Project Report

Final Project Description

v0.1, November 20, 2022

by

Angel González González

universidad de león Online

Contents

1

Introduction

GitHub link: `https://github.com/agonzg40/TFG-correcto`
Presentación: https://drive.google.com/file/d/1G3P$_h$$xtSiR0TIeHjNOm9$ − $ViH4I634lIG/view?usp=sharing$

1.1. Intro

A lo largo de este documento se van a explicar las diferentes fases del trabajo final de mi TFG, las fases que se van a explicar son las siguientes:

- Arquitectura
- Diseño
- Implementación
- Operaciones
- Automatización y test

1.2. Descripción

Este proyecto es una aplicación utilizando la metodología de Scrum, desarrollada en Python con la utilización de la tecnología de ROS y ROS2 (Robot Operating System), para suplir una serie de retos en la parte de funcionalidad propuestos por la ERL (European Robotic League), estos retos son:

- Reconocimiento y análisis de comandos de voz en inglés a través del micrófono del robot, dando la salida del análisis a un archivo de texto.
- Navegación en un apartamento del robot que se activa cuando el comando de voz anteriormente mencionado tiene el verbo "go" y el lugar al que desea ir.
- Reconocimiento de objetos y personas a través de la cámara del robot mediante el uso de opencv y coco_names

Este programa solo tiene despliegue en local, pero para su funcionamiento tiene que estar conectado a Internet, ya sea por cable o por WIFI, puesto que se utiliza la librería de speech_recognition de google para la obtención del texto que se dice al robot a través del micrófono y para situar un umbral por encima del ruido para obtener correctamente el mensaje.

Aunque esta aplicación se ha probado y utilizado satisfactoriamente en el MIC (Módulo de Investigación Cibernetica) también puede probarse (la parte de navegación) en Gazebo, la parte de reconocimiento de audio y reconocimiento de objetos y personas puede realizarse sin software adicional, con una cámara y un micrófono, está todo sobre su instalación, dependencias y como realizar su ejecución en el readme del repositorio de GitHub.

2

Architecture

2.1. Introduction

En este apartado se va a tratar la arquitectura del proyecto, tanto como es la arquitectura actual, como los problemas encontrados y sus posibles soluciones.

2.2. Estado actual de la arquitectura

Para poder explicar correctamente la arquitectura que sigue el programa, voy a dividir en dos figuras y voy a explicar cada una por separado, en la Figura 2.1 voy a enseñar la estructura para dos primeros retos, análisis de comandos y la navegación.

En la Figura 2.2 voy a enseñar el análisis de comandos y navegación:

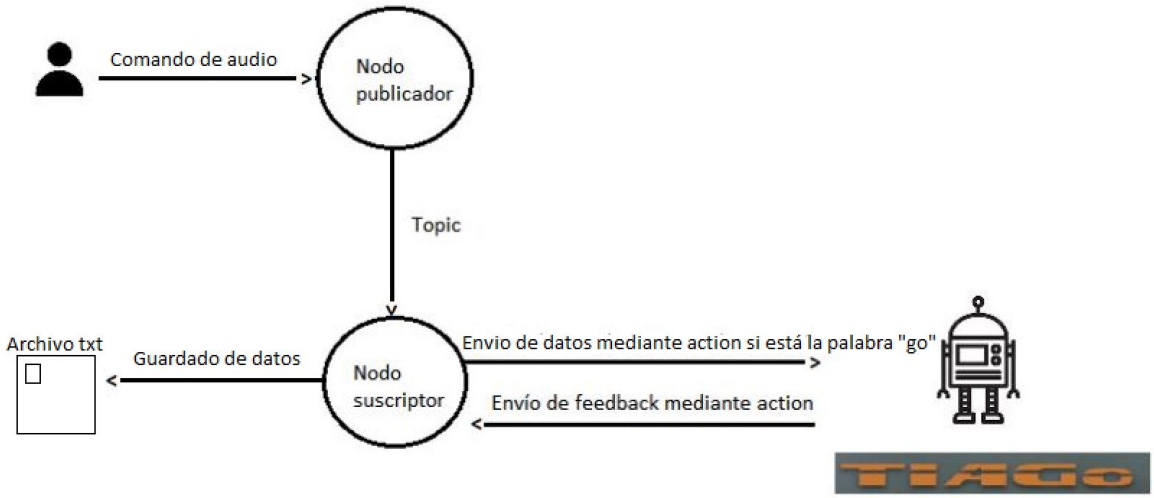

Figure 2.1: Arquitectura análisis de comandos y navegación

La Figura 2.1 enseña la estructura de dos retos, pero la navegación solo se iniciará si el comando dicho por el usuario tiene la palabra "go".

El usuario a través del micrófono dice un comando al robot, que este es recogido por el Nodo Publicador que lo pasa a un texto, este lo manda a través de un topic al Nodo Susciptor, que lo analiza y envía los resultados a un txt, en el caso de que tenga la palabra "go", se enviarán las coordenadas del lugar al que se quiere mandar el robot, el cuál avisará una vez que llega a su destino.

Como podemos ver en la Figura 2.2 está describe el funcionamiento para el reconocimiento de objetos a través de una cámara, el nodo publicador recoge los datos de la cámara del robot mediante fotos cada 0.1 segundos y los envía a través de un topic al nodo suscriptor, este junto con los datos del modelo entrenado

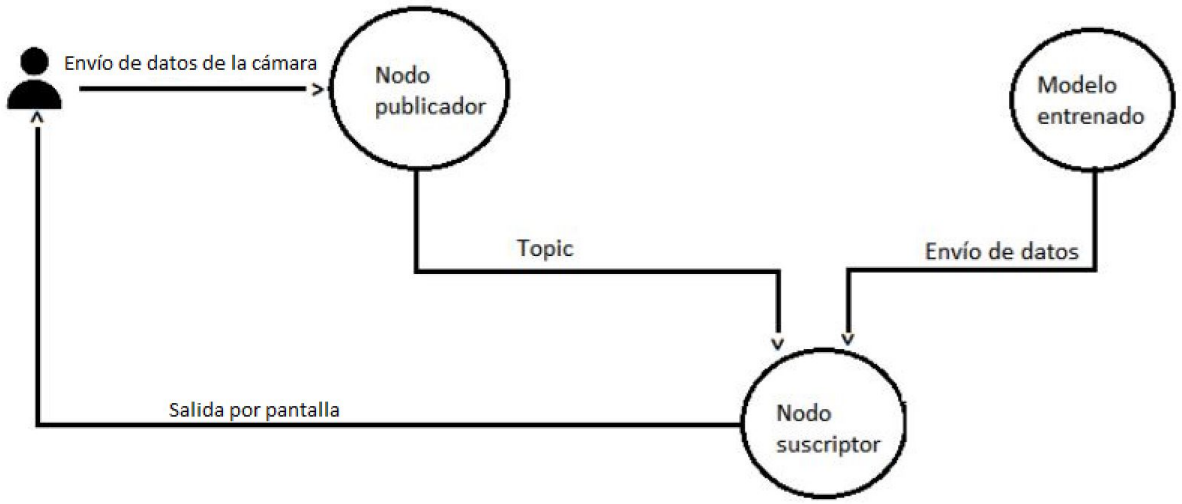

Figure 2.2: Arquitectura reconocimiento de objetos

va identificando que objetos que hay en cada imágen, y por último muestra los resultado a través de una ventana al usuario.

2.3. Problemas encontrados

Este análisis de los problemas encontrados es sobre esta aplicación tal y como está en gitHub, este proyecto se llevo a cabo para superar el TFG sin pensar en ningun momento abrirlo al público, para funcionar a medio de pruebas en el MIC, se han encontrado los siguientes problemas que pueden conllevar a ataques malintencionados:

- **Atacar el topic:** Se utiliza un topic para transportar los datos entre un nodo y otro, este topic se basa en la suscripcion del nodo suscriptor al nodo publicador a través del nombre de ese topic, con lo cuál si el atacante adivina el nombre del topic que esta hardcodeado, podrá enviar el mensaje que quiera al nodo suscriptor, es decir se estaría produciendo un spoofing.
- **Saturación del nodo:** De la misma forma que puedes enviar al nodo suscriptor lo que quieras sabiendo el nombre que se utiliza en el topic, también se puede saturar el nodo suscriptor con una gran cantidad de envíos seguidos al nodo, con esto se podría producir una denegación de servicios.
- **Comunicación del topic:** Los mensajes enviados a través del topic no tienen ningún tipo de encriptamiento, con lo cuál pueden ser interceptados, modificados y accedidos fácilmente por el atacante.
- **Lugares en código:** Los lugares a los que el robot se va a desplazar cuando escuche la palabra "go", son coordenadas que están en el código, con lo cuál el atacante podrá cambiarlas y poner las que quiera.
- **Entrada de micro:** La entrada de comandos a través del micrófono no esta sanitizada, con lo cual si se es capáz de pronunciarlo, se podría conseguir inyectar código.
- **Entrada topic:** Los datos que llegan a traves del topic no se comprueban.
- **Logs:** No existe ningun registro de logs.
- **Archivo con palabras:** Las palabras clave para el análisis de comandos están dentro de una carpeta del programa en unos archivos de texto. (UTILIZAR BASE DE DATOS CIFRADA)
- **Datos reconocimiento de objetos:** El archivo de coco names con los nombres en la posición enumerada de lo que puede reconocer está dentro de una carpeta en un archivo de texto en el programa.
- **No se han realizado tests:** No se ha pasado ningún de tipo de test unitario ni de otro tipo más alla de las pruebas que han sido realizadas en el MIC con el robot o las pruebas realizadas en el simulador de Gazebo en mi casa.

- **Guías de estilo:** No se ha utilizado ninguna guía estilo para realizar el código, pero durante el mismo se mantiene el mismo formato.

2.4. Soluciones a los problemas

Algunas de las soluciones que se podrían aplicar para resolver los problemas de apartado 2.3 Problemas encontrados:

- **Atacar el topic:** Es complicado corregir este problema, con lo cuál la mejor solución es buscar nombres que son muy complicados de sacar mediante el uso de fuerza bruta, adicionalmente a lo anterior, se pueden guardar los nombres en una base de datos encriptada.
- **Saturación del nodo:** Se puede utilizar sros2, que aunque está siendo desarrollada y es experimental, se puede utilizar [1]
- **Lugares en código:** En lugar de tenes los lugares hardcodeados, utilizar una base de datos encriptada para tener las coordenadas y su lugar correspondiente.
- **Comunicación del topic:** Una forma de resolver este problema es utilizar la tecnología de SROS, que utiliza TLS para las comunicaciones, aunque actualmente está bajo un fuerte desarrollo y está en una fase muy experimental.
- **Entrada de micro:** Sanitizar la entrada del micrófono una vez recogida permitiendo solo letras sin otro tipo de caracter: [Aa-Zz].
- **Entrada topic:** Comprobar los valores que llegan al topic permitiendo solo letras sin otro tipo de caracter: [Aa-Zz].
- **Logs:** Poner trazas que indiquen que esta pasando en todo momento en el programa y almacenarlas en un archivo.
- **Archivo con palabras:** Se pueden almacenar las palabras en una base de datos cifrada de tal forma que nadie pueda acceder a ella para no cambiar ningún caracter, el cifrado utilizado debe ser difícil de descifrar, por ejemplo utilizar una contraseña que contenga más caracteres que solo letras y números.
- **Datos reconocimiento de objetos:** Al igual que el archivo con palabras, se debe almacenar los datos de coco_names en una base de datos bajo cifrado.
- **No se han realizado tests:** Realizar tests unitarios a todo el código comprobando toda la funcionalidad y funciones.
- **Guías de estilo:** Seguir las guías de estilo de Python, se pueden encontrar aqui (2) REFERENCIA https://www.python.org/c

3

Design

3.1. Intro

Durante esta sección, se van a explicar varios puntos del problema, estos van a ser los posibles atacantes de nuestra aplicación, también se van a explicar los problemas que se han encontrado tal y como está el proyecto ahora mismo, así como algunas soluciones que pueden aplicarse a estos problemas encontrados.

3.2. Atacantes

Los atacantes pensados si este proyecto esta siendo diseñado para ser desarrollado y lanzado por una organización son los siguientes:

- Personas desde dentro de la organización
- Peronas ajenas a la organización con fines maliciosos
- Ataques accidentales

3.3. Problemas encontrados

- **Denegación de Servicios:** Ya que no se han implementado herramientas para evitar los ataques de tipo DDOS, si el atacante adivina el nombre del topic podrá realizar este tipo de ataques, mandando una gran cantidad de paquetes.
- **Ataque de tipo Man in the Middle:** Un atacante podría interceptar los paquetes enviados entre los nodos mediante los topic y editarlos haciendo llegar lo que quiera.
- **Acceso a los datos:** Los datos están almacenados en carpetas dentro del directorio de trabajo del proyecto, a los que se puede acceder facilmente y cambiar los valores.
- **Uso de software de terceros:** Para la navegación del programa se utiliza la propia navegación de ROS a la que solo se le pasan las coordenadas, la cual puede tener cualquier vulnerabilidad.
- **Librerías de terceros:** Se utiliza una gran cantidad de librerias que no son realizadas por la organización las cuales pueden tener vulnerabilidades.
- **Spoofing:** El atacante puede hacerse pasar por el nodo publicador en el caso de que consiga adivinar el nombre del topic y enviar paquetes.
- Eavesdropping: El atacante puede suscribirse al topic en el que se envían los datos, con lo cuál cuando el publicador envíe un mensaje lo recibirá el atacante.

3.4. Soluciones a los problemas

- **Denegación de Servicios:** Este ataque utilizando ROS es difícil de evitar , la mejor forma es buscar nombres que sean muy complicados de adivinar, y almacenar esos nombres en una base de datos encriptada sin que aparezcan los nombres hardcodeados.
- **Ataque de tipo Man in the Middle:** Una forma de solucionar este tipo de ataques es utilizar la tecnología de SROS [1] que aunque está bajo desarrollo y experimental se puede empezar a usar, pues lo que hace es utilizar TLS para las comunicaciones.
- **Acceso a los datos:** Almacenar los datos en una base de datos encriptada en vez de tenerlos en carpetas en el directorio de trabajo.
- **Uso de software de terceros:** No hay mucho que se pueda hacer en contra de esto más que estar atento a los avisos de sus creadores por si se encuentra alguna vulnerabilidad.
- **Librerías de terceros:** No hay mucho que se pueda hacer en contra de esto más que comprobar el código, y ver los avisos de sus creadores por si se encuentra alguna vulnerabilidad.
- **Spoofing:** Al igual que en el punto de Denegación de Servicios, una de las formas es guardar los nombres en una base de datos encriptada, y utilizar nombres que sean difíciles de averiguar por fuerza bruta utilizando caracteres alfanuméricos y especiales.
- **Eavesdropping:** Este tipo de ataques se podría resolver de la misma forma que el punto anterior Spoofing, y utilizar la tecnología de SROS para que aunque intercepten los paquetes, tengan seguridad.

3.5. Cuestionario

El cuestionario estará situado en el apartado de los Apendices.

4

Implementation

4.1. Intro

Durante la implementación se va a explicar como está el código del software, así como las herramientas, librerías utilizas y su lenguaje, la distribución del código fuente en cuanto a su estructura, y la evaluación con un analizador estático como es sonarqube, por último se darán algunas soluciones para resolver estos problemas encontrados.

4.2. Herramientas y librerías utilizadas

El proyecto se ha desarrollado utilizando el lenguaje de Python en la versión 3.10

Primero voy a hablar de las herramientas que se han usado para el desarrollo del software:

Herramienta	Version	Distribución
IDE Visual Studio Code	17.0	
OpenCV	4.5.5	
ROS2	8.2.5	foxy
Gazebo	11.9.1	
Rviz	1.14.13	

Las librerías utilizadas son:

Herramienta	Version	Version actual
rclpy	1.0.7	1.0.11
Sphinx	1.1.3	4.4.0
Pocketsphinx	0.1.15	5.0.0
string-utils	1.0.0	1.0.0
Stanfordnlp	0.2.0	0.2.0
Node	0.9.28	1.1
nltk	3.6.7	3.7
Pillow	9.1.1	9.3.0
Cv_Bridge	3.0.0	3.0.7
cv2	4.5.4.60	4.6.0
sys	2.1.4	3.2

4.3. Código fuente

El código de este proyecto es Open Source y se puede encontrar en el siguiente enlace de GitHub: código

La estructura que tiene el proyecto es la siguiente:

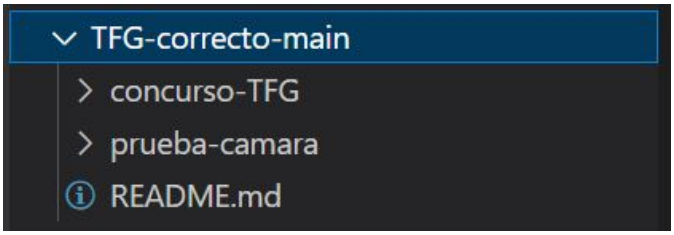

Figure 4.1: Estructura Principal

Como se ve en la Figura 4.1 el proyecto esta divido en 2 grupos, ya que la ejecución de este programa se hace por separado, primero voy a explicar la estructura de analisis de comandos y navegación que es la carpeta llamada concurso-TFG, la cual habría que cambiarle el nombre, y después explicare el reconocimiento de objetos que es la otra carpeta, el README.txt explica el manual de instalación del programa:

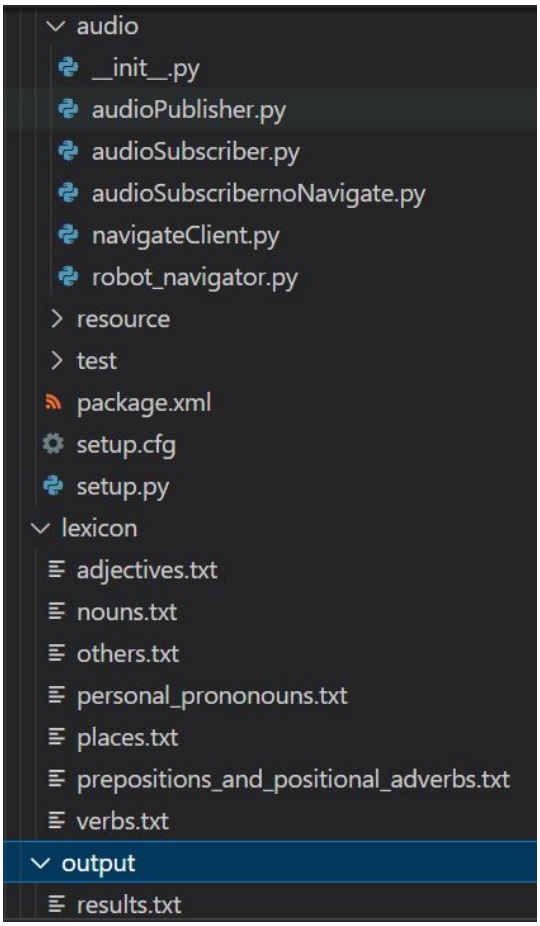

Figure 4.2: Estructura analisis de comandos y navegación

En la Figura 4.2 aparece la estructura de análisis de comandos y navegación, está dividida por:

- audio: esta carpeta contiene todos los archivos de ejecución del programa, es decir el código, dentro de la carpeta audio/audio se encuentra todo el código en el que los archivos audioSubscribernoNavigate.py y navigateClient.py son archivos de unas pruebas que se quedaron en la subida, habría que eliminarlos.
- lexicon: Esta carpeta contiene todas las palabras que se utilizan para analizar los comandos de voz, ya que se utilizan para analizar lenguaje natural.
- output: Esta carpeta contiene la salida del análisis del comando de voz.

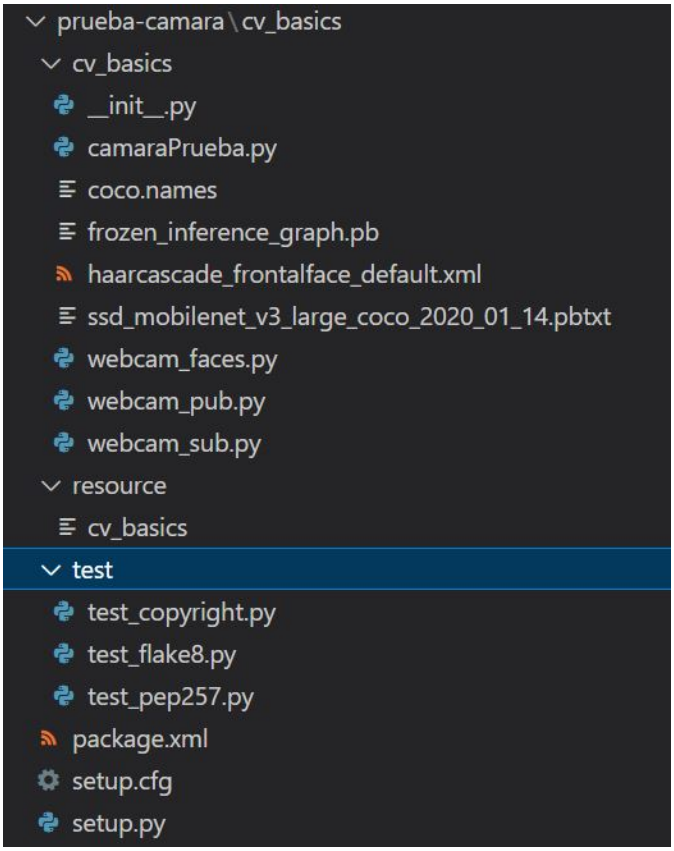

Figure 4.3: Estructura Principal

En la Figura 4.3 aparece la estructura de reconocimiento de objetos, está dividida por:

- cv_basics: Esta carpeta contiene todo el código realizado, las otras carpetas contienen archivos de configuración o se autocreado. Aquí habría que eliminar tres archivos que no son útiles, ya que se crearon para realizar pruebas y no fueron eliminados, estos son camaraPrueba.py, haarcascade_frontalface_default.xml, webcam_faces.py.

4.4. Evaluación de riesgos

Se ha realizado un análisis del código utilizando sonarqube del proyecto:

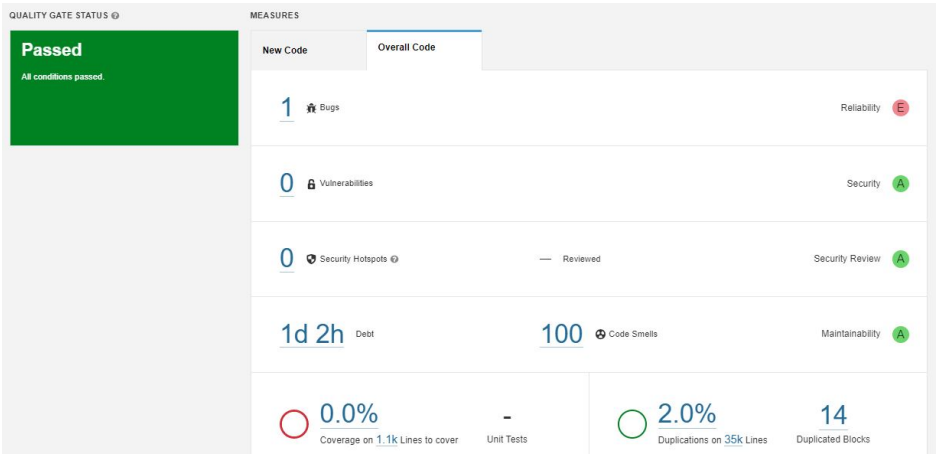

Figure 4.4: Análisis de sonarqube

Como se ve en la Figura 4.4, el análisis de sonarqube nos ha dado que de todo el proyecto hay un bug, el cual se va a explicar más adelante, además tiene 100 code smells, los cuales están distribuidos por todo el proyecto y la mayoría son nombres de variables o funciones. También nos indica que hay un 0% covereado, esto es porque no hay ningún test realizado.

Figure 4.5: Code smells y bug

En la Figura 4.5 se pueden ver algunos de los code smells que da sonarqube, así como un bug que dice que falta un argumento en la función identify se piden tres argumentos pero solo se pasan dos, además de esto los code smells es por problemas de tener variables declaradas que no se utilizan o por no utilizar nombres muy descriptivos.

Los 9 errores que son criticos es por tener una complejidad cognitiva muy elevada. Los otros 14 problemas "major" es por nombres de funcionalidades o por tener código comentado.

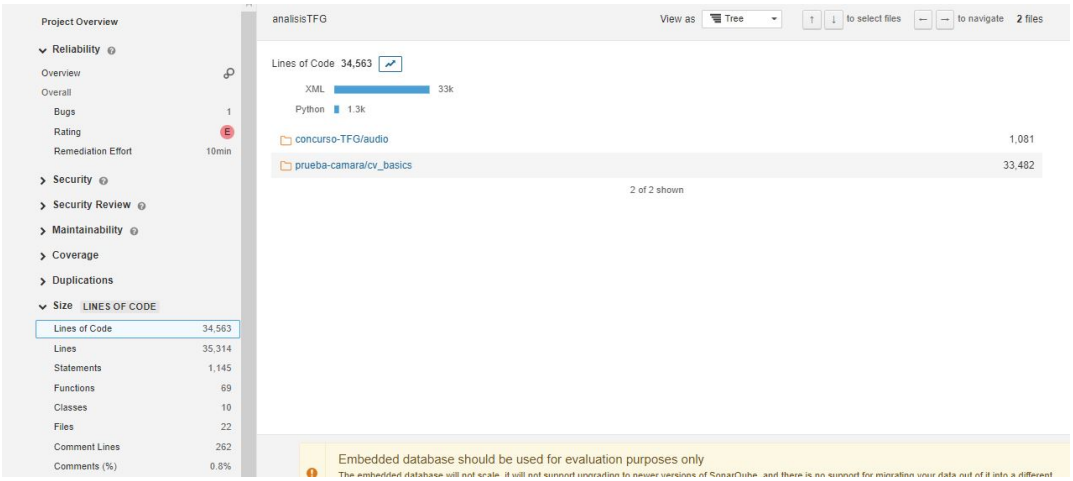

Figure 4.6: Lenguaje código

Por último la Figura 4.6 nos da información sobre el código, el lenguaje utilizado y las líneas de cada uno, en este proyecto se ha utilizado Python, y las 33k líneas de XML es el modelo entrenado que se ha utilizado para el reconocimiento de objetos y personas.

Además a la izquierda nos da la información de cuantas funciones hay, las clases, los ficheros y las líneas comentadas, como el tanto por ciento de estas.

Otro de los problemas es la utilización de librerias y software que en su momento estaban actualizados, pero ahora mismo ya están deprecados, estas librerias y software se reflejadan en las tablas del apartado 4.2.

4.5. Soluciones propuestas

Primero para solucionar lo que code smells analiza:

- Code smells (Bug): Comprobar si es necesario ese parámetro que falta, y en el caso de que sea necesario darle un valor en la llamada de la función identify.

- Code smells minor: Ir uno por uno en sonarqube haciendo lo que nos está diciendo por ejemplo para los que dice que no encuentra la expresión regular habría que quitar las mayúsculas como nos lo aconseja sonarqube.
- Code smells major: Al igual que el punto anterior, realizar lo que sonarqube recomienda, por ejemplo eliminar todas las líneas de código comentadas, esto se refiere a todas las que no sean explicativas de una función o de cualquier cosa.
- Code smells critical: Ya que estos code smells son de que la complejidad cognitiva es muy elevada, habría que pensar en el nombre de las funciones de otra forma para que fueran más fáciles de entender.

Los errores de uso de librerias y software deprecado, habría que actualizarlos a su versión actual, comprobando que no tengan ninguna vulnerabilidad notificada por el desarrollador.

5

Operations

5.1. Intro

En este capítulo se van a dar una serie de buenas prácticas que pueden ser aplicadas a este proyecto, así como el estado actual

5.2. Estado actual

Ya que este proyecto no se realizó pensando en ser desplegada, no se pensó en ello, con lo cuál si se quisiera desplegar se pueden realizar algunas de las siguientes soluciones:

- Entorno de test: Habría que buscar un entorno en el que se puedan realizar los testeos necesarios para saber que todo cumple con lo establecido en el diseño.
- Entorno de desarrollo: No existe ningún entorno para que los desarrolladores puedan realizar las pruebas y comprobar el funcionamiento desarrollado.
- Entorno para su lanzamiento: No se especifica en ningún sitio como se deben realizar los despliegues de la aplicación

5.3. Buenas prácticas

Algunas de las buenas prácticas que se podrían empezar a utilizar pueden ser:

- Dividir datos de configuración: Habría que empezar a hacer una división entre estos dos valores y tenerlos bien diferenciados.
- Tener un entorno de red controlado: Utilizar un entorno de red que sepamos que es seguro y los atacantes no van a poder atacarnos, por ejemplo comenzar a utilizar un buen firewall.
- Realizar auditorías periódicas: Para descubrir todo tipo de vulnerabilidades y problemas antes del despliegue de la aplicación habría que realizar auditorias por varias personas de la organización, y si se pudiese por personas fuera de la organización contratandolas.

6

Automation

6.1. Intro

En este capítulo se van a dar una serie de soluciones propuestas al estado actual del proyecto.

6.2. Estado actual y soluciones propuestas

En este apartado se muestran como se puede aplicar la automatización y las pruebas a realizar, ya que actualmente no se utiliza ninguna más las realizadas en el MIC para comprobar que todo funcionaba correctamente.

- Análisis estático: Realizar análisis estáticos cada cierto periodo de tiempo para comprobar que todo está correcto, por ejemplo sonarqube.
- Análisis dinámico: Utilizar alguna herramienta para realizar un análisis dinámico cada cierto tiempo, como puedo ser Tenable.IO
- Integración continua: Realizar pruebas de integración continua para saber en todo momento si el código tiene algún problema, como por ejemplo Jenkins
- Test unitarias: Realizar pruebas unitarias de todo el código y todas las funciones, para saber que todas realizan la función para las que fueron diseñanadas, en este caso se puede utilizar unittest ya que el código esta desarrollado en Python.
- Test de rendimiento: Realizar por lo menos 6 tipos de pruebas de rendimiento comunes, estas son pruebas de carga, resistencia, estrés, picos, volumen, escalabilidad, una herramienta que se puede utilizar es LoadNinja (de pago) o StormForge (tiene prueba gratuita)
- Realizar una cobertura del código: Para saber que partes del código están cubiertas por los tests, una herramienta que podemos utilizar es coverage.

7

Conclusion

Con la realización de este trabajo, he podido ver que mi aplicación es muy insegura, ya que no se realizan practicamente ninguna medida de seguridad, ya sea por temas que desconocía totalmente, como problemas de la propia tecnología, y he visto que para un atacante malicioso no es muy difícil conseguir lo que quiera de esta aplicación.

Además he visto que la seguridad es un factor que hay que tener en cuenta desde el principio en el propio diseño del programa, ya que después es mucho más complicado implementarlo y va a tener un coste mayor puesto que al pensarlo desde el inicio ya se realiza el resto del programa entorno a eso, pero si no hay que buscar las formas de implemetarlo cambiando la estructura ya hecha lo que tendría un coste.

También la seguridad es un factor que hay que tener en cuenta constantement, ya que avanza muy rápido, y este proyecto que hace un año no era tan inseguro, hoy en día es mucho más inseguro y tiene una gran cantidad de vulnerabilidades que antes no tenía.

Por último como trabajo futuro para este proyecto habría que realizar las comprobaciones redactadas en este documento, y aplicar las soluciones propuestas u otras que obtengan el resultado de tener una aplicación segura ante atacantes.

Appendices

A

Supporting Material

1. DATA CLASSIFICATION

Purpose *This section identifies the highest sensitivity level of data that the project involves. This information is needed to determine baseline data security requirements that must be addressed during the project.*

1.1. The project involves: *(check all that apply)*

☐ Non-Public University Information

Subcategories:

☐ Personally Identifiable Information

☐ Educational records and/or other information subject to FERPA regulations

☐ Information regarding an individual's mental or physical condition and/or history of health services use and/or other information subject to HIPAA regulations

☐ Financial information, including credit card and bank information, budgeting, salary and financial aid information

☐ Human Resources information

☐ Research information

☐ Other data the project sponsor considers sensitive, private, confidential or non-public

If any box above is checked, explain the nature, type and quantity of the data and why the involvement of this non-public university data is essential to the system or service to be delivered by the project:

Click here to enter text.

2. USE OF VENDOR IT SERVICES

Purpose *This section describes the intent, if any, to acquire ongoing vendor IT services (e.g., application software hosting, hardware/software infrastructure, data storage facilities, staffing, etc.) in support of this project or service. This information is needed to determine security requirements that should considered when evaluating vendor services and negotiating vendor contracts.*

2.1 Will the project acquire ongoing vendor IT services (e.g., application software hosting, hardware/software infrastructure, data storage facilities, staffing, etc)?

☐ Yes

☒ No. If checked, skip to Section 3.

2.2 The vendor service(s) will be acquired via:

☐ Request For Proposal

☐ Sole Source Procurement

☐ Purchase Order

☐ Agreement to vendor's online license user agreement

☐ Other. If checked, describe here:

Click here to enter text.

2.3 Briefly describe below the service(s) to be acquired, including names of desired vendor(s) if known:

Click here to enter text.

3. Identity Management, Access Control, Authorization

Purpose

This section identifies the user population who will have access to the IT product or service to be delivered by the project, as well as planned security access controls. This information will help determine if additional controls are needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

3.1. Who will access this application or system?

☒ Faculty

☒ Staff

☒ Students

☐ Consultants and temporary employees

☐ Other (please explain):

Click here to enter text.

3.2. If not covered above, what entities external to the University will have access to the application or service? Click here to enter text.

3.3. Is access limited to only those individuals whose job or function requires such access? Click here to enter text.

3.4. Is any part of the system open to the public or to an anonymous class of users?

All the project is open to the public, it is an open source project.

3.5. Briefly describe the process by which authorization of users will likely be accomplished, if known.

None, the code is uploaded in GitHub and public, they can access it if they want

3.6. Are there different levels of authorization in the system? (e.g., full access, limited access, read-only access, etc.)

All is full access

3.7. Is there an identified authority that approves requests for access to this system? Who would that be?

No

3.8. Is there a process for the access administrator to be notified when a user's status or role changes?

No

3.9. Will there be uniquely identifiable accounts for all users requiring access?

No

3.10. How will accounts which are no longer needed be recognized and deleted in timely and manageable manner?

There aren't any accounts.

3.11. How will this system authenticate users?

- ☐ CUNY Portal LDAP Single-Sign On
- ☐ Active Directory (cuny.adlan)
- ☐ CUNY Enterprise Active Directory
- ☐ CUNYfirst Single-Sign On
- ☐ Local Authentication

☐ Other:

Click here to enter text.

3.12. Where local authentication is used, provide details on the enforced password complexity and expiration policy.

There is any authentication.

3.13. Where local authentication is used, provide details on how passwords are securely stored within the system (e.g., encrypted using a salted hash).

3.14. Does the application automatically log off, lock or terminate a session after a predetermined time of inactivity?

There isn't any log off or disconnection, because there isn't any account, isn't needed.

4. Network Access and Communication

Purpose

This section identifies the scope of network access requirements. This information will help determine controls needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

4.1. Is this system required to be network accessible? ☒yes ☐ no

4.2. If so, will it be accessible:

☐ only within CUNY Central Office networks

☐ only within one or more CUNY Campus networks (specify)

☐ both CUNY Central Office and CUNY Campus networks

☒ the Internet at large

☐ other – please explain:

Click here to enter text.

4.4. Will this system be accessible through means other than the network (e.g., telephone)?

It is only accessible via computer and need the network to function properly.

5. Data Protection

Purpose

This section identifies available data protections and requirements. This information will help determine controls needed to reduce the risk of unauthorized or otherwise inappropriate access to sensitive data.

5.1. Are there restrictions on what quantity or type of data can leave the system?

The data that is used is a real data information, and isn't any restriction of the quantity of data leave the system, because all the data is used in real time, and delete it when is used.

5.2. Are shadow copies of any of the data anticipated to be created? For example, would users copy or download data to their own devices?

The only data that is saved is the coordinated of the apartment, but the users can't upload any data, only interact with the project with the voice.

5.3. Does data associated with this application or system interface with other applications or systems?

No

5.4. Is non-public university data encrypted while at rest?

No

5.5. Is the data encrypted while transmitted over an untrusted network? Click here to enter text.

No

5.6. What type of encryption is used? How is it configured and deployed?

There isn't any encryption.

6. Logging and Auditing

Purpose

This section identifies available activity logging and auditing capability. This information will help determine whether additional logging and auditing features needs to be established.

6.1. Describe logs and/or audit trails that are produced by the application or service.

The only logs that the program has is the real tiem logs, that appear in the command prompt about the command recognised and the navigation.

6.2. Is sensitive data embedded in the logs?

No

6.3. Can logs and/or audit trails link actions to individual users?

No

6.4. Are successful/unsuccessful accesses logged? With client network address?

No

6.5. For how long are logs retained?

When the system ends, the logs are cleaned.

7. Business Continuity / Disaster Recovery

Purpose

This section identifies business continuity and disaster recovery provisions and requirements.

7.1. Is there a documented business continuity / disaster recovery plan that addresses procedures to restore any lost data or functionality in the event of an emergency or other occurrence, the staff responsible for carrying out data restoration, emergency contact names and numbers, important business partners and other business supply information necessary for a temporary office setup to support data restoration?

No.

B

bibliography

Bibliography

[1]

SROS `https://github.com/ros2/sros2`.