Laboratorio 7

|  |  |
| --- | --- |
| Owner | Reviewer |
| Juan Sara, Marcelo Zapata, Luis Gutierrez |  |

**Summary:**

La siguiente propuesta de arquitectura, será orientada a una aplicación web para la subida y descarga de videos y PDF’s desde zonas rurales (aquellas que no contienen con mucho acceso a internet). Por ende, tendremos en cuenta lo siguiente:

- Web app (para no depender de una conexión a internet)

- Offline-first: La aplicacion web cachea las credenciales de usuarios (cada noche se realiza una syncronizacion de datos)

**Requerimientos:**

- Visualización de videos sin internet (streaming local)

- Cargar los PDF's de manera local

- Interfaz web muy simple, accesible desde navegadores comunes.

- Gestión de usuarios por escuela

- Sincronización automática de contenidos cuando haya conexión.

- Capacidad de borrar los contenidos semanalmente para limpiar almacenamiento

- Descarga de material durante la madrugada

**Out of Scope:**

* Alta escalabilidad
* Plataformas móviles nativas (apps para Android o iOS)
* Interacción en tiempo real entre alumnos y profesores (chat, videollamadas, etc.)
* Evaluaciones, quizzes o sistemas de calificaciones en línea
* Desarrollo de contenidos pedagógicos

**Keywords:**

- Servicio de autenticación: Valida credenciales de usuario que son periódicamente replicadas al servidor edge / CDN de forma que el login pueda ser offline

- Base de datos de contenido (PDF/videos) y de metadata (IDs, titles, versions, chunk info, state) que se replica en el servidor edge / CDN

- Uploader / chunk manager: Un cron job en Lima que interdiaria / semanalmente transmite el contenido en chunks usando una cola de eventos (BMQ) (esto nos ayuda por si se pierde la conexion porque podemos retomarla)

- Edge server sync services: Puede ser un servidor local o un CDN. Este servidor escucha los eventos emitidos por el BMQ y reconstruye los chunks en el contenido completo (PDF/Video). Una vez reconstruido, el contenido es agregado al cache del servidor edge o CDN.

- Edge server content cache: Este servidor o CDN mantiene de forma local (cache) contenido (PDF/Videos) y metadata. Esto le sirve a los estudiantes a hacer requests, si el contenido no se encuentra, el servidor edge lo llama a la central en ese momento (no es un caso ideal pero es lo que hay)

- Cliente offline (web app): La web app que corre en las laptops de los estudiantes se conectan al servidor edge (wifi o Lan). La app usa service workers (offline capabilities) y tambien el cache local de la maquina para guardar elementos de UI para que funcione de forma local (no puede guardar un video porque es muy pesado). Pero la app puede funcionar sin conexion a internet. La app llama al edge server usando un API gateway / LB para el login y la solicitud de contenido en ese momento.

- Cronjob de credenciales: Cada noche, un cron job envía nuevas o actualizadas credenciales a través de una cola de mensajes. Esto permite reintentos en caso de pérdida de conexión. El Edge Sync Service recibe las credenciales, las almacena en la Edge Content Cache y luego en el Edge Server, dejándolas disponibles localmente para la app web.

- Cronjob de contenido: También de forma nocturna, otro cron job descarga PDF y videos desde la base de datos en Lima. Usa un chunk manager y una cola de mensajes (BMQ) para tolerancia a fallos. El Edge Sync Service reconstruye los archivos y los guarda en la caché del Edge Server, listos para los estudiantes.

- Agente de la App Web: La app web almacena credenciales en caché para funcionar sin conexión. Si no están disponibles, consulta al servicio de login en Lima (evitándose si es posible). Luego accede al Edge Server y descarga el contenido directamente desde la caché local.

**Endpoints (cliente - edge server - backend):**

POST /login (authenticate user)

Get /content (puede dividirse en getVideo y getPDF) (list available content from edge cache or origin)

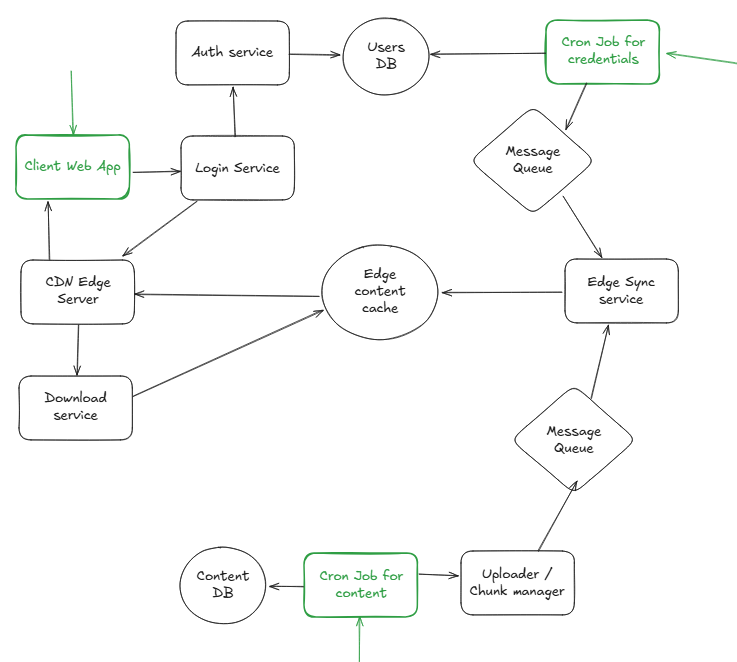
Get /download/{id} (download content by ID from edge cache or origin)

POST /sync-chunk (edge <-> central) (fetch pending content chunks (

POST /auth/sync (edge side): pull credentials from central (cron job)

**Technical proposal:**

Este es el diagrama de arquitectura:



Sin embargo, este posee las siguientes arquitecturas:

* MVC
* Arquitectura basada en eventos

A continuación, mostramos la propuesta de arquitectura

Diagrama

El contenido generado por IA puede ser incorrecto.

**Modularidad:**  
Diagrama

El contenido generado por IA puede ser incorrecto.

* Login/Authentication: Se encarga del login y gestión de usuarios.
* Reconnection: Maneja reintentos y sincronización en caso exista un corte de conexion.
* Download: proporciona contenido al usuario desde la caché o servidores CDN.
* Upload: gestiona la subida de los videos y PDF’s en el Edge Content Cache.

**Cohesion**: Existe una alta cohesión debido a que cada modulo esta separa del otro, por ejemplo:

* El Login/Authentication Module solo gestiona login y autenticación.
* El Download se enfoca en caché y distribución de contenido.
* El Upload Module maneja exclusivamente la carga y sincronización del contenido.
* El Reconnection Module está bien definido para mantener conectividad/credenciales.

**Coupling:** Es bajo, debido a que se comunican principalmente mediante colas de mensajes y caches compartidas, lo cual separa completamente los componentes. Además, se utilizan cron jobs, los cuales reducen aún más el acoplamiento, ya que actúan como intermediarios para ejecutar procesos en momentos determinados sin necesidad de una invocación directa entre módulos. Finalmente, aunque los módulos de Upload y Download interactúan con la misma base de datos (Edge content cache), ambos manejan los mismos atributos y tipos de contenido, lo que asegura coherencia sin generar dependencia lógica entre ellos.

**Fitness Functions:**

**Fitness Function (Disponibilidad - Fiabilidad del Sistema):**

El enfoque de la arquitectura se basa en asegurar la disponibilidad constante del contenido educativo, independientemente de la calidad de la conexión a internet.

La idea es que al menos el 90% de las solicitudes hechas por los estudiantes para acceder al contenido (videos o PDFs) sean procesadas correctamente y entreguen el contenido deseado, considerando tanto almacenamiento local como la sincronización automática de contenidos.

### Casos

\* Si el porcentaje de solicitudes procesadas exitosamente es ≥ 90% → Excelente

\* Si el porcentaje está entre 75% y 89% → Aceptable

\* Si el porcentaje está entre 50% y 74% → Deficiente

\* Si el porcentaje es < 50% → Inaceptable (sistema inestable)

Justificación técnica:

Esta métrica garantiza que, pese a limitaciones de red o almacenamiento, el sistema cumpla con la disponibilidad mínima para asegurar una experiencia educativa satisfactoria.

**Fitness Function (Performance - Rendimiento en Solicitudes):**

El enfoque de la arquitectura gira en torno a responder eficientemente a múltiples solicitudes simultáneas (Queries Per Second - QPS) para acceso al contenido educativo almacenado localmente o en streaming local.

La idea es que nuestro sistema soporte eficazmente hasta 10 QPS en condiciones estándar con el hardware comunitario propuesto (servidor local Raspberry Pi o equivalente).

### Casos

\* Si el sistema maneja ≥ 10 QPS de forma fluida → Excelente

\* Si el sistema maneja entre 6 y 9 QPS de forma estable → Decente

\* Si el sistema maneja entre 3 y 5 QPS → Poco eficiente

\* Si el sistema maneja ≤ 2 QPS → Inaceptable (demasiado lento)

Justificación técnica:

Esta métrica garantiza que la infraestructura propuesta pueda satisfacer simultáneamente las necesidades educativas del aula o grupo en zonas rurales, permitiendo una experiencia ágil y sin interrupciones.

**Fitness Function (Escalabilidad - Actualización/Agregado de Contenido):**

El enfoque de la arquitectura busca agilidad en la actualización o incorporación de nuevos contenidos educativos (videos, PDFs) sin interrumpir el uso del sistema por parte de los estudiantes.

El objetivo es que el sistema permita integrar o actualizar nuevos contenidos en menos de 1 hora desde que se inicia el proceso hasta que está disponible para los usuarios finales.

### Casos

\* Si el tiempo para actualizar o añadir contenido es ≤ 1 hora → Excelente

\* Si el tiempo está entre 1 y 2 horas → Aceptable

\* Si el tiempo está entre 2 y 4 horas → Deficiente

\* Si el tiempo es ≥ 4 horas → Inaceptable (no escalable)

Justificación técnica:

Esta métrica permite verificar que el sistema tiene procesos claros, optimizados y automatizados para incorporar rápidamente contenido educativo relevante según la necesidad académica o curriculares cambiantes.

**Modularidad – Cohesión:**

**Additional information:**

Arquitectura:

- Se utiliza un pipeline event-driven (cada noche a traves de un cron job) nuevo contenido es procesado por el uploader, que envia el contenido en chunk desde lima al edge server / CDN usando el BMQ.

- El edge server reconstruye el contenido y guardados en cache local (funciona como un CDN)

- Usar el BMQ permite la transimision de datos aunque exista una perdida de conexion

- Authentication y acceso offline: Junto al cron job y la sincronizacion de datos se envian los datos de usuarios nuevos y actualies (credenciales) en un JWT (json web token) el cual es guardados en las noches directamente en el cache de la web app en las computadoras de los alumnos ya que no pesa mucho) y esto asegura acceso offline a la app (las credenciales son sincronizadas periodicamente.