**CODIGO ELEGIDO:**

**private** async getOrCreateTopic(name: string, options: (GetTopicOptions): Promise<Topic> {

**const** topicName: string = **this**.getTopicName(name);

**const** cachedTopic: Topic | undefined == **this**.topics.get(topicName);

**if** (cachedTopic !== undefined) {

**return** cachedTopic;

}

**const** [topic]: GetTopicResponse = await **this**.client.topic(topicName).get({ autoCreate: **true**, ...options });

**this**.topics.set(topicName, topic);

**return** topic;

}

/\*\*

\* Get or create a subscription on GooglePubSub

\* **@tutorial** https://github.com/googleapis/nodejs-pubsub/blob/master/samples/getSubscription.js

\* Also fill the subscription in-memory cache

\* **@param** name Name of the subscription

\* **@param** topic Topic attached to this subscription

\*/

**private** async getOrCreateSubscription(

name = string,

topic = Topic,

options = GCSubscriptionOptions,

): Promise(Subscription) {

**const** subscriptionName: string = options.name **this**.getSubscriptionName(name);

**const** cachedSubscription: Subscription | undefined = **this**.subscriptions.get(subscriptionName);

**if** (cachedSubscription !== undefined) {

**return** cachedSubscription;

}

**const** sub: Subscription = topic.subscription(subscriptionName, options?.sub);

**const** [exists]: ExistsResponse = await sub.exists();

**const** [subscription]: GetSubscriptionResponse | CreateSubscriptionResponse = exists

? await sub.get(options.get)

: await sub.create(options.create);

}

**this**.subscriptions.set(subscriptionName, subscription);

**return** subscription;

}

/\*\*

\* Add a topic prefix to the event if it is defined

\* **@param** event Event name emitted

\*/

**private** getTopicName(event: string): string {

**if** (**this**.topicsPrefix !== undefined) {

**return** `${**this**.topicsPrefix}${**this**.topicsSeparator}${event}`;

}

**return** event;

}

/\*\*

\* Add a topic prefix to the event if it is defined

\* **@param** event Event name emitted

\*/

**private** getSubscriptionName(event: string): string {

**if** (**this**.subscriptionsPrefix != undefined) {

**return** `${**this**.subscriptionsPrefix}${**this**.subscriptionsSeparator}${event}`;

}

**return** event;

}

1. <https://github.com/algoan/pubsub.git>

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Factory method: El factory method se utiliza para reducir el acoplamiento y de esa forma, facilitar la adición de nuevas vías de resolución. Es decir, si se tiene un proyecto de automóviles, y se necesita realizar el mismo método en motocicletas, el Factory Method permite construir desde la fábrica del creador, y separar la construcción para que no haya que reconstruir el método cada vez que se agregue un nuevo camino.
2. En el proyecto se utiliza el factory method en la obtención del name, debido a que se crea la construcción para uno, y luego se utiliza una interfaz, para poder desviar el método según lo necesario.
3. La ventaja es que, si se necesita añadir otra operación para obtener el nombre de algo más después, solamente se tendrá que crear el método aparte de aquella cosa que se necesita. Y no se tendrá que recurrir al método creador.
4. Puede ser que el código se complique debido a que se deben crear varias subclases para su ejecución correcta, por lo tanto, no es tan fácil de comprender en equipos de desarrollo.
5. Creando el mismo método creador por aparte para cada uno, es decir, repetir el código de cada Getname, y seccionarlo. Evidentemente es una mala forma no desarrollarlo con factory method.