A progressive Node.js framework for building efficient and scalable server-side applications.

Description

Nest framework TypeScript starter repository.

Installation

```
$ npm install
```

Running the app

```
# development
$ npm run start

# watch mode
$ npm run start:dev

# production mode
$ npm run start:prod
```

Test

```
# unit tests
$ npm run test

# e2e tests
$ npm run test:e2e

# test coverage
$ npm run test:cov
```

WhatsApp Business

Crear y administrar plantillas

Las plantillas se usan al enviar mensajes de plantilla mediante la API de nube, alojada en Meta, o la API local. La API de nube revisa las plantillas y los parámetros de las variables mediante aprendizaje automático para proteger la seguridad y la integridad de los servicios de la API de nube. Cuando la API de nube revisa las plantillas y el texto de las variables, no se comparte información con WhatsApp.

Las plantillas se pueden crear con la **API de administración de WhatsApp Business** (nos centraremos en esta opcion) o el Administrador de WhatsApp Business.

Numero de plantillas que puede tener una cuenta de WhatsApp Business:

- Empresa principal verificada: 6000
- Empresa no verificada: 250

Crear plantillas

Envía una solicitud **POST** al extremo Cuenta de WhatsApp Business > Plantillas de mensajes para crear una plantilla.

Sintaxis de la solicitud:

```
POST /<WHATSAPP_BUSINESS_ACCOUNT_ID>/message_templates
```

Cuerpo de la solicitud:

La solicitud se debe hacer a la siguiente url:

`https://graph.facebook.com/v19.0/WHATSAPP_BUSINESS_ACCOUNT_ID/message_templates`

```
{
  "name": "<NAME>",
  "category": "<CATEGORY>",
  "allow_category_change": <ALLOW_CATEGORY_CHANGE>,
  "language": "<LANGUAGE>",
  "components": [<COMPONENTS>]
}
```

Para saber como son los tipos de los parametros que hay que mandar en la peticion, consultar con la documentacion oficial.

el parametro `allow_category_change` es opcional pero se aconseja ponerlo por defecto a `true`. Explicacion:

Se establece en verdadero para permitirnos asignar automáticamente una categoría. Si se omite, es posible que la plantilla se rechace a causa de una categorización incorrecta.

Componentes de plantilla

Las plantillas están formadas por varios componentes de texto, de contenido multimedia e interactivos en función de las necesidades de la empresa. Consulta el documento Componentes de las plantillas para obtener una lista de todos los componentes posibles y sus requisitos, así como muestras y consultas de ejemplo.

Al crear una plantilla, debes definir sus componentes; para ello, asigna una matriz de objetos de componentes a la propiedad “components” en el cuerpo de la solicitud.

Por ejemplo, a continuación se muestra una matriz que contiene un componente de cuerpo de texto con dos variables y valores de ejemplo, un componente de botón de número de teléfono y un componente de botón de URL:

```
[
  {
    "type": "BODY",
    "text": "Thank you for your order, {{1}}! Your confirmation number is {{2}}.
If you have any questions, please use the buttons below to contact support. Thank
you for being a customer!",
    "example": {
      "body_text": [
        [
          "Pablo","860198-230332"
        ]
      ]
    }
  },
  {
    "type": "BUTTONS",
    "buttons": [
      {
        "type": "PHONE_NUMBER",
        "text": "Call",
        "phone_number": "15550051310"
      },
      {
        "type": "URL",
        "text": "Contact Support",
        "url": "https://www.luckyshrub.com/support"
      }
    ]
  }
]
```

Validación de la categoría

Al enviar una solicitud de creación de plantillas, validamos inmediatamente su categoría según nuestras normas para la categorización de plantillas.

Si estamos de acuerdo con la categoría que has designado, creamos la plantilla y establecemos el valor de status en PENDING. A continuación, la plantilla se somete a una revisión. Si no estamos de acuerdo con tu designación, creamos la plantilla, pero establecemos el valor de status en REJECTED y activamos un webhook de actualización de estado de plantilla de mensaje con el valor de reason establecido en INCORRECT_CATEGORY. Te recomendamos que escuches este webhook para identificar plantillas rechazadas o solicitar el campo rejected_reason en las plantillas recién creadas, que tendrán el valor TAG_CONTENT_MISMATCH. En ambos casos, el estado inicial de la plantilla se devuelve como parte de la respuesta de la API.

Si el estado de la plantilla se ha establecido en REJECTED como parte de la validación de la categoría, tienes varias opciones:

Edita los componentes de la plantilla para que se ajusten a nuestras normas. Edita la categoría de la plantilla para que se ajuste a nuestras normas. Crea una nueva plantilla.

Ejemplos

Existen multitud de posibilidades. A continuacion se ve un ejemplo de peticion de creacion de una platilla segun la documentacion oficial:

Solicitud para crear una plantilla de promoción estacional compuesta por los siguientes componentes:

- un encabezado de texto
- un cuerpo de texto
- un pie de página
- dos botones de respuesta rápida

Para obtener ejemplos adicionales, consulta Ejemplos de solicitudes.

```
curl 'https://graph.facebook.com/v19.0/102290129340398/message_templates' \
-H 'Authorization: Bearer EAAJB...' \
-H 'Content-Type: application/json' \
-d '
{
  "name": "seasonal_promotion",
  "language": "en_US",
  "category": "MARKETING",
  "components": [
    {
      "type": "HEADER",
      "format": "TEXT",
      "text": "Our {{1}} is on!",
      "example": {
        "header_text": [
          "Summer Sale"
        ]
      }
    },
```

```
    {
      "type": "BODY",
      "text": "Shop now through {{1}} and use code {{2}} to get {{3}} off of all
merchandise.",
      "example": {
        "body_text": [
          [
            "the end of August","25OFF","25%"
          ]
        ]
      }
    },
    {
      "type": "FOOTER",
      "text": "Use the buttons below to manage your marketing subscriptions"
    },
    {
      "type":"BUTTONS",
      "buttons": [
        {
          "type": "QUICK_REPLY",
          "text": "Unsubscribe from Promos"
        },
        {
          "type":"QUICK_REPLY",
          "text": "Unsubscribe from All"
        }
      ]
    }
  ]
}'
```

Ejemplo de respuesta:

```
{
    "id": "572279198452421",
    "status": "PENDING",
    "category": "MARKETING"
}
```

Dentro de esta seccion de ejemplos yo Sergio Radigales como desarrollador voy a dejar un **ejemplo sencillo de como crear un endpoint HTTP con Nest.js** (aunque se podria hacer con cualquier otro framework) y de los datos que necesitaremos para que esta se lleve a cabo satisfactoriamente:

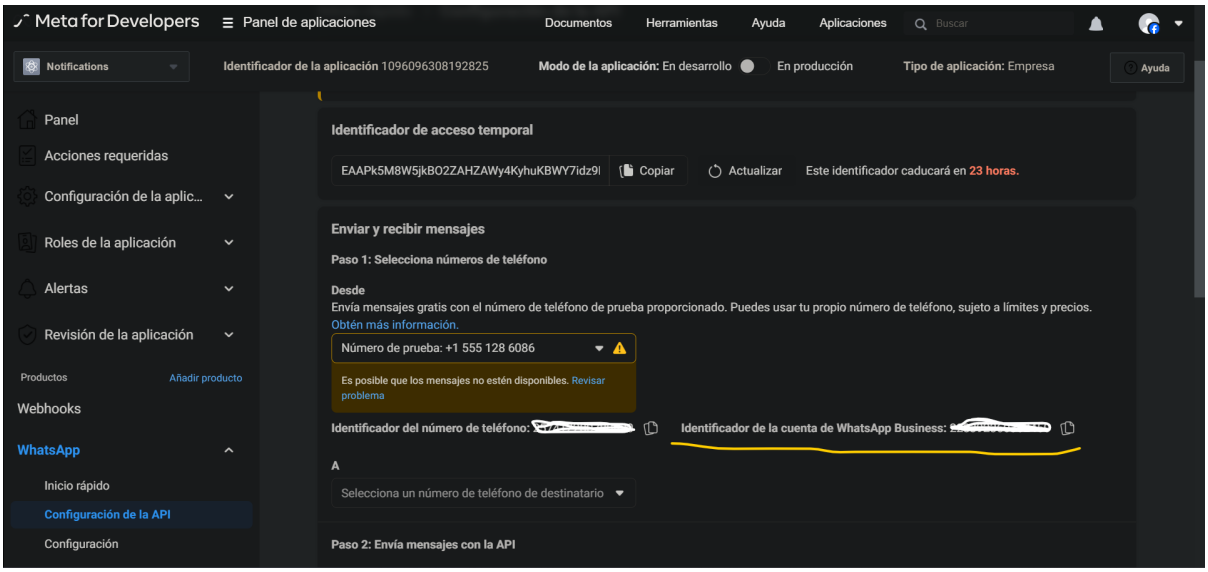
service.ts

```
public async createTemplate(
    name: string,
    category: TemplateCategory,
```

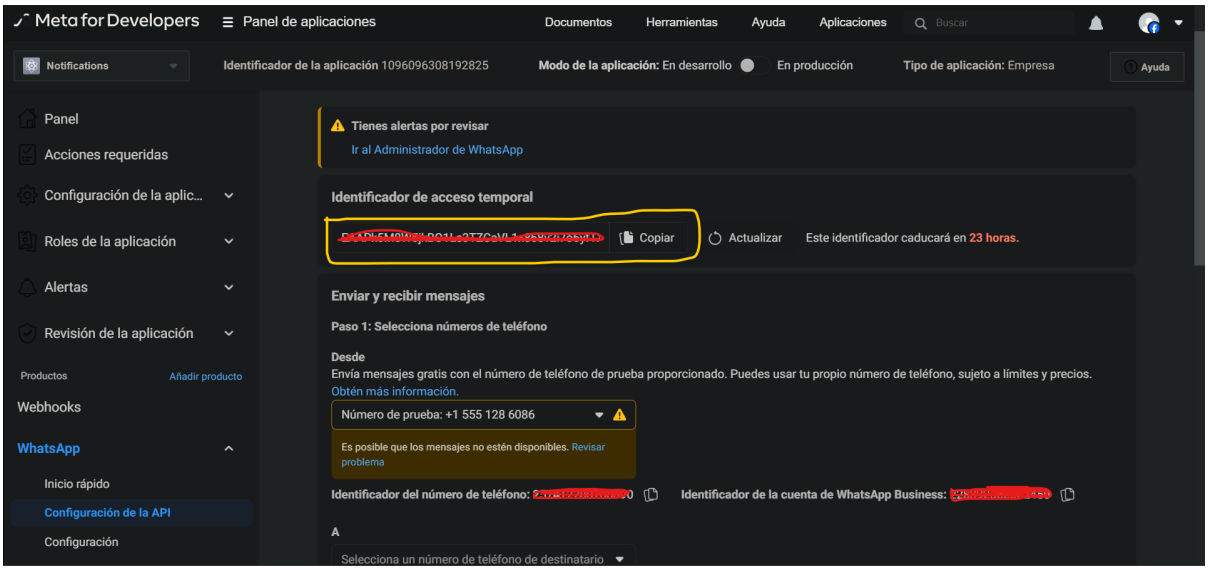
```
    isHeader: boolean,
    isBody: boolean,
    isFooter: boolean,
    headerText?: string,
    bodyText?: string,
    footerText?: string,
  ) {
    try {
      const response = await fetch(this.createTemplateUrl, {
        method: "POST",
        headers: this.headers,
        body: JSON.stringify({
          name,
          category,
          allow_category_change: true,
          language: "es",
          components: buildTemplatesComponents(
            isHeader,
            isBody,
            isFooter,
            headerText,
            bodyText,
            footerText,
          ).components,
        }),
      });
      return await response.json();
    } catch (error) {
      throw error;
    }
  }
```

Datos a tener en cuenta:

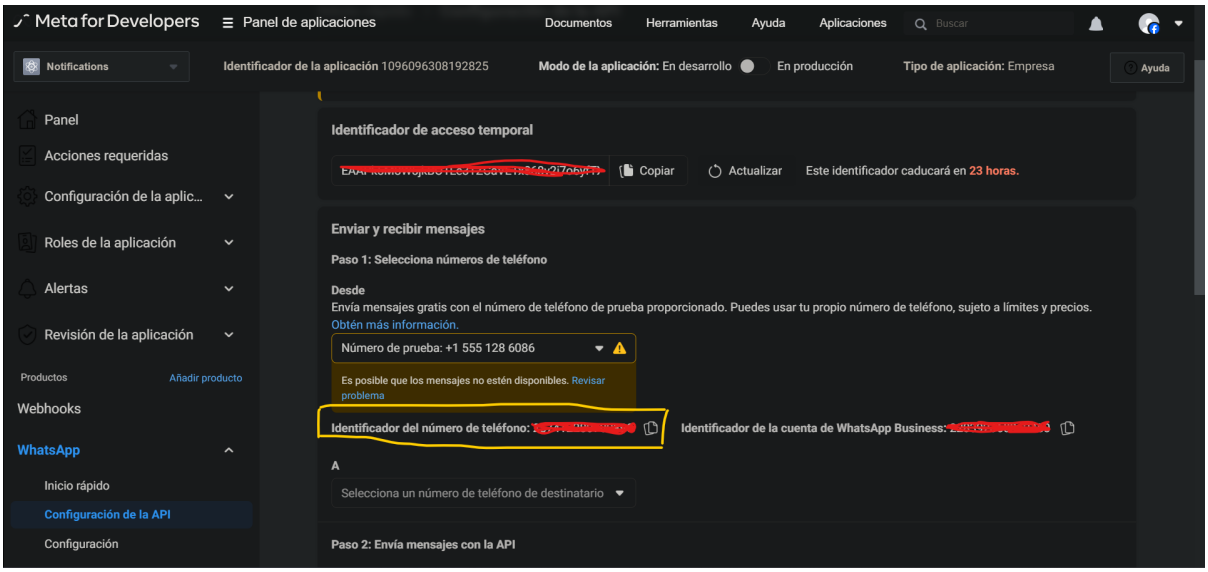
- Variables de entorno:
  - Relacionadas con la creacion de plantillas:
    - WHATSAPP_BUSINESS_ACCOUNT_ID. Este valor lo podemos encontrar en el panel de aplicaciones de Meta



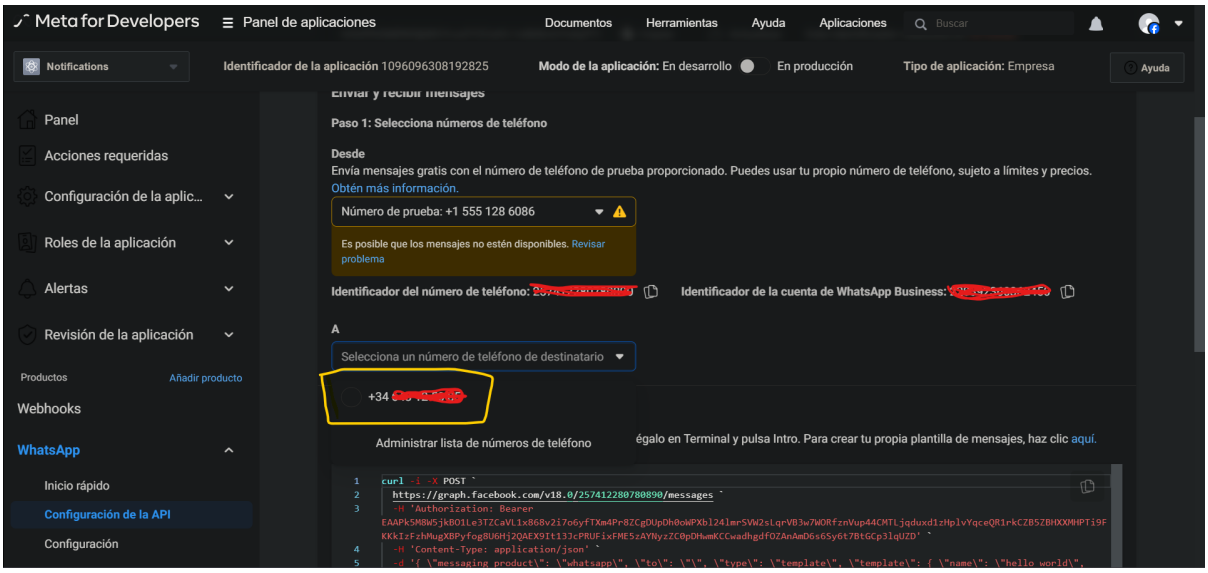
- Relacionadas con el envio de mensajes a clientes:
  - TEMPORAL_TOKEN: Este es un token que en las aplicaciones de prueba al no tener la empresa verificada expira cada 23 horas, por lo que tendra que ser updateado cada dia mientras desarrollamos. Lo encontramos en el panel de aplicaciones de Meta



  - PHONE_NUMBER_ID: panel de aplicaciones de Meta



- DESTINATION_NUMBER: panel de aplicaciones de Meta



- Los parametros que se pasan al metodo `createTemplate` son los que yo he decido que necesito para crear mi nueva plantilla.
  - name: da el nombre a la nueva plantilla.
  - category: define la categoria de la plantilla. En este caso he creado un enum de TypeScript para limitar los valores a los aceptados por la api de administracion de WhatsApp Business.

```
export enum TemplateCategory {
  AUTHENTICATION,
  MARKETING,
  UTILITY,
}
```

  - Los demas parametros que se pasan son fruto de la funcionalidad que yo he querido implementar para la creacion de mis plantillas. Hay que tener en cuenta que existen varios componentes como son **Header, Body, Footer y Buttons**. En mi caso y a modo de prueba he querido implementar la funcionalidad que mediante una peticion HTTP se pueda crear una

plantilla que tenga o no header, body y footer. Para ello he creado una funcion que crea la matriz de componentes. A continuacion la muestro con las distintas interfaces creadas para ello:

```
import { Body, Components, Footer, Header } from "src/whatsapp/template";

export const buildTemplatesComponents = (
  isHeader: boolean,
  isBody: boolean,
  isFooter: boolean,
  headerText?: string,
  bodyText?: string,
  footerText?: string,
): Components => {
  const components: Components = { components: [] };
  if (isHeader) {
    components.components.push(buildHeader(headerText));
  }
  if (isBody) {
    components.components.push(buildBody(bodyText));
  }
  if (isFooter) {
    components.components.push(buildFooter(footerText));
  }

  return components;
};

const buildHeader = (text: string): Header => {
  return {
    type: "HEADER",
    format: "TEXT",
    text,
  };
};

const buildBody = (text: string): Body => {
  return {
    type: "BODY",
    text,
  };
};

const buildFooter = (text: string): Footer => {
  return {
    type: "FOOTER",
    text,
  };
};
```

- Interfaces

```
export interface Components {
  components: [header?: Header, body?: Body, footer?: Footer];
}

export interface Header {
  type: "HEADER";
  format: "TEXT";
  text: string;
}

export interface Body {
  type: "BODY";
  text: string;
}
export interface Footer {
  type: "FOOTER";
  text: string;
}
```

controler.ts

```
@Post("create-template")
  async createTemplate(
    @Body("name")
    name: string,
    @Body("category")
    category: TemplateCategory,
    @Body("isHeader")
    isHeader: boolean,
    @Body("isBody")
    isBody: boolean,
    @Body("isFooter")
    isFooter: boolean,
    @Body("headerText")
    headerText?: string,
    @Body("bodyText")
    bodyText?: string,
    @Body("footerText")
    footerText?: string,
  ) {
    return await this.service.createTemplate(
      name,
      category,
      isHeader,
      isBody,
      isFooter,
      headerText,
      bodyText,
      footerText,
    );
  }
```