Análisis Detallado de los Esquemas Lexicon de AT Protocol para la Integración con el Sistema CORE

1. Introducción

El presente informe tiene como objetivo analizar y documentar en detalle los esquemas de datos ("Lexicons") utilizados por el AT Protocol, con un enfoque específico en aquellos observados en el flujo de eventos en tiempo real (firehose) de la red Bluesky, accesible mediante el endpoint com.atproto.sync.subscribeRepos. Esta documentación es fundamental para la correcta integración de estos esquemas con el "Sistema CORE", permitiendo un procesamiento de datos preciso y eficiente.

AT Protocol es un protocolo para redes sociales descentralizadas que busca ofrecer portabilidad de cuentas, elección algorítmica y federación interoperable. Un componente central de AT Protocol es el sistema "Lexicon", que define los esquemas para todas las estructuras de datos y API dentro del ecosistema.[1] Estos esquemas garantizan la interoperabilidad y la validación de datos entre diferentes implementaciones y servicios.

El firehose de Bluesky proporciona un flujo continuo de eventos de la red, como la creación de posts, likes, reposts, follows y actualizaciones de perfiles. Cada evento en este flujo, correspondiente a una operación en un repositorio de datos personal (PDS), incluye un registro de datos cuyo tipo está identificado por un campo $type. Este campo contiene el Identificador Namespaced (NSID) del Lexicon que define la estructura del registro.[2]

La metodología empleada para este informe incluye la identificación de los NSIDs relevantes a partir de la observación del firehose y la consulta de fuentes primarias, principalmente el repositorio oficial de AT Protocol. Posteriormente, se localizan las definiciones oficiales de los esquemas Lexicon, se analizan sus campos, tipos de datos, formatos, obligatoriedad y restricciones, y se documentan de manera estructurada.

Una comprensión profunda de estos Lexicons es crucial para el "Sistema CORE", ya que permitirá el parseo correcto de los eventos del firehose, la validación de la estructura de los datos recibidos, y la implementación de una lógica de procesamiento que pueda manejar la diversidad y evolución de estos esquemas.

2. Fuentes de Lexicons y Metodología de Identificación

La principal fuente para las definiciones canónicas de los Lexicons de AT Protocol,

incluyendo los específicos de la aplicación Bluesky (app.bsky.*) y los del protocolo base (com.atproto.*), es el repositorio oficial bluesky-social/atproto en GitHub.[1] Este repositorio contiene los archivos JSON que definen cada Lexicon.

Fuentes secundarias, como repositorios comunitarios (por ejemplo, lexicon-community/lexicon [5]), documentación oficial y no oficial, y Kits de Desarrollo de Software (SDKs) [2], también pueden ofrecer contexto y ejemplos de uso, aunque las definiciones del repositorio bluesky-social/atproto se consideran la autoridad.

La identificación de los Lexicons relevantes para este informe se basa en su aparición en el flujo de eventos (firehose) de com.atproto.sync.subscribeRepos. Cada operación de repositorio (commit) que aparece en este flujo contiene un registro de datos. Este registro incluye un campo especial denominado $type, cuyo valor es el Identificador Namespaced (NSID) del Lexicon que describe la estructura de dicho registro.[2] Por ejemplo, un post de Bluesky tendrá un $type de app.bsky.feed.post.

Un NSID es una cadena única, legible por humanos, que identifica un esquema Lexicon. Sigue un formato de nombre de dominio inverso, como com.atproto.server.createSession o app.bsky.feed.post.[8] Esta nomenclatura permite organizar los esquemas de manera jerárquica y asociarlos con la autoridad emisora del dominio.

Una vez identificado un NSID a través del firehose, el siguiente paso es localizar su archivo de definición JSON correspondiente dentro del directorio /lexicons/ del repositorio bluesky-social/atproto.[1] Estos archivos contienen la estructura completa del Lexicon, incluyendo su descripción, tipo (record, query, procedure, etc.), claves (si aplica), y la definición detallada de sus propiedades.

3. Documentación Detallada de Lexicons Relevantes del Firehose

A continuación, se documentan los Lexicons más comunes y relevantes observados en el firehose de Bluesky, esenciales para la integración con el "Sistema CORE". Para cada Lexicon, se proporciona su NSID, una descripción, el tipo, la clave (si es un tipo record), y un análisis detallado de su esquema.

3.1. com.atproto.repo.strongRef

- **NSID:** com.atproto.repo.strongRef
- **Fuente del Esquema:** [9]

```JSON
{
  "lexicon": 1,
```

```
 "id": "com.atproto.repo.strongRef",
 "description": "A URI with a content-hash fingerprint.",
 "defs": {
  "main": {
   "type": "object",
   "required": ["uri", "cid"],
   "properties": {
    "uri": { "type": "string", "format": "at-uri" },
    "cid": { "type": "string", "format": "cid" }
   }
  }
 }
}
```

- **Descripción Oficial:** "A URI with a content-hash fingerprint." (Una URI con una huella digital de hash de contenido).[9]
- **Tipo de Lexicon:** object (definición principal).
- **Análisis del Esquema (defs.main.properties):**
  - uri: La AT-URI del registro al que se hace referencia.
    - Tipo: string.
    - Formato: at-uri.
    - Requerido: Sí.
  - cid: El Content Identifier (CID) del registro específico al que se hace referencia, asegurando la inmutabilidad.
    - Tipo: string.
    - Formato: cid.
    - Requerido: Sí.
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
uri	AT-URI del registro referenciado.	string	at-uri	Sí	Debe ser una AT-URI válida.
cid	CID del contenido específico	string	cid	Sí	Debe ser un CID válido.

	del registro.				

- Este Lexicon es fundamental en AT Protocol, ya que proporciona un mecanismo para referenciar de manera inmutable una versión específica de un registro. Al incluir tanto la AT-URI (que identifica el "lugar" del registro) como el CID (que identifica el "contenido" exacto del registro en un momento dado), se asegura que la referencia apunta a un estado del dato que no puede cambiar sin cambiar también su CID. Esto es crucial para la integridad de las interacciones como "likes" y "reposts", así como para incrustar otros registros (por ejemplo, en citas de posts), ya que garantiza que la referencia siempre apunte al contenido exacto que se pretendía referenciar en el momento de la creación de la referencia.[4] El "Sistema CORE" debe ser capaz de interpretar estas referencias para vincular correctamente los datos relacionados.

3.2. app.bsky.feed.post

- **NSID:** app.bsky.feed.post
- **Fuente del Esquema:**.[1111]
- **Descripción Oficial:** "Record containing a Bluesky post." (Registro que contiene un post de Bluesky).[11]
- **Tipo de Lexicon:** record.
- **Clave (Key):** tid (Timestamp Identifier).[11]
- **Análisis del Esquema (defs.main.record.properties):**
  - text: Contenido principal del post.
    - Tipo: string. Requerido.
    - Restricciones: maxLength: 3000 (bytes), maxGraphemes: 300. Puede ser una cadena vacía si hay contenido incrustado (embeds).[11]
  - facets: Anotaciones de texto enriquecido (menciones, URLs, hashtags). Reemplaza al campo obsoleto entities.
    - Tipo: array. Opcional.
    - Items: Referencia a app.bsky.richtext.facet.[11]
  - reply: Objeto que indica si el post es una respuesta a otro.
    - Tipo: object (referencia a #/defs/replyRef). Opcional.
    - replyRef contiene root y parent, ambos de tipo com.atproto.repo.strongRef, apuntando al post raíz de la conversación y al post padre directo, respectivamente.[11]
  - embed: Contenido incrustado en el post.
    - Tipo: union. Opcional.
    - Referencias (refs): ["app.bsky.embed.images", "app.bsky.embed.video", "app.bsky.embed.external", "app.bsky.embed.record",

"app.bsky.embed.recordWithMedia"].[2] El objeto embed debe tener un campo $type que indique cuál de estos Lexicons describe su estructura.
    - langs: Array de códigos de idioma del contenido del post.
        - Tipo: array. Opcional.
        - Items: {"type": "string", "format": "language"}.
        - Restricciones: maxLength: 3 (para el array).[11]
    - labels: Auto-etiquetas aplicadas al post, como advertencias de contenido.
        - Tipo: union. Opcional.
        - Referencias (refs): ["com.atproto.label.defs#selfLabels"].[11]
    - tags: Hashtags adicionales, además de los que puedan estar en text y facets.
        - Tipo: array. Opcional.
        - Items: {"type": "string", "maxLength": 640, "maxGraphemes": 64}.
        - Restricciones: maxLength: 8 (para el array).[11]
    - createdAt: Marca de tiempo declarada por el cliente de cuándo se creó originalmente el post.
        - Tipo: string. Requerido.
        - Formato: datetime.[11]
    - entities: *OBSOLETO*. Reemplazado por facets para anotaciones de texto enriquecido.[11] Aunque obsoleto, puede aparecer en posts antiguos.
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
text	Contenido principal del post. Puede estar vacío si hay embeds.	string	-	Sí	maxLength: 3000, maxGraphemes: 300
facets	Anotaciones de texto (menciones, URLs, hashtags).	array	app.bsky.richtext.facet	No	-
reply	Referencias si el post es una respuesta.	object	#/defs/replyRef (usa com.atproto.repo.strongR	No	-

			ef)		
embed	Contenido incrustado (imágenes, vídeos, posts, etc.).	union	app.bsky.embed.images, app.bsky.embed.video, app.bsky.embed.external, app.bsky.embed.record, etc.	No	$type debe ser uno de los refs
langs	Idiomas del contenido del post.	array	language (string)	No	maxLength: 3
labels	Auto-etiquetas (ej. advertencias de contenido).	union	com.atproto.label.defs#selfLabels	No	-
tags	Hashtags adicionales.	array	string	No	maxLength: 8 (array), maxLength: 640, maxGraphemes: 64 (string items)
createdAt	Fecha de creación declarada por el cliente.	string	datetime	Sí	-
entities	*OBSOLETO*. Reemplazado por facets.	array	#/defs/entity	No	-

- La estructura de app.bsky.feed.post es central para la actividad en Bluesky. La transición del campo entities al más robusto y extensible facets (que utiliza app.bsky.richtext.facet) indica una mejora en cómo se manejan las anotaciones de texto enriquecido.[11] El "Sistema CORE" debería priorizar el procesamiento de

facets para la información de texto enriquecido, aunque podría necesitar manejar entities por retrocompatibilidad con posts más antiguos. La flexibilidad del campo embed es una característica poderosa, ya que es una "unión abierta".[2] Esto significa que, aunque hay un conjunto de tipos de incrustación comunes definidos en refs, teóricamente podrían aparecer otros tipos definidos por terceros. El "Sistema CORE" debe inspeccionar el campo $type dentro del objeto embed para determinar cómo procesar su contenido y estar preparado para encontrar tipos no anticipados.

3.3. app.bsky.feed.like

- **NSID:** app.bsky.feed.like
- **Fuente del Esquema:**.[1010]
- **Descripción Oficial:** "Record declaring a 'like' of a piece of subject content." (Registro que declara un "like" a una pieza de contenido).[10]
- **Tipo de Lexicon:** record.
- **Clave (Key):** tid.[10]
- **Análisis del Esquema (defs.main.record.properties):**
  - subject: Referencia fuerte al contenido (ej. un post) que recibe el "like".
    - Tipo: ref. Requerido.
    - Referencia: com.atproto.repo.strongRef.[10]
  - createdAt: Marca de tiempo de cuándo se creó el "like".
    - Tipo: string. Requerido.
    - Formato: datetime.[10]
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
subject	Referencia fuerte al contenido que recibe el "like".	object	com.atproto.repo.strongRef	Sí	-
createdAt	Fecha de creación del "like".	string	datetime	Sí	-

- Los registros de tipo app.bsky.feed.like son una de las interacciones más frecuentes en el firehose.[3] Su estructura es simple, conteniendo esencialmente

una referencia al objeto del "like" y una marca de tiempo. La utilización de com.atproto.repo.strongRef para el campo subject es crucial, ya que asegura que el "like" está vinculado de forma inequívoca a una versión específica e inmutable del contenido al que se le dio "like".[10] Esto evita ambigüedades si el contenido original es modificado posteriormente. Dada la alta frecuencia de estos eventos, el "Sistema CORE" debe estar optimizado para procesarlos eficientemente, incluyendo la resolución y el almacenamiento de la relación entre el "like" y el contenido referenciado.

3.4. app.bsky.feed.repost

- **NSID:** app.bsky.feed.repost
- **Fuente del Esquema:**.[44]
- **Descripción Oficial:** "Record representing a 'repost' of an existing Bluesky post." (Registro que representa un "repost" de un post de Bluesky existente).[4]
- **Tipo de Lexicon:** record.
- **Clave (Key):** tid.[4]
- **Análisis del Esquema (defs.main.record.properties):**
  - subject: Referencia fuerte al post original que se está reposteando.
    - Tipo: ref. Requerido.
    - Referencia: com.atproto.repo.strongRef.[4]
  - createdAt: Marca de tiempo de cuándo se creó el repost.
    - Tipo: string. Requerido.
    - Formato: datetime.[4]
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
subject	Referencia fuerte al post original que se está reposteando.	object	com.atproto.repo.strongRef	Sí	-
createdAt	Fecha de creación del repost.	string	datetime	Sí	-

- La estructura de un app.bsky.feed.repost es notablemente similar a la de un app.bsky.feed.like. Ambos consisten en un campo subject que es una

com.atproto.repo.strongRef y un campo createdAt.[4] Esta similitud estructural subraya un patrón de diseño donde interacciones distintas pueden compartir una forma de datos común, diferenciándose semánticamente por su NSID (el valor del campo $type del registro). Para el "Sistema CORE", es imperativo distinguir entre un "like" y un "repost" basándose en el $type del registro general, y no solo en la estructura de sus campos, para interpretar correctamente la intención del usuario y la naturaleza de la interacción.

3.5. app.bsky.graph.follow

- **NSID:** app.bsky.graph.follow
- **Fuente del Esquema:** [13] (El contenido JSON se puede encontrar en este enlace).
- **Descripción Oficial:** "Record declaring a social 'follow' relationship of another account. Duplicate follows will be ignored by the AppView." (Registro que declara una relación social de "seguimiento" de otra cuenta. Los seguimientos duplicados serán ignorados por la AppView).[13]
- **Tipo de Lexicon:** record.
- **Clave (Key):** tid.[13]
- **Análisis del Esquema (defs.main.record.properties):**
  - subject: El DID (Decentralized Identifier) de la cuenta que se está siguiendo.
    - Tipo: string. Requerido.
    - Formato: did.[13]
  - createdAt: Marca de tiempo de cuándo se creó la relación de seguimiento.
    - Tipo: string. Requerido.
    - Formato: datetime.[13]
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
subject	El DID (Decentralized Identifier) de la cuenta a la que se sigue.	string	did	Sí	Debe ser un DID válido.
createdAt	Fecha de creación de la relación de	string	datetime	Sí	-

	seguimiento.				

- A diferencia de los Lexicons app.bsky.feed.like y app.bsky.feed.repost que utilizan com.atproto.repo.strongRef para referenciar una pieza de *contenido* específico, el Lexicon app.bsky.graph.follow utiliza un did (Decentralized Identifier) en su campo subject.[13] Esta distinción es lógica y fundamental: una acción de "seguir" se dirige a una *cuenta* o *identidad* en su totalidad, representada por su DID, y no a un registro o contenido particular emitido por esa cuenta. El "Sistema CORE" debe, por lo tanto, tratar estas referencias de manera diferente, vinculando los eventos de seguimiento a identificadores de actor en lugar de a identificadores de contenido.

3.6. app.bsky.actor.profile

- **NSID:** app.bsky.actor.profile
- **Fuente del Esquema:**.[14][14]
- **Descripción Oficial:** "A declaration of a Bluesky account profile." (Una declaración de un perfil de cuenta de Bluesky).[14]
- **Tipo de Lexicon:** record.
- **Clave (Key):** literal:self.[14] Esto indica que solo puede existir un registro de este tipo por repositorio de actor, es decir, un perfil único por cuenta.
- **Análisis del Esquema (defs.main.record.properties):**
  - displayName: Nombre visible del perfil.
    - Tipo: string. Opcional.
    - Restricciones: maxGraphemes: 64, maxLength: 640.[14]
  - description: Texto de descripción de formato libre del perfil.
    - Tipo: string. Opcional.
    - Restricciones: maxGraphemes: 256, maxLength: 2560.[14]
  - avatar: Imagen pequeña para mostrar junto a los posts de la cuenta (foto de perfil).
    - Tipo: blob. Opcional.
    - Restricciones: accept: ["image/png", "image/jpeg"], maxSize: 1000000 (1MB).[14]
  - banner: Imagen horizontal más grande para mostrar detrás de la vista del perfil.
    - Tipo: blob. Opcional.
    - Restricciones: accept: ["image/png", "image/jpeg"], maxSize: 1000000 (1MB).[14]
  - labels: Valores de auto-etiquetado específicos de la aplicación Bluesky, aplicados a la cuenta en general.

        - Tipo: union. Opcional.
        - Referencias (refs): ["com.atproto.label.defs#selfLabels"].[14]
    - joinedViaStarterPack: Referencia al paquete de inicio si el usuario se unió a través de uno.
        - Tipo: ref. Opcional.
        - Referencia: com.atproto.repo.strongRef.[14]
    - pinnedPost: Referencia a un post fijado en el perfil del usuario.
        - Tipo: ref. Opcional.
        - Referencia: com.atproto.repo.strongRef.[14]
    - createdAt: Fecha de creación del perfil.
        - Tipo: string. Opcional.
        - Formato: datetime.[14]
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
displayName	Nombre visible del perfil.	string	-	No	maxGraphemes: 64, maxLength: 640
description	Descripción de formato libre del perfil.	string	-	No	maxGraphemes: 256, maxLength: 2560
avatar	Imagen pequeña para mostrar junto a los posts (foto de perfil).	blob	-	No	accept: ["image/png", "image/jpeg"], maxSize: 1000000
banner	Imagen horizontal más grande para mostrar detrás de la vista del perfil.	blob	-	No	accept: ["image/png", "image/jpeg"], maxSize: 1000000

labels	Valores de auto-etiquetado para la cuenta en general.	union	com.atproto.label.defs#selfLabels	No	-
joinedViaStarterPack	Referencia al paquete de inicio si el usuario se unió a través de uno.	ref	com.atproto.repo.strongRef	No	-
pinnedPost	Referencia a un post fijado en el perfil.	ref	com.atproto.repo.strongRef	No	-
createdAt	Fecha de creación del perfil.	string	datetime	No	-

- El uso de literal:self como clave para app.bsky.actor.profile es un patrón importante en AT Protocol que designa un registro "singleton" dentro de un repositorio de actor.[14] Esto significa que cada cuenta solo puede tener un perfil. La opcionalidad de la mayoría de los campos permite que los perfiles varíen en su grado de detalle. La inclusión de campos más recientes como joinedViaStarterPack y pinnedPost [14] demuestra cómo los perfiles pueden evolucionar para incorporar metadatos más estructurados y referencias a otras entidades dentro de la red. El "Sistema CORE" debe estar preparado para manejar perfiles con diferentes niveles de completitud y para la posible adición de nuevos campos opcionales en el futuro.

3.7. app.bsky.graph.block

- **NSID:** app.bsky.graph.block
- **Fuente del Esquema:** [15] (El contenido JSON se puede encontrar en este enlace).
- **Descripción Oficial:** "Record declaring a 'block' relationship against another account. NOTE: blocks are public in Bluesky; see blog posts for details." (Registro que declara una relación de "bloqueo" contra otra cuenta. NOTA: los bloqueos son públicos en Bluesky; ver publicaciones de blog para más detalles).[15]
- **Tipo de Lexicon:** record.

- **Clave (Key):** tid.[15]
- **Análisis del Esquema (defs.main.record.properties):**
  - subject: El DID de la cuenta que se está bloqueando.
    - Tipo: string. Requerido.
    - Formato: did.
    - Descripción: "DID of the account to be blocked.".[15]
  - createdAt: Marca de tiempo de cuándo se creó el bloqueo.
    - Tipo: string. Requerido.
    - Formato: datetime.[15]
- **Tabla de Documentación:**

Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
subject	DID de la cuenta a ser bloqueada.	string	did	Sí	Debe ser un DID válido.
createdAt	Fecha de creación del bloqueo.	string	datetime	Sí	-

- La estructura de app.bsky.graph.block es similar a la de app.bsky.graph.follow, ya que ambas acciones se dirigen a una cuenta identificada por su did.[13] Una característica distintiva de los bloqueos en Bluesky, como se indica en su descripción oficial, es su naturaleza pública.[15] Esta es una decisión de diseño con implicaciones significativas para la dinámica social y la privacidad en la plataforma. Si el "Sistema CORE" necesita procesar o almacenar información sobre bloqueos, debe ser consciente de esta publicidad y manejar los datos de acuerdo con las políticas de privacidad y uso correspondientes.

3.8. Lexicons de Incrustación (app.bsky.embed.*)

Estos Lexicons se utilizan dentro del campo embed del Lexicon app.bsky.feed.post para incluir diversos tipos de contenido multimedia o referencias a otros registros.

3.8.1. app.bsky.embed.images

- **NSID:** app.bsky.embed.images
- **Fuente del Esquema:**.[1616]
- **Descripción Oficial:** "A set of images embedded in a Bluesky record (eg, a

post)." (Un conjunto de imágenes incrustadas en un registro de Bluesky, ej. un post).[16]

- **Tipo de Lexicon:** object (para defs.main).
- **Análisis del Esquema:**
  - defs.main.properties.images: Un array de objetos de imagen.
    - Tipo: array. Requerido.
    - Items: Referencia a #/defs/image.
    - Restricciones: maxLength: 4.[16]
  - defs.image: Define la estructura de una imagen para su creación y almacenamiento.
    - image: El blob de la imagen (datos binarios).
      - Tipo: blob. Requerido.
      - Restricciones: accept: ["image/*"] (o más específicamente ["image/png", "image/jpeg"]), maxSize: 1000000 (1MB).[16]
    - alt: Texto alternativo para accesibilidad.
      - Tipo: string. Requerido.[16]
    - aspectRatio (Opcional): Proporción de la imagen.
      - Tipo: ref. Referencia a app.bsky.embed.defs#aspectRatio.[16]
  - defs.viewImage (para visualización, a menudo con URLs de CDN):
    - thumb: URL completamente calificada del thumbnail de la imagen.
      - Tipo: string. Formato: uri. Requerido.[16]
    - fullsize: URL completamente calificada de la imagen en tamaño grande.
      - Tipo: string. Formato: uri. Requerido.[16]
    - alt: Texto alternativo.
      - Tipo: string. Requerido.[16]
    - aspectRatio (Opcional): Proporción.
      - Tipo: ref. Referencia a app.bsky.embed.defs#aspectRatio.[16]
- **Tabla de Documentación:**

Objeto/Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
main	Contenedor principal para imágenes incrustadas.	object	-	-	-
main.images	Array de	array	#/defs/imag	Sí	maxLength:

	imágenes.		e		4
image (defs)	Definición de una imagen para almacenamiento/creación.	object	-	-	-
image.image	El blob de la imagen.	blob	-	Sí	accept: ["image/*"], maxSize: 1000000
image.alt	Texto alternativo para accesibilidad.	string	-	Sí	-
image.aspectRatio	Proporción de la imagen.	ref	app.bsky.embed.defs#aspectRatio	No	-
viewImage (defs)	Definición de una imagen para visualización (con URLs de CDN).	object	-	-	-
viewImage.thumb	URL del thumbnail de la imagen.	string	uri	Sí	-
viewImage.fullsize	URL de la imagen a tamaño completo.	string	uri	Sí	-
viewImage.alt	Texto alternativo para accesibilidad	string	-	Sí	-

	.				
viewImage.aspectRatio	Proporción de la imagen.	ref	app.bsky.embed.defs#aspectRatio	No	-

- Es crucial distinguir entre la definición image (que contiene el blob de la imagen, usado al crear el post) y la definición viewImage (que contiene URLs thumb y fullsize, probablemente apuntando a una CDN, usada para mostrar la imagen en clientes).[16] Cuando un post con imágenes se crea y aparece en el firehose, el registro contendrá la estructura image con el blob (o una referencia al mismo). El "Sistema CORE" debe estar preparado para manejar estos blobs si necesita procesar las imágenes directamente, o entender que las URLs en viewImage son para consumo del cliente final.

3.8.2. app.bsky.embed.external

- **NSID:** app.bsky.embed.external
- **Fuente del Esquema:** Aunque el JSON completo no se encuentra directamente en los fragmentos proporcionados [24], su estructura es referenciada y utilizada en app.bsky.feed.post.[2] Su uso para generar vistas previas de enlaces se describe en.[17]
- **Descripción Oficial:** (Deducida) "An external link embedded in a record, typically with preview information." (Un enlace externo incrustado en un registro, típicamente con información de vista previa).
- **Tipo de Lexicon:** object.
- **Análisis del Esquema (deducido de app.bsky.embed.external#main y su uso):**
  - external: Objeto que contiene los detalles del enlace externo y su vista previa.
    - uri: La URI del recurso externo.
      - Tipo: string. Formato: uri. Requerido.[12]
    - title: Título de la página enlazada.
      - Tipo: string. Requerido.
    - description: Descripción de la página enlazada.
      - Tipo: string. Requerido.
    - thumb (Opcional): Blob o URL de la imagen en miniatura para la vista previa.
      - Tipo: blob (para el registro) o string con formato uri (para la vista del cliente, apuntando a una imagen de la vista previa).[17]
- **Tabla de Documentación:**

Objeto/Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
main	Contenedor principal para el enlace externo incrustado.	object	-	-	-
main.external	Objeto que contiene los detalles del enlace externo.	object	#/defs/external (hipotético)	Sí	-
external (defs)	Definición del enlace externo.	object	-	-	-
external.uri	URI del recurso externo.	string	uri	Sí	-
external.title	Título de la página enlazada.	string	-	Sí	-
external.description	Descripción de la página enlazada.	string	-	Sí	-
external.thumb	Miniatura para la vista previa del enlace.	blob (o string uri)	(o string uri en la vista del cliente)	No	accept: ["image/*"], maxSize: 1000000 (si es blob)

- La generación de la información de vista previa de un enlace (título, descripción, miniatura) es típicamente una responsabilidad del cliente o de un servicio intermediario en el momento de la creación del post.[17] Por lo tanto, el registro app.bsky.embed.external que aparece en el firehose generalmente contendrá

estos campos ya poblados. El "Sistema CORE" puede esperar estos metadatos pre-extraídos y utilizarlos para mostrar información del enlace o para análisis.

3.8.3. app.bsky.embed.record

- **NSID:** app.bsky.embed.record
- **Fuente del Esquema:** Referenciado en app.bsky.feed.post.[2] Su uso para citar posts se describe en..[17][25]
- **Descripción Oficial:** (Deducida) "A record (e.g., another post) embedded in a record." (Un registro, por ejemplo, otro post, incrustado en un registro).
- **Tipo de Lexicon:** object.
- **Análisis del Esquema (deducido de su uso):**
  - record: Una referencia fuerte al registro que se está incrustando.
    - Tipo: ref. Requerido.
    - Referencia: com.atproto.repo.strongRef.
- **Tabla de Documentación:**

Objeto/Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
main	Contenedor principal para el registro incrustado.	object	-	-	-
main.record	Referencia fuerte al registro que se está incrustando.	object	com.atproto.repo.strongRef	Sí	-

- Este Lexicon es fundamental para la componibilidad dentro de Bluesky, permitiendo funcionalidades como citar posts.[17] Al incrustar un com.atproto.repo.strongRef, se asegura una referencia inmutable al post citado. El "Sistema CORE" necesitará resolver esta strongRef (obteniendo el registro por su URI y CID) si desea acceder, mostrar o analizar los detalles del post incrustado.

3.8.4. app.bsky.embed.recordWithMedia

- **NSID:** app.bsky.embed.recordWithMedia
- **Fuente del Esquema:** Referenciado en la unión embed de app.bsky.feed.post.[2]

- **Descripción Oficial:** (Deducida) "A record embedded alongside media (images or external links)." (Un registro incrustado junto con medios como imágenes o enlaces externos).
- **Tipo de Lexicon:** object.
- **Análisis del Esquema (deducido de su nombre y contexto):**
  - record: El componente de registro incrustado.
    - Tipo: object. Requerido.
    - Propiedades:
      - record: Referencia fuerte al registro principal que se está incrustando (ej. un post que se cita).
        - Tipo: ref. Referencia: com.atproto.repo.strongRef. Requerido.
  - media: El componente de medios incrustado junto al registro.
    - Tipo: union. Requerido.
    - Referencias (refs): ["app.bsky.embed.images", "app.bsky.embed.external"]. El objeto media debe tener un campo $type que indique cuál de estos Lexicons describe su estructura.
- **Tabla de Documentación:**

Objeto/Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
main	Contenedor principal para el registro con medios incrustados.	object	-	-	-
main.record	El componente de registro incrustado.	object	Contiene un campo record de tipo com.atproto.repo.strongRef	Sí	-
main.record.record	Referencia fuerte al registro principal incrustado.	object	com.atproto.repo.strongRef	Sí	-

main.media	El componente de medios incrustado.	union	app.bsky.embed.images, app.bsky.embed.external	Sí	$type debe ser uno de los refs

- app.bsky.embed.recordWithMedia permite combinar una referencia a otro registro (como una cita) con contenido multimedia adicional (imágenes o un enlace externo) dentro de un único objeto embed. Esta estructura anidada implica que el "Sistema CORE" debe estar preparado para parsear el objeto record (que contiene una strongRef) y, por separado, el objeto media, que a su vez es una unión y requerirá la inspección de su propio campo $type para determinar si contiene imágenes o un enlace externo.

3.9. app.bsky.richtext.facet

- **NSID:** app.bsky.richtext.facet
- **Fuente del Esquema:**.[18][18]
- **Descripción Oficial:** "Annotation of a sub-string within rich text." (Anotación de una sub-cadena dentro de texto enriquecido).[18]
- **Tipo de Lexicon:** object (para defs.main).
- **Análisis del Esquema (defs.main.properties):**
  - index: Especifica el rango del sub-string (dentro del texto del post) al que se aplica la faceta.
    - Tipo: ref. Requerido.
    - Referencia: #/defs/byteSlice.[18]
  - features: Un array de características de la faceta. Cada característica define el tipo de anotación (mención, enlace, etiqueta).
    - Tipo: array. Requerido.
    - Items: {"type": "union", "refs": ["#mention", "#link", "#tag"]}. Cada item en el array debe tener un campo $type (ej. app.bsky.richtext.facet#mention).[18]
- **Definiciones Auxiliares (defs):**
  - mention: Característica de faceta para la mención de otra cuenta.
    - Contiene did (Tipo: string, Formato: did, Requerido).[18]
  - link: Característica de faceta para una URL.
    - Contiene uri (Tipo: string, Formato: uri, Requerido).[18]
  - tag: Característica de faceta para un hashtag.
    - Contiene tag (Tipo: string, Requerido, Restricciones: maxLength: 640, maxGraphemes: 64).[18]
  - byteSlice: Especifica el rango del sub-string al que se aplica una

característica de faceta. Los índices cuentan bytes del texto codificado en UTF-8.
    - Contiene byteStart (Tipo: integer, minimum: 0, Requerido) y byteEnd (Tipo: integer, minimum: 0, Requerido). byteStart es inclusivo, byteEnd es exclusivo.[18]
- **Tabla de Documentación:**

Objeto/Campo	Descripción	Tipo de Dato	Formato/Ref	Requerido	Restricciones/Validación
main	Anotación de un sub-string en texto enriquecido.	object	-	-	-
main.index	Rango del sub-string al que se aplica la faceta.	object	#/defs/byteSlice	Sí	-
main.features	Array de características de la faceta (mención, enlace, tag).	array	union: [#mention, #link, #tag]	Sí	Cada feature tiene su $type.
mention (defs)	Característica de faceta para mención de cuenta.	object	-	-	(Dentro de features union)
mention.did	DID de la cuenta mencionada.	string	did	Sí	-
link (defs)	Característica de faceta para URL.	object	-	-	(Dentro de features union)

link.uri	URI completa del enlace.	string	uri	Sí	-
tag (defs)	Característica de faceta para hashtag.	object	-	-	(Dentro de features union)
tag.tag	El tag del hashtag (sin '#', excepto para 'double hash tags').	string	-	Sí	maxLength: 640, maxGraphemes: 64
byteSlice (defs)	Rango de sub-string (contando bytes UTF-8).	object	-	-	(Referenciado por main.index)
byteSlice.byteStart	Índice de inicio del byte (inclusivo).	integer	-	Sí	minimum: 0
byteSlice.byteEnd	Índice de fin del byte (exclusivo).	integer	-	Sí	minimum: 0

- El Lexicon app.bsky.richtext.facet es el método estándar y actual para anotar texto enriquecido en Bluesky, habiendo reemplazado al campo entities en app.bsky.feed.post.[11] Una especificación técnica crucial es que byteSlice utiliza índices basados en bytes del texto codificado en UTF-8.[18] Esto es particularmente importante para los desarrolladores del "Sistema CORE", ya que muchos lenguajes de programación (como JavaScript) utilizan internamente UTF-16 o puntos de código Unicode para la indexación de cadenas. Se requerirán conversiones precisas a arrays de bytes y un manejo cuidadoso de estos índices para interpretar correctamente las facetas y aplicar las anotaciones a las subcadenas correctas del texto del post.

4. Tipos de Datos, Formatos y Patrones Comunes en Lexicons

El sistema Lexicon de AT Protocol emplea un conjunto de tipos de datos primitivos y complejos, formatos específicos y patrones de diseño que son consistentes a través de los diferentes esquemas.

4.1. Tipos Primitivos y Complejos

- **string**: Cadenas de texto. A menudo vienen con restricciones como maxLength (longitud máxima en bytes) y maxGraphemes (número máximo de grafemas, para un conteo más preciso de caracteres visibles).[11]
- **integer**: Números enteros. Pueden tener restricciones como minimum y maximum.[11]
- **boolean**: Valores lógicos verdadero/falso.
- **object**: Estructuras de datos anidadas que contienen un conjunto de propiedades con nombre. Son la base para definir la mayoría de los registros y tipos complejos.[4]
- **array**: Listas ordenadas de ítems. Los arrays pueden tener una maxLength y sus ítems suelen ser de un tipo específico o una referencia.[11]
- **blob**: Datos binarios, comúnmente utilizados para imágenes. Los blobs tienen propiedades como accept (una lista de tipos MIME permitidos) y maxSize (tamaño máximo en bytes).[14]
- **ref**: Referencias a otras definiciones. Pueden ser referencias locales dentro del mismo archivo Lexicon (ej., #/defs/someObject) o referencias a otros Lexicons completamente calificados (ej., com.atproto.repo.strongRef).[4]
- **union**: Permite que un campo pueda ser uno de varios tipos definidos. Los tipos posibles se listan en una propiedad refs. El objeto real en una instancia de datos debe incluir un campo $type que discrimine cuál de los tipos de la unión está presente.[2]

4.2. Formatos Específicos Comunes

Ciertos campos de tipo string utilizan formatos predefinidos para asegurar la consistencia y validación de datos específicos:

- **datetime**: Fechas y horas, generalmente en formato ISO 8601. Usado para campos como createdAt.[4]
- **uri**: Identificadores Uniformes de Recursos genéricos.[16]
- **at-uri**: Un esquema URI específico de AT Protocol para referenciar de manera única repositorios, colecciones y registros dentro de la red.[9]
- **cid**: Content Identifiers, usados en IPFS y otros sistemas de contenido direccionable. Identifican un bloque de datos por su hash.[9]
- **did**: Identificadores Descentralizados (especificación del W3C), usados para

identificar de manera única a los actores (usuarios, servicios) en la red.[13]
- **at-identifier**: Un identificador que puede ser un did o un handle (nombre de usuario legible por humanos, ej. nombre.bsky.social).[20]
- **language**: Códigos de idioma estándar, como "en" para inglés o "es" para español.[11]

4.3. Uniones Abiertas y el Discriminador $type

Un patrón de diseño particularmente importante en Lexicons es el uso de "uniones", especialmente "uniones abiertas". Como se observa en el campo embed de app.bsky.feed.post, una unión permite que un campo contenga diferentes tipos de objetos.[2] Cada uno de estos objetos debe llevar un campo $type que especifica el NSID del Lexicon que define su estructura particular.[2]

La naturaleza de "unión abierta", como se describe en [2], significa que el campo embed puede, en teoría, contener cualquier tipo de objeto definido por un Lexicon, incluso si ese Lexicon no está listado explícitamente en la propiedad refs de la definición de la unión en app.bsky.feed.post. Esto ofrece una gran extensibilidad, permitiendo que nuevas aplicaciones o características introduzcan sus propios tipos de contenido incrustable sin necesidad de modificar el Lexicon central de app.bsky.feed.post.[23]

Para el "Sistema CORE", esta flexibilidad implica que no se puede asumir que el contenido de un campo de unión siempre corresponderá a uno de los tipos predefinidos. Es esencial inspeccionar el valor del campo $type y utilizarlo para seleccionar el esquema correcto para el parseo y la validación. Si se encuentra un $type desconocido, el sistema debe tener una estrategia definida para manejarlo, como ignorar el contenido desconocido, almacenarlo como datos opacos o generar una alerta para una futura actualización del esquema.

5. Relaciones Inter-Lexicon e Incrustación (Embedding)

Los Lexicons no existen en aislamiento; están interconectados a través de mecanismos de referencia y patrones de incrustación, lo que permite la creación de estructuras de datos ricas y compuestas.

5.1. Mecanismos de Referencia (ref)

El tipo ref es el principal mecanismo por el cual un Lexicon puede hacer referencia a definiciones dentro de sí mismo o a otros Lexicons.

- **Referencias locales:** Usan la sintaxis #/defs/definitionName para apuntar a una

definición (generalmente un object) dentro de la sección defs del mismo archivo Lexicon. Un ejemplo es el campo reply en app.bsky.feed.post, que referencia a #/defs/replyRef.[11]
- **Referencias externas:** Usan el NSID completo de otro Lexicon, a veces seguido de #definitionName si se refiere a una definición específica dentro de ese Lexicon externo (aunque comúnmente se refiere a la definición main). Por ejemplo, el campo subject en app.bsky.feed.like referencia a com.atproto.repo.strongRef.[10] De manera similar, el campo facets en app.bsky.feed.post es un array de ítems que referencian a app.bsky.richtext.facet.[11]

Estas referencias permiten la reutilización de esquemas comunes (como strongRef) y la construcción de tipos complejos a partir de componentes más simples.

5.2. Estructura del Contenido Incrustado (embed) en Posts

El campo embed dentro del Lexicon app.bsky.feed.post es un ejemplo prominente de cómo se logra la composición y la flexibilidad en AT Protocol. Este campo es de tipo union, con una lista de refs que apuntan a varios Lexicons de incrustación, tales como app.bsky.embed.images, app.bsky.embed.external, app.bsky.embed.record, y app.bsky.embed.recordWithMedia.[2]

Cuando un post contiene contenido incrustado, el objeto embed tendrá un campo $type que indica cuál de estos Lexicons de incrustación define su estructura. Por ejemplo, si un post incrusta imágenes, el campo embed tendrá $type: "app.bsky.embed.images" y su contenido seguirá el esquema de app.bsky.embed.images.

Esta arquitectura es central para la riqueza expresiva de los posts en Bluesky. Para el "Sistema CORE", implica la necesidad de implementar una suerte de "despachador" (dispatcher) o lógica condicional basada en el valor del campo $type del objeto embed. Una vez que se determina el tipo de incrustación, el sistema puede aplicar el esquema Lexicon correspondiente para parsear, validar y procesar adecuadamente el contenido incrustado específico. Esta aproximación asegura que el sistema pueda manejar la diversidad de tipos de incrustaciones y adaptarse a la posible adición de nuevos tipos en el futuro.

6. Consideraciones Clave para la Integración con el "Sistema CORE"

La integración exitosa de los datos del firehose de AT Protocol con el "Sistema CORE"

requiere atención a varios aspectos técnicos y de diseño.

6.1. Parseo y Validación de Eventos del Firehose

Los eventos del firehose se transmiten generalmente en formato JSON (aunque Jetstream, un servicio intermediario, los convierte de CBOR a JSON 3). El "Sistema CORE" debe ser capaz de parsear estos objetos JSON.
Fundamentalmente, cada registro de datos dentro de un evento (por ejemplo, el contenido de un create o update) debe ser validado contra su esquema Lexicon correspondiente. Este esquema se identifica mediante el campo $type presente en el registro.
Se pueden utilizar herramientas como lex-cli (mencionada en el contexto de la generación de interfaces TypeScript a partir de Lexicons 2) o bibliotecas de validación de esquemas JSON adaptadas para Lexicons para realizar esta validación. Generar modelos de datos o clases en el lenguaje de programación del "Sistema CORE" a partir de las definiciones Lexicon también puede facilitar un parseo y manejo de datos más seguro y tipado.

6.2. Manejo de Operaciones de Repositorio (create, update, delete)

El firehose de com.atproto.sync.subscribeRepos no solo emite eventos para nuevos registros (operación create), sino también para actualizaciones de registros existentes (update) y eliminaciones de registros (delete).3
El "Sistema CORE" debe estar diseñado para manejar estas tres operaciones y actualizar su estado interno o base de datos en consecuencia.

- Para operaciones create, el evento contendrá el registro completo.
- Para operaciones update, el evento también contendrá el registro completo con los nuevos datos.
- Para operaciones delete, el evento típicamente solo proporciona los identificadores clave del registro a eliminar: el did del repositorio, la collection (NSID del tipo de registro) y la rkey (record key) del registro dentro de la colección.[3] No se incluye el cuerpo del registro eliminado.

El manejo de operaciones delete es particularmente importante. Dado que solo se proporciona esta información mínima, el "Sistema CORE" debe mantener un índice eficiente que le permita localizar y eliminar (o marcar como eliminado) el registro correspondiente en su propio almacenamiento de datos.

6.3. Estrategias para la Evolución de Esquemas y Uniones Abiertas

Como se discutió anteriormente (Sección 4.3), los Lexicons pueden utilizar "uniones abiertas", lo que significa que el "Sistema CORE" podría encontrar valores de $type en campos de unión que no reconoce o para los cuales no tiene un esquema predefinido. Se debe implementar una estrategia robusta para estos casos, como registrar el tipo desconocido y los datos asociados, o ignorarlos si no son críticos.

Los esquemas Lexicon también pueden evolucionar con el tiempo (nuevas versiones, adición de campos opcionales, obsolescencia de campos antiguos). El "Sistema CORE" debe ser diseñado con la flexibilidad para adaptarse a estos cambios. Esto podría implicar:

- Actualizar periódicamente las definiciones de Lexicon que utiliza.
- Ser tolerante a campos desconocidos (especialmente si son opcionales) en los registros recibidos.
- Implementar una lógica de migración o adaptación si los cambios en los esquemas son significativos. Considerar un "modo permisivo" (que intente procesar datos incluso si hay pequeñas discrepancias con el esquema) versus un "modo estricto" (que rechace cualquier dato que no valide perfectamente) puede ser útil, dependiendo de los requisitos de robustez y flexibilidad del sistema.

6.4. Localización y Actualización de Definiciones de Lexicon

Actualmente, las definiciones canónicas de los Lexicons se encuentran versionadas como archivos JSON en el repositorio bluesky-social/atproto en GitHub.[1] Para la integración inicial, el "Sistema CORE" probablemente necesitará empaquetar estas definiciones conocidas o implementar un mecanismo para obtenerlas y actualizarlas periódicamente desde esta fuente.

Es importante notar que existen discusiones sobre la evolución de cómo se publican y resuelven los Lexicons. En el futuro, los esquemas Lexicon podrían publicarse como registros dentro de repositorios de AT Protocol, lo que permitiría una resolución más dinámica y descentralizada de los esquemas directamente a través del protocolo.[8] Aunque esto representa una visión a más largo plazo, diseñar el "Sistema CORE" con una capa de abstracción para la carga y gestión de esquemas podría facilitar la adaptación a tales cambios futuros. Esta capa podría, por ahora, cargar archivos JSON desde una ubicación conocida, y más adelante, interactuar con la red AT Protocol para resolver esquemas.

7. Conclusión

El análisis de los Lexicons de AT Protocol, especialmente aquellos que emanan del firehose de Bluesky, revela un sistema de tipado de datos estructurado y extensible, diseñado para la interoperabilidad en un entorno descentralizado. La comprensión detallada de Lexicons clave como com.atproto.repo.strongRef, app.bsky.feed.post, app.bsky.feed.like, app.bsky.graph.follow, app.bsky.actor.profile, y los diversos esquemas de incrustación (app.bsky.embed.*), es indispensable para la integración efectiva con el "Sistema CORE".

Patrones recurrentes, como el uso del campo $type para la discriminación de uniones, la referencia inmutable mediante com.atproto.repo.strongRef, y la distinción entre referencias a contenido y referencias a actores (DIDs), son fundamentales para el correcto procesamiento de los datos. La evolución de los esquemas, como la transición de entities a facets en app.bsky.feed.post, y la naturaleza de las "uniones abiertas", subrayan la necesidad de que el "Sistema CORE" sea robusto y adaptable.

Para una integración exitosa, el "Sistema CORE" debe:

1. Implementar un parseo y validación rigurosos de los registros basados en su $type y el Lexicon correspondiente.
2. Manejar correctamente las diversas operaciones de repositorio (create, update, delete) que aparecen en el firehose.
3. Desarrollar estrategias para la evolución de los esquemas y el manejo de tipos desconocidos en uniones abiertas.
4. Establecer un mecanismo para la obtención y actualización de las definiciones de Lexicon.

La documentación estructurada presentada en este informe, incluyendo el análisis detallado de los campos, tipos de datos, formatos y restricciones de cada Lexicon relevante, tiene como objetivo proporcionar una base sólida para que el equipo de desarrollo del "Sistema CORE" pueda acometer estas tareas con un entendimiento profundo de las estructuras de datos con las que interactuará. La correcta interpretación de estos esquemas permitirá al "Sistema CORE" no solo consumir datos de la red Bluesky, sino también participar potencialmente en el ecosistema AT Protocol de manera más amplia y significativa.

Obras citadas

1. bluesky-social/atproto: Social networking technology created by Bluesky - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto
2. atproto/api v0.14.0 release notes - Bluesky Documentation, fecha de acceso: mayo 22, 2025, https://docs.bsky.app/blog/api-v0-14-0-release-notes
3. bluesky-social/jetstream: A simplified JSON event stream for AT Proto - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/jetstream
4. atproto/lexicons/app/bsky/feed/repost.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/feed/repost.json
5. An ATProtocol community Lexicon - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/lexicon-community/lexicon

6. ATProtoKit/API_GUIDELINES.md at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/MasterJ93/ATProtoKit/blob/main/API_GUIDELINES.md
7. atproto.dart/website/docs/packages/atproto.md at main · myConsciousness/atproto.dart · GitHub, fecha de acceso: mayo 22, 2025, https://github.com/myConsciousness/atproto.dart/blob/main/website/docs/packages/atproto.md
8. RFC: Lexicon Resolution · bluesky-social atproto · Discussion #3074 - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/discussions/3074
9. atproto/lexicons/com.atproto.repo.strongRef.json at main · MarshalX/atproto - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/MarshalX/atproto/blob/main/lexicons/com.atproto.repo.strongRef.json
10. atproto/lexicons/app/bsky/feed/like.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/feed/like.json
11. atproto/lexicons/app/bsky/feed/post.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/feed/post.json
12. Setting a post thumb to a string instead of a blob ref permanently disables the user account · Issue #1089 · bluesky-social/atproto - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/issues/1089
13. atproto/lexicons/app/bsky/graph/follow.json at main · bluesky-social ..., fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/graph/follow.json
14. atproto/lexicons/app/bsky/actor/profile.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/actor/profile.json
15. atproto/lexicons/app/bsky/graph/block.json at main · bluesky-social ..., fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/graph/block.json
16. atproto/lexicons/app/bsky/embed/images.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/embed/images.json
17. Creating Records on Bluesky Social, fecha de acceso: mayo 22, 2025, https://christopherkenny.r-universe.dev/articles/bskyr/creating_records.html
18. atproto/lexicons/app/bsky/richtext/facet.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/richtext/fa

cet.json
19. atproto/lexicons/app/bsky/feed/getTimeline.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/feed/getTimeline.json
20. atproto/lexicons/app/bsky/actor/getProfiles.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/actor/getProfiles.json
21. atproto/lexicons/app/bsky/feed/getLikes.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/feed/getLikes.json
22. atproto/lexicons/app/bsky/graph/getFollowers.json at main - GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/graph/getFollowers.json
23. How to extend app.bsky.feed.post? · bluesky-social atproto · Discussion #3523 · GitHub, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/discussions/3523
24. github.com, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/embed/external.json
25. github.com, fecha de acceso: mayo 22, 2025, https://github.com/bluesky-social/atproto/blob/main/lexicons/app/bsky/embed/record.json