling faster delivery to users by reducing latency and improving load times for static and dynamic content.

By leveraging these advanced cloud services and technologies, the platform will not only achieve high availability and performance but also ensure robust security and scalability as user demands grow. This strategic approach will facilitate a smooth user experience while minimizing operational overhead.