Twilio-python

Arquitectura e Integración de Sistemas Software

Enero 2023

David Guillén Fernández (davguifer@alum.us.es)
Javier Gutiérrez Pastor(javgutpas@alum.us.es)
Javier Lozano Gómez (javlozgom@alum.us.es)
Rafael Molina García (rafmolgar2@alum.us.es)

Tutor: Irene Bedilia Estrada Torres
Número de grupo: 2

Link del Proyecto en Github: https://github.com/twilio/twilio-python

Link del proyecto en SonarCloud:

https://sonarcloud.io/project/overview?id=twilio_twilio-python

HISTORIAL DE VERSIONES

Date	Version	Description	Participants
23/02/2023	1.0	-Versión inicial	D.Guillén J.Gutiérrez J.Lozano R.Molina

Índice

1 Introducción / Introduction

El proyecto "twilio-python" es un módulo de Python que sirve para comunicarse con la API de Twilio y generar TwiML.

Twilio es una de las principales plataformas de comunicación en la nube que permite a las empresas agregar capacidades de comunicación como SMS, voz, video, correo electrónico,WhatsApp y más.

Para ello se usa TwiML (Twilio Markup Language) que es un lenguaje utilizado por la plataforma de comunicaciones en la nube Twilio. Es un conjunto de etiquetas XML que se utilizan para definir y controlar la interacción entre las aplicaciones de Twilio y los dispositivos de comunicación, como teléfonos móviles y teléfonos de escritorio.

Con este módulo, los usuarios pueden enviar y recibir mensajes de texto, hacer y recibir llamadas telefónicas y realizar otras funciones de comunicación a través de la API de Twilio.

El módulo de "twilio-python" es de código abierto y está disponible en GitHub, lo que significa que los desarrolladores pueden contribuir al código fuente y mejorar la funcionalidad del módulo. Además, es compatible con versiones de Python de la 3.6 a la 3.11, lo que la hace accesible para una amplia gama de proyectos de Python.

2 Visión general / Overview

El proyecto de Twilio Python es un módulo de Python que permite a los desarrolladores interactuar con la API de Twilio, que es una plataforma de comunicaciones en la nube que permite enviar y recibir mensajes de texto, llamadas y otros medios a través de su plataforma.

En términos generales, el proyecto consta de tres bloques funcionales principales:

- Autenticación y manejo de credenciales
- Interacción con la API de Twilio
- Manejo de respuestas y errores

A continuación, se describen con más detalle cada uno de estos bloques, junto con sus entradas y salidas:

1. **Autenticación y manejo de credenciales:**

El bloque de autenticación maneja la verificación de las credenciales de autenticación del usuario y crea una instancia de Client para que el usuario pueda interactuar con la API de Twilio. También proporciona una forma para que los usuarios actualicen o cambien sus credenciales de autenticación.

- **Entradas:** Las credenciales de autenticación proporcionadas por el usuario, que incluyen un identificador de cuenta de Twilio y un token de autenticación. Estas credenciales se utilizan para autenticar las solicitudes a la API de Twilio.
- **Salidas:** Una instancia de la clase Client que se utiliza para realizar solicitudes a la API de Twilio.

En el proyecto, la autenticación y manejo de credenciales se encuentra en el archivo ***'twilio/rest/__init__.py'***.

2. **Interacción con la API de Twilio:**

Este bloque de funciones es el responsable de enviar y recibir mensajes de texto, realizar y recibir llamadas, y otras interacciones con la API de Twilio. Las funciones se dividen en sub-bloques que se centran en la funcionalidad de los diferentes servicios que Twilio ofrece, como el envío de mensajes de texto o la realización de llamadas.

- **Entradas:** Una instancia de Client creada a partir del bloque de autenticación, y cualquier parámetro adicional requerido por las diferentes funciones de la API

de Twilio. Estos parámetros varían según la función, pero pueden incluir, por ejemplo, el número de teléfono de origen, el número de teléfono de destino y el mensaje de texto a enviar.

- **Salidas:** Una respuesta de la API de Twilio, que puede incluir un ID de mensaje para confirmar que se ha enviado un mensaje de texto correctamente, o información sobre llamadas entrantes o salientes.

En el proyecto, ***'twilio/rest/__init__.py'*** define la estructura de los módulos que interactúan con la API de Twilio, ***'twilio/rest/api/v2010/account/message.py'*** incluye métodos para enviar, obtener y eliminar mensajes y ***'twilio/rest/api/v2010/account/call.py'*** incluye métodos para realizar, obtener y eliminar llamadas.

3. Manejo de respuestas y errores:

Este bloque maneja las respuestas de la API de Twilio y las transforma en un formato que sea más fácil de manejar para el usuario. También maneja los errores de la API de Twilio, y proporciona información sobre cómo solucionar problemas comunes.

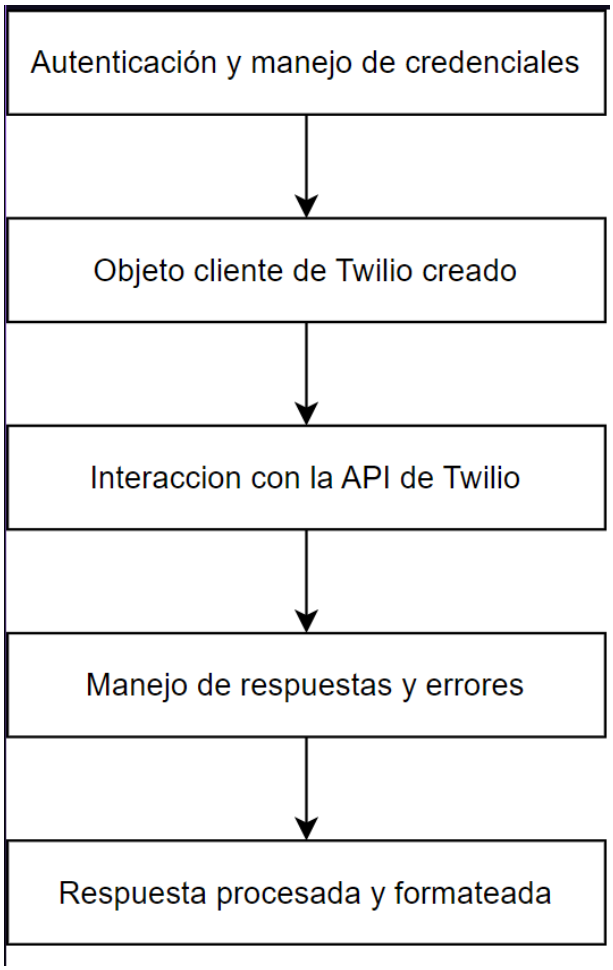
- **Entradas:** Una respuesta de la API de Twilio generada por el bloque de interacción con la API.
- **Salidas:** Una respuesta procesada y formateada para que sea más fácil de interpretar por el usuario.

En el proyecto, ***'twilio/rest/__init__.py'*** incluye métodos para realizar peticiones HTTP a la API de Twilio y manejar las respuestas y errores que se reciben, ***'twilio/rest/api/v2010/account/message.py'*** incluye métodos para enviar, obtener y eliminar mensajes, y maneja las respuestas y errores que se reciben de la API de Twilio y ***'twilio/exceptions.py'*** que se utiliza para manejar los errores que se producen al interactuar con la API de Twilio.

La secuencia en la que se realizan estas funciones es la siguiente:

1. El usuario proporciona sus credenciales de autenticación, que se verifican y se utilizan para crear un objeto cliente de Twilio.
2. El usuario llama a las funciones en el objeto cliente de Twilio para interactuar con la API de Twilio, proporcionando los parámetros necesarios para cada función.
3. La API de Twilio responde con una respuesta, que se maneja y procesa para que sea más fácil de interpretar por el usuario.

Si se produce un error, se maneja y se proporciona información adicional para ayudar a resolver el problema.


Autenticación y manejo de credenciales
Objeto cliente de Twilio creado
Interaccion con la API de Twilio
Manejo de respuestas y errores
Respuesta procesada y formateada

3 Participantes / Stakeholders

Debido a que el proyecto ha sido creado y mantenido por la comunidad, vamos a tener numerosos roles:

Desarrolladores: Son los responsables del desarrollo del módulo twilio-python.

De acuerdo con las estadísticas del proyecto, tenemos cuatro desarrolladores principales “jingming”,“codejudas”,”skimbrel”,”kyleconroy”, quienes son los que han aportado más al proyecto, aunque hay más desarrolladores secundarios.

Testers: La función de este grupo es detectar problemas y crear o modificar casos de prueba para solucionarlos. Los que integran este equipo son los desarrolladores, otros contribuyentes al proyecto e incluso la propia comunidad de Python.

Soporte: Brindan soporte técnico sobre el módulo Twilio Python para que funcione correctamente. De nuevo son los propios desarrolladores los que aportan esta función de soporte, en especial los desarrolladores principales.

Usuarios finales: Aquellos usuarios que utilizan el módulo Twilio Python para enviar y recibir mensajes de texto y llamadas telefónicas a través de Twilio.

Soporte de Twilio: Aportan un interés indirecto hacia el proyecto, ya que mantienen la aplicacion Twilio libre de errores y funcionando.

Otros desarrolladores: Este módulo puede usarse también para ser incluído en otros proyectos realizados por la comunidad desarrolladora de Python.

Mencionar que el total de contribuyentes es de 103 personas además de algunos bots de Twilio.

4 Vistas / Views

La arquitectura de Twilio-Python puede describirse con las siguientes vistas.

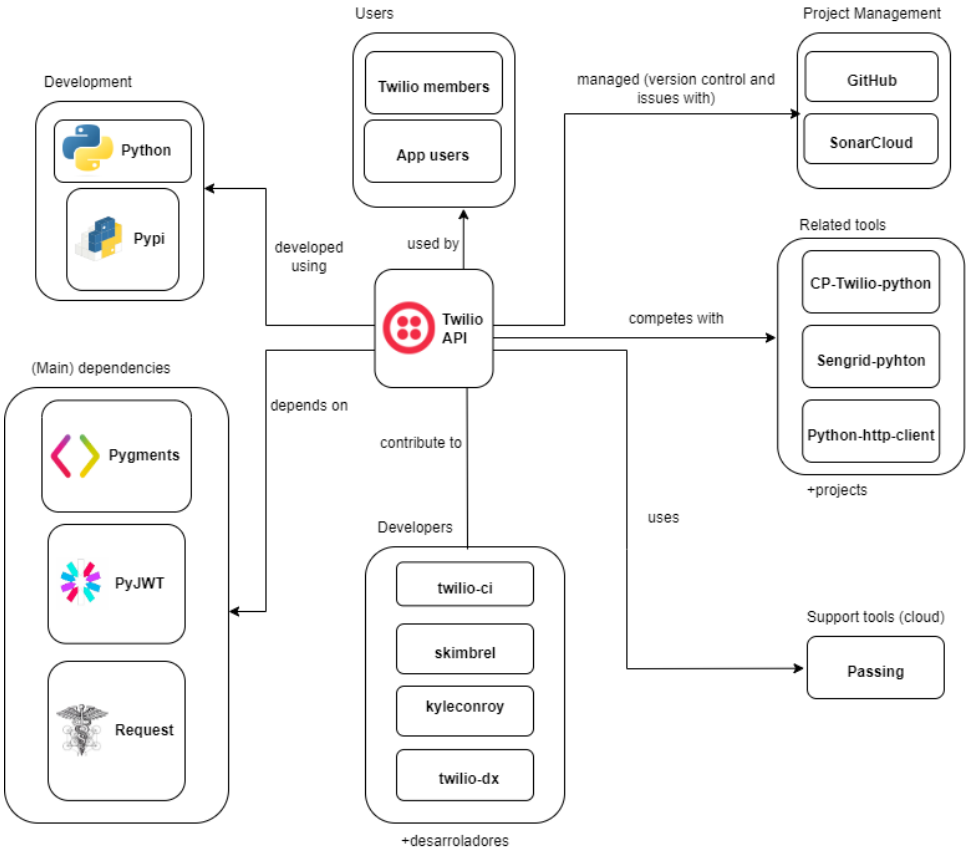
4.1 Vista de context / Context view

Esta vista define la relación y dependencias entre el sistema y su entorno, que incluye cómo el sistema interactúa con otros sistemas, organizaciones y personas.

La siguiente imagen muestra el diagrama de contexto de Twilio-python. El proyecto se encuentra alojado en GitHub para el control de versiones, seguimiento de problemas y para alojar la documentación del proyecto. El proyecto está asociado con servicios en la nube como SonarCloud para el seguimiento de la calidad del código. Los principales desarrolladores son personas como Kyleconroy y Skimbrell. Esta herramienta es principalmente usada por empresas como forma de comunicación.

Este proyecto está completamente escrito en Python y usa Pypi que nos ayuda a agregar nuevas librerías. Las principales dependencias del proyecto son la librería Pygments que se encarga de embellecer el código, la librería PyJWT que nos permite cifrar y descifrar JSON Web Tokens y la librería Requests que nos facilita el trabajo con las peticiones http.

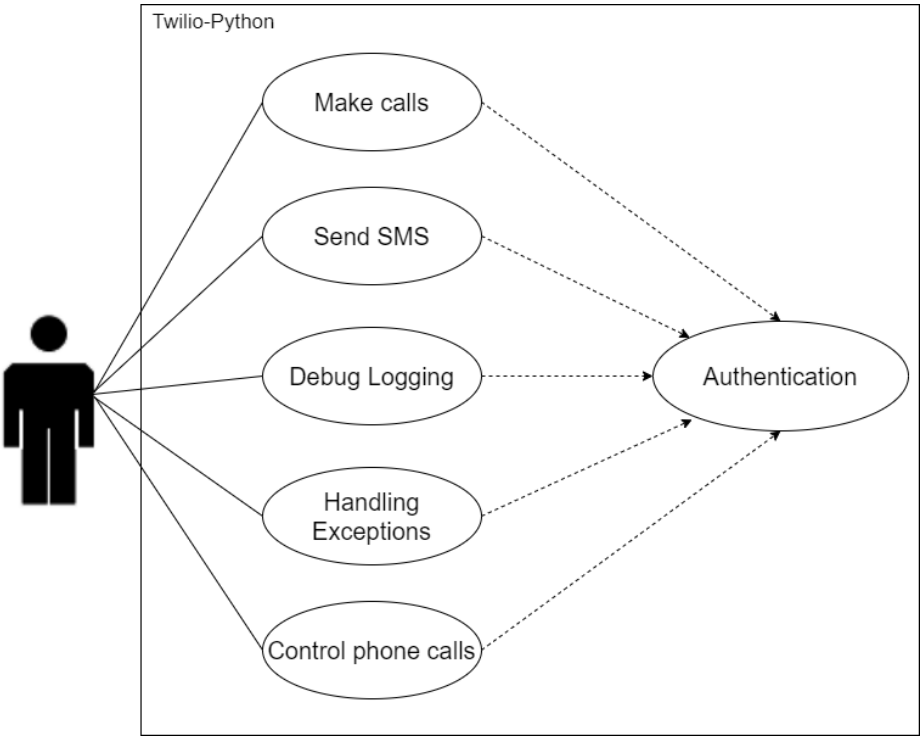
También, existen algunas herramientas relacionadas como *CP-Twilio-Python-Text-App*, *sendgrid-python* y *python-http-client*.



4.2 Escenarios de uso / Scenarios view

Identificamos los siguientes escenarios de uso:

- **Hacer llamadas:** Twilio-Python permite hacer y recibir llamadas usando el servicio de Twilio. Muy útil para integrar llamadas telefónicas.
- **Enviar SMS:** Twilio-Python permite enviar mensajes de texto utilizando los servicios de Twilio. Útil para enviar notificaciones y/o alertas a los usuarios de la aplicacion.
- **Habilitar el registro de depuración:** Twilio-Python permite activar el registro de depuración lo que permite obtener información sobre los errores del administrador de licencias.
- **Manejar excepciones:** Twilio-Python permite el manejo de excepciones de manera que la aplicacion genere mensajes para informar al usuario de las excepciones.
- **Control de llamadas:** Twilio-Python permite el control de llamadas con TwiML. Lo que permite implementar a la aplicacion respuestas por voz.



4.3 Vista functional / Functional view

La vista funcional de Twilio-Python es la de un módulo que permite la integración de la plataforma Twilio en aplicaciones Python, lo que significa que se puede utilizar para enviar y recibir mensajes de texto (SMS), mensajes multimedia (MMS), realizar y recibir llamadas telefónicas, enviar y recibir faxes, y crear respuestas de voz personalizadas en aplicaciones Python.

Los componentes principales de Twilio-Python son:

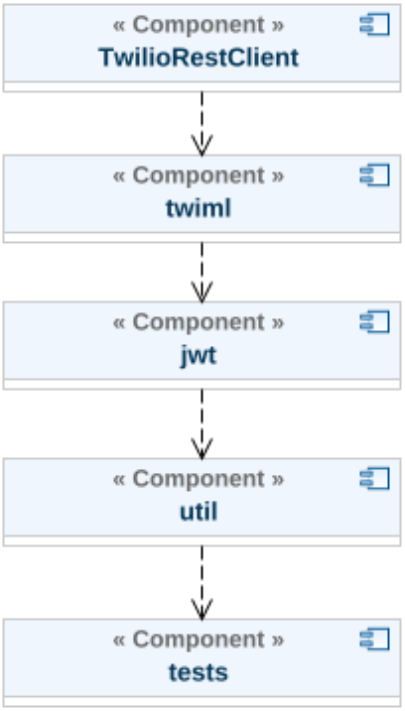
Cliente Twilio (TwilioRestClient): Es el objeto principal del módulo que se utiliza para autenticar y hacer llamadas a la API de Twilio.

Respuestas TwiML (twiml): Esta funcionalidad permite crear respuestas personalizadas de voz y SMS utilizando el lenguaje de marcado TwiML.

JWT (jwt): El módulo Twilio-Python proporciona funciones para crear y verificar JWTs que se pueden utilizar para autenticar aplicaciones en la API de Twilio.

Herramientas de utilidad (util): Twilio-Python incluye una serie de funciones y herramientas de utilidad para simplificar la integración de Twilio en aplicaciones Python.

Pruebas (tests): Twilio-Python viene con un conjunto de pruebas unitarias e integración para garantizar su correcto funcionamiento y para que los desarrolladores puedan probar y depurar sus aplicaciones.

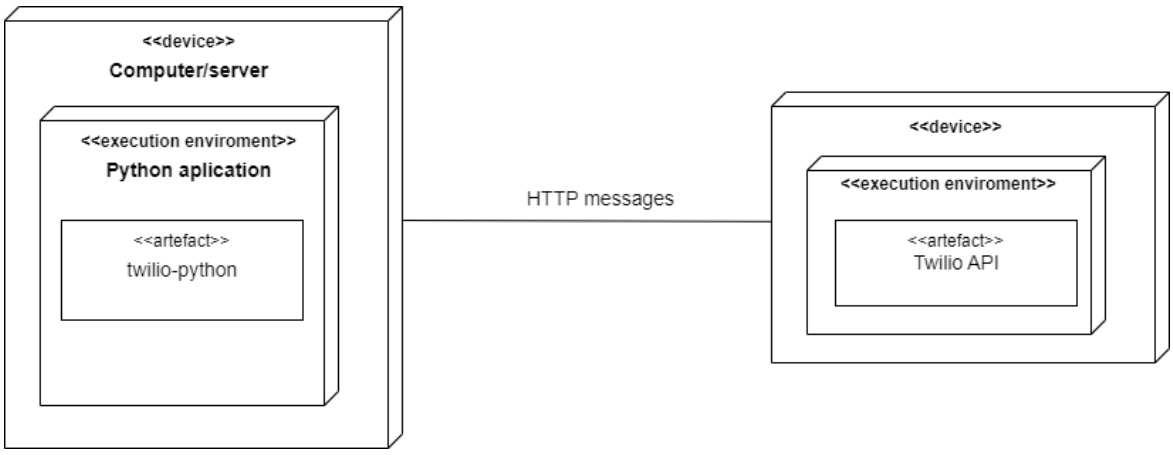


4.4 Vista de despliegue / Deployment view

La siguiente imagen representa el digrama de despligue de Twilio-python. Como se puede observar necesitaríamos tener instalado un entorno python donde poder ejecutar el módulo. Dado que la API de Twilio se encuentra en la nube, también necesitaríamos una conexión a Internet para la comunicación con la misma.

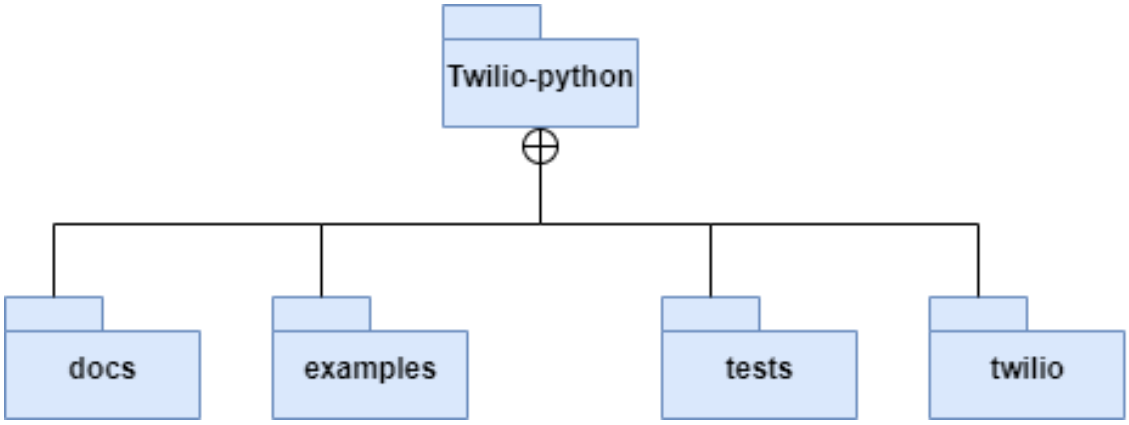
Para el correcto despliegue de la aplicación tendremos que crearnos una cuenta en la API de Twilio y posteriormente tendremos que conseguir unas credenciales de autenticación.

La comunicación entre ambos bloques se realiza mediante mensajes HTTP.



4.5 Vista de Desarrollo / Development view

A continuación, veamos los principales componentes del proyecto Twilio-Python



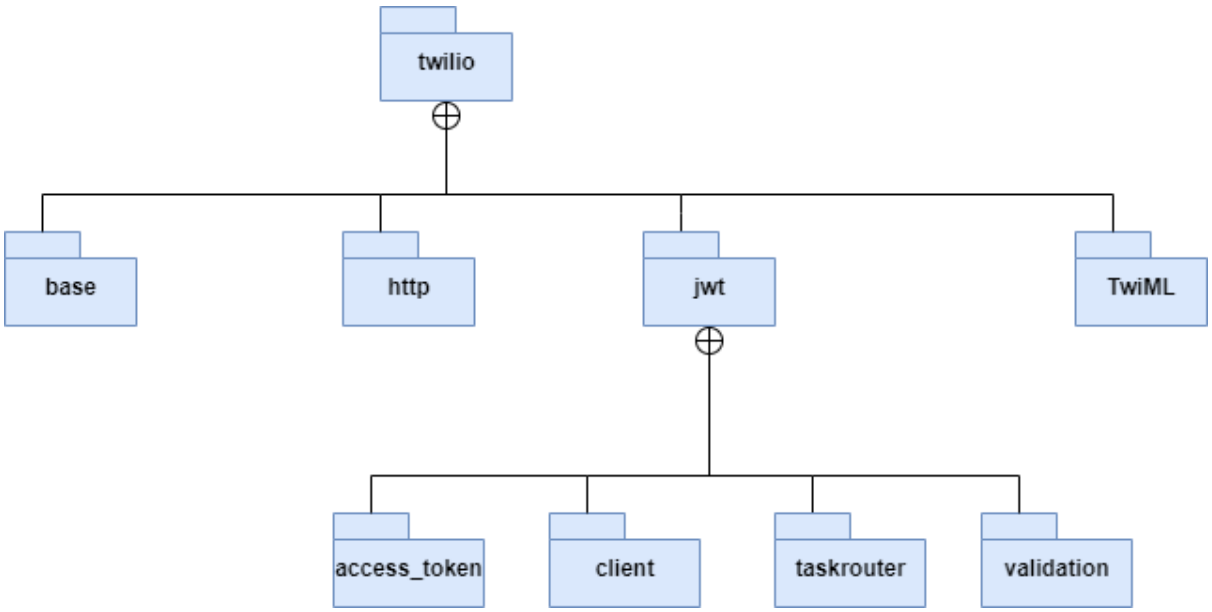
- **Docs:** Esta carpeta contiene la documentación básica de Twilio-Python, que incluye la documentación del código y guías de inicio rápido.
- **Examples:** Esta carpeta contiene ejemplos de como utilizar Twilio-Python en diferentes casos de uso.
- **Tests:** Esta carpeta contiene archivos de prueba de Twilio-Python, que incluyen casos de prueba para las clases y funciones principales, integraciones con Twilio (API) y otras pruebas de integración. Dentro de test existen las siguientes carpetas:
    - **Unit:** Esta carpeta contiene los archivos de prueba unitaria para las clases y funciones principales. Se encargan de verificar el funcionamiento de cada función y clase de manera individual.
    - **Integration:** Esta carpeta contiene pruebas de integración que verifican el funcionamiento al interactuar con Twilio (API).
- **Twilio:** Esta carpeta es la principal de Twilio-Python, contiene archivos con la funcionalidad básica del módulo, con inicializacion de paquetes, definición de funciones y clases principales,archivos para manejar la autenticación, gestión de excepciones y más.

En la siguiente tabla se ven los paquetes principales de la carpeta Twilio. El código esta distribuido en 8 paquetes, 92 clases de Python y alrededor de 2300 lineas de código.

Package name	Description	Classes	LoC
Base	Métodos base que se utilizan en el resto de paquetes	9	376

HTTP	Realiza solicitudes HTTP	5	193
Jwt	Genera tokens JSON Web para la autenticación de clientes Twilio	19	583
Twiml	Genera respuestas en XML TwiML, que controlan el flujo de comunicación	59	1123
	Total	**92**	**2275**

Diagrama UML de los paquetes en Twilio-Python:



Twilio-Python esta desarrollado en Python y utiliza pip como gestor de dependencias (base en python). Dependencias principales:

- **Requests:** Bibilioteca de Python para realizar solicitudes HTTP.
- **Pygments:** Biblioteca de Python para resaltado de sintaxis en código fuente.
- **PyJWT:** Biblioteca de Python para la generación y verificación de tokens JSON Web.

Twilio-Python tiene conexión con SonarCloud una herramienta de análisis, que brinda a los desarrolladores con análisis detallados del proyecto incluyendo vulnerabilidades, cobertura de código y métricas del código.

5 Puntos de variabilidad y extensión / Variability and extension points

Los *puntos de variabilidad* definen partes donde el sistema puede adoptar diferentes alternativas (variantes).

Identificamos los siguientes *puntos de variabilidad* de Twilio-Python:

Tipo de mensaje: El usuario puede seleccionar el tipo de mensaje que mandar (voz, texto...). Se muestra en el README del proyecto.

Excepciones: Este módulo incluye varios tipos de excepciones que pueden ocurrir al acceder a la API de Twilio (también ocurren excepciones por métodos y clases obsoletas del módulo). Los desarrolladores pueden personalizar cómo se realizan estas excepciones, como se puede ver el siguiente enlace:

https://github.com/twilio/twilio-python/blob/main/twilio/base/exceptions.py

Credenciales de la API: Twilio necesita credenciales propias para el correcto funcionamiento, este módulo permite pasarlos directamente al constructor o mediante variables de entorno. Como se puede ver en el README del proyecto.

Cliente HTTP personalizado: Este módulo nos permite, a través del mismo, utilizar un cliente HTTP personalizado. Como se puede ver en el README del proyecto.

Version de Sphinx: el módulo nos permite poner una versión mínima de Sphinx Python (software generador de documentación) en caso de que se necesite. Se muestra en el siguiente enlace:

https://github.com/twilio/twilio-python/blob/main/docs/conf.py

Los *puntos de extensión* son aquellas partes del sistema donde se espera que el sistema pueda ser extendido en un futuro.

El proyecto presenta los siguientes *puntos de extensión*:

Adición de nuevas formas de comunicación: Además de mensajes de texto y llamadas de voz, Twilio permite a los usuarios enviar y recibir mensajes de otro formato, como videos, chats de Whatsapp...El módulo debe ser extensible para poder dar soporte a estas formas de comunicación.

Opciones de personalización: El módulo debe tener la posibilidad de añadir nuevas formas de personalización, tales como poder personalizar el tipo de letra del mensaje, color de letra...

Otros métodos de identificación: La plataforma de Twilio admite numerosas opciones de autenticación, por lo que el sistema se debe poder extender para dar soporte a todas

las formas de autenticación para mejorar la experiencia del usuario y satisfacer todas sus necesidades.

6 Análisis de atributos de calidad / Analysis of quality attributes

Veamos el siguiente análisis de los atributos de calidad:

Reliability: O “fiabilidad” ,se refiere a la capacidad del software de realizar sus funciones de manera confiable, sin errores o fallos inesperados. Analizando los datos proporcionados por SonarCloud observamos que dicho atributo está calificado con “A” ya que ahora mismo no hay errores o fallos abiertos en el proyecto.

Cabe destacar que se han analizado mas de 200 archivos de forma independiente y todos ellos han sido calificados con “A”.

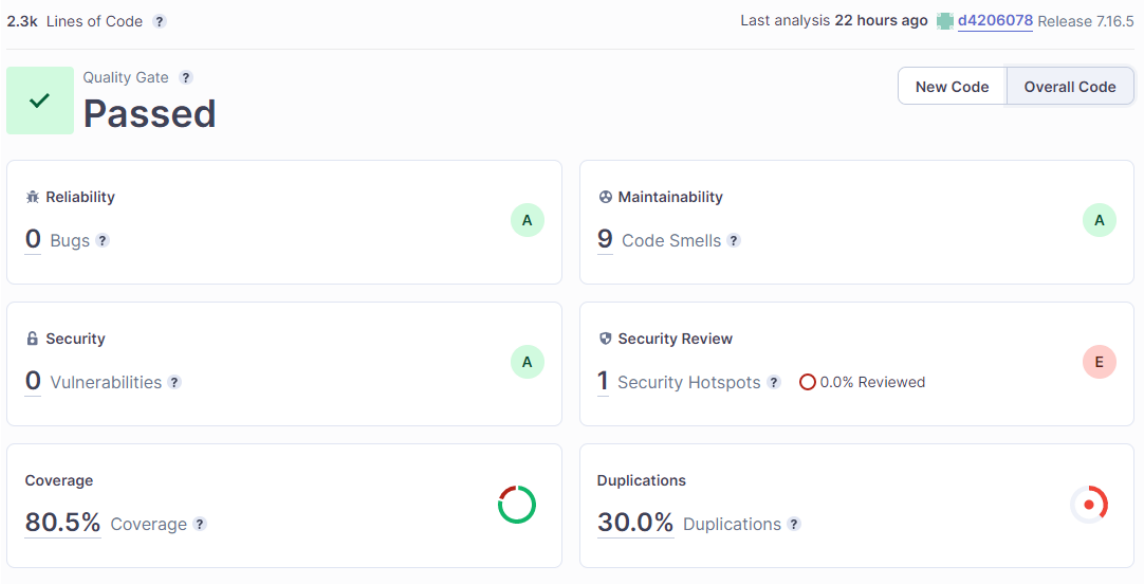
Usability: O “usabilidad”, se refiere a la facilidad con la que los usuarios pueden utilizar el software de manera efectiva, eficiente y satisfactoria. La usabilidad de Twilio-python es bastante buena, su instalación no es para nada compleja, como dijimos en apartados anteriores y basándonos en la documentación ofrecida por el proyecto, sólo necesitariamos un entorno python, crearnos un usuario y por último obtener las credenciales de autenticación. Si entramos en la página del proyecto en GitHub observamos algunos problemas relacionados con la codificación, aunque no suponen grandes fallos como para que el software deje de funcionar.

Mantenibility: O “mantenibilidad, se refiere a la capacidad del software para ser fácilmente modificado o actualizado, y a su capacidad para ser entendido y corregido por los desarrolladores. Creemos que la mantenibilidad de twilio-python es bastante buena ,echando un vistazo al código observamos una gran modularidad lo cuál facilita el mantenimiento enfocado tanto a fallos como a futuras mejoras, en SonarCloud este atributo obtiene una calificación de “A” gracias a los factores que acabamos de comentar además de que no encontramos malas prácticas como por ejemplo Code Smells.

Reusability: O “reutilización”, se refiere a la facilidad con la que se pueden reutilizar partes del sistemas. Bajo nuestro punto de vista gran parte del código de twilio-python puede ser reutilizado. Aunque la base del proyecto esté enfocada a la comunicación con la API de Twilio (un programa independiente), este no deja de ser un software basado en la comunicacion entre los usuarios del mismo, por lo tanto gran parte de su código como el diseño de chats,llamadas etc... podría ser reutilizado en la arquitectura de aplicaciones similares.

Security: O “seguridad”, se refiere a la capacidad del software para proteger los datos y recursos del sistema de posibles amenazas y ataques externos. Basandonos en los datos ofrecidos por SonarCloud podemos observar que la seguridad de twilio-python es bastante buena, calificada con “A”, aunque uno de los parámetros que se consideran a la hora de medir la seguridad “overall code” tiene un rating de “E” debido a que han encontrado un security hotspot.

Estos hotpoints son áreas críticas del código fuente que presentan problemas de calidad o riesgos potenciales para la estabilidad, seguridad etc...



7 Sugerencias de mejora / Suggestions for improvement

Con base en nuestro análisis de la arquitectura del proyecto, sugerimos las siguientes mejoras:

Seguridad: Hay que revisar que los datos hash están seguros en esa parte del código (hay que revisar si el código presenta algún riesgo). Esto es debido a que hay algoritmos de hash criptográficos como MD2,MD4... que pueden tener collisions (un pequeño esfuerzo computacional es suficiente para encontrar dos o más entradas diferentes que produzcan el mismo hash). Presentamos una captura de la parte del código a modificar además del enlace del archivo:

https://sonarcloud.io/project/security_hotspots?id=twilio_twilio-python&file=twilio%2Frequest_validator.py&fileUuid=AXrjm4UZ_cU5C3dHjWlF&tab=code:

```
# compute signature and compare signatures
mac = hmac.new(self.token, s.encode("utf-8"), sha1)
computed = base64.b64encode(mac.digest())
computed = computed.decode('utf-8')
```

Documentación: Aunque la documentación del proyecto es buena, siempre es posible mejorarla, añadiendo más ejemplos, aclaraciones, detalles...Por ejemplo, en el README en la parte de "Uso de un cliente HTTP personalizado", se podrían añadir ejemplos de personalización y de cómo se podría implementar.

Mayor cobertura de pruebas: Aunque la cobertura del módulo es alta (por ejemplo la cobertura de pruebas de la carpeta Twilio es del 80.5%, como podemos ver aquí https://sonarcloud.io/code?id=twilio_twilio-python), es preferible tener una cobertura lo más alta posible.

8 Contribuciones al proyecto / Contributions to the project

Hemos hecho las siguientes contribuciones al módulo Twilio-Python:

Evitar duplicación de hashes:

https://github.com/twilio/twilio-python/blob/main/twilio/request_validator.py

```
mac = hmac.new(self.token, s.encode("utf-8"), sha1)
```

Al considerarse que el algoritmo criptográfico SHA-1 ya no es actualmente considerado seguro, por las posibles colisiones (2 o más inputs que producen el mismo hash) y que se utilizan en la verificación criptográfica de mensajes, la mejor opción es cambiar el algoritmo criptográfico a uno que no esté incompleto como SHA-512.

Importando de la librería “hashlib”: hashlib.sha512()

Y sustituirlo de sha1 en la línea de código mencionada.

9 Conclusiones / Conclusions

En este proyecto hemos documentado el diseño arquitectónico del módulo Twilio Python, un módulo de código abierto con la que los usuarios de Twilio pueden enviar y recibir mensajes y llamadas, además de realizar otras funciones. El módulo presenta un diseño bien elaborado y extensible, cuya capacidad de extensión es amplia. Sin embargo, tiene numerosos aspectos a mejorar como al documentación, las pruebas y mejoras en el código para que funcione correctamente.

Referencias / References

https://github.com/twilio/twilio-python

https://sonarcloud.io/project/overview?id=twilio_twilio-python

https://www.twilio.com/docs/libraries/python/custom-http-clients-python