# Project Report

## Final Project Description

v0.1, January 21, 2023

by

Jorge Blanco Prieto

universidad de león Online

# Contents

# 1 Introducción

**Github link:** `https://github.com/jblanp03/dps`
**DPS en Agora:** agora.unileon.es

## 1.1. Introducción

El presente documento muestra la aplicación de las 5 fases del libro Secure Coding Principles and Practices [4] a un proyecto consistente en el desarrollo de un asistente virtual de texto. Este documento tiene como objetivo poder evaluar la seguridad del proyecto presentado a partir de unos principios que se centran en evaluar distintos aspectos del software, como la calidad del sistema y el desarrollo seguro. Las 5 fases que se utilizan para dicha evaluación son:

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

## 1.2. Descripción del proyecto

El proyecto a evaluar es un desarrollo de asistente virtual, chatbot de flujo pregunta-respuesta guiado, desplegado en un entorno web que pretende mantener una conversación guiada con el usuario final, alumnos de un curso, con el objetivo de obtener un feedback más detallado acerca tanto del desempeño del curso como de los agentes participantes (instructores, ponencias, etc.), instalaciones y materiales, lecturas, etc. En una fase ajena al sistema desarrollado, los alumnos completan un cuestionario proporcionando un feedback sobre el curso. En funcion de las respuestas se generan tres tipos de usuarios finales:

- Usuario nivel 0: Usuarios con nivel de satisfacción es 100%.
- Usuario nivel 1: Usuarios con nivel de satisfacción se encuentra entre 75-99%
- Usuario nivel 2: Usuarios con nivel de satisfacción es superior a 50% e inferior a 75%
- Usuario nivel 3: Son usuarios cuyo nivel de satisfacción es inferior al 50%.

El uso de distintos perfiles de usuarios es para elaborar un flujo de pregunta-respuesta por parte del chatbot distinto, aunque de cara a la activación y uso del chatbot, todos los usuarios lo realizan de la misma manera. Los alumnos que estén registrados en la base de datos del sistema, es decir, aquellos que el centro

haya registrado su cuestionario, se les proporcionará un código de forma manual con el que podrán activar este chatbot instalado en un entorno web. Si los usuarios acceden al chatbot previo a tener registrado sus respuestas del cuestionario, el chatbot no les permitirá activar el flujo.

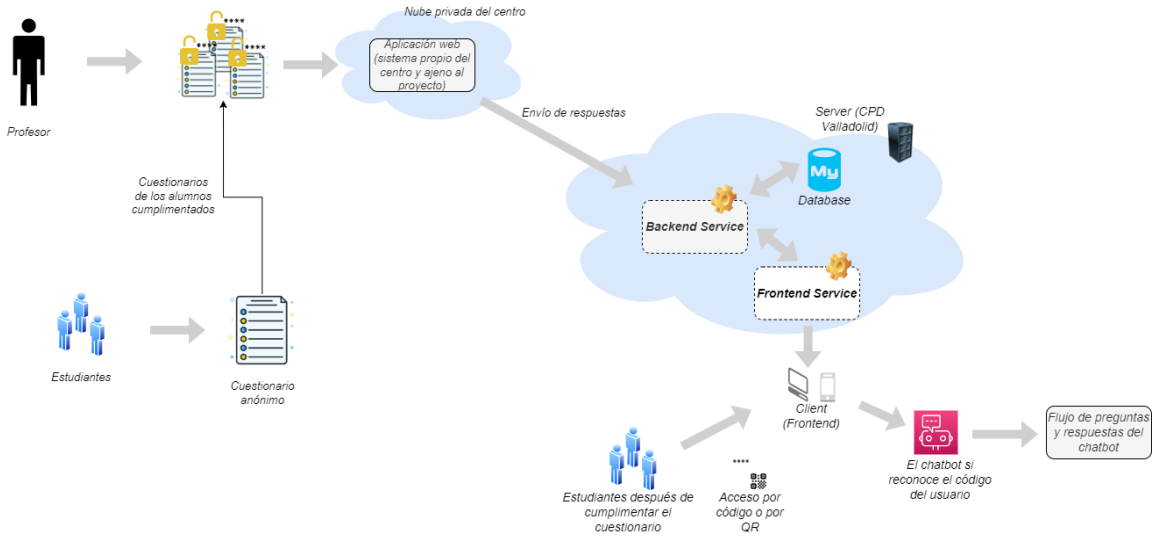


Figure 1.1: Esquema general

Procedimiento:

1. El profesor de la formación imprime un cuestionario de valoración que los alumnos deben cumplimentar en el aula. A cada cuestionario les proporciona un código de 6 dígitos y un código QR. El profesor se queda con una copia para asignar las respuestas posteriormente dentro de un sistema propio.
2. Con el cuestinario cumplimentado, el profesor introduce las respuestas y el sistema obtiene el grado de satisfacción de cada alumno. Se almacena su grado de satisfacción relacionado con el código (anónimo).
3. El sistema del asistente virtual interpreta dicho código y su grado de satisfacción y le asigna un tipo de usuario para el flujograma del proceso lógico.
4. El usuario accede al servicio del asistente para interactuar y el asistente detecta el código y su tipo de usuario.
5. El asistente entabla conversación con el usuario en función del flujograma de preguntas y respuestas.
6. Finaliza el flujograma y se finaliza el proceso.

# 2

# Arquitectura

## 2.1. Introduction

En esta sección se mostrará la arquitectura del asistente virtual web. Para facilitar la comprensión de la arquitectura empleada, se dispone de un esquema detallado del sistema expuesto en la figura 2.1. En ella se muestran los distintos agentes participantes en el sistema, así como las infraestructuras principales y sus relaciones.

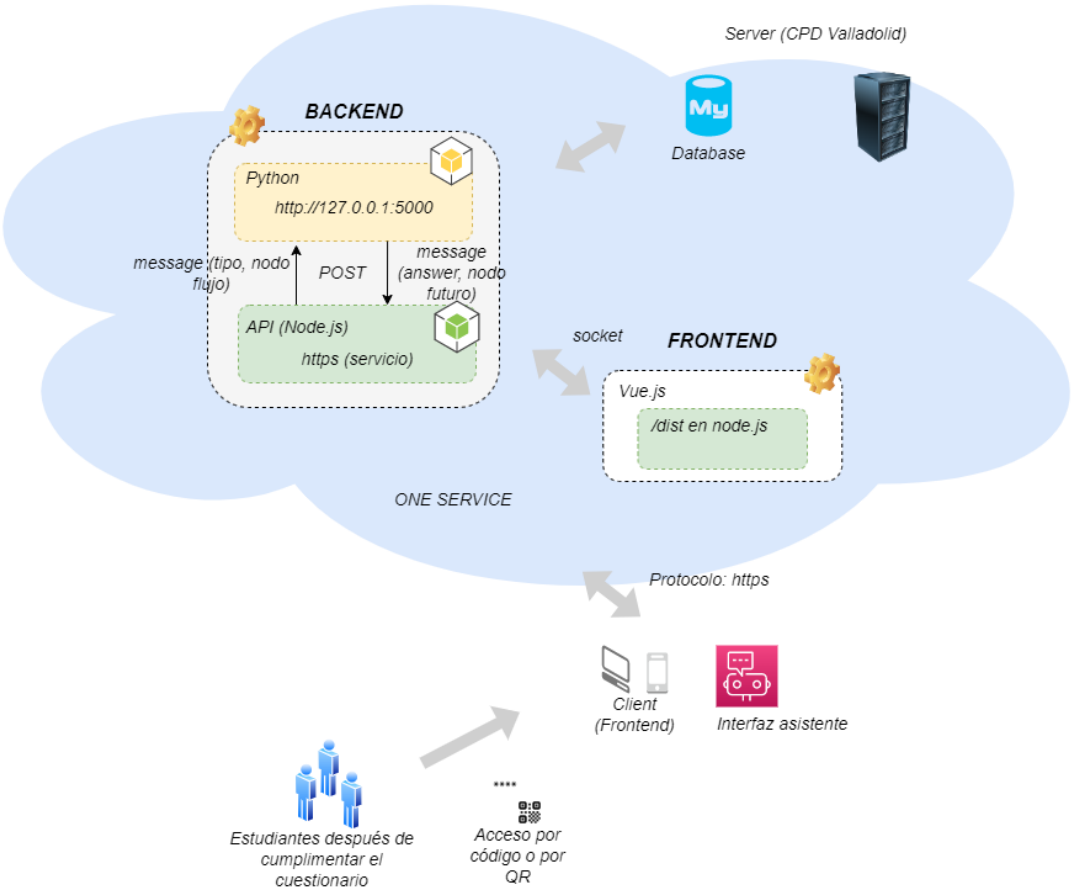


Figure 2.1: Arquitectura actual del sistema

La arquitectura del sistema es una parte fundamental de cualquier desarrolla y permite el correcto diseño.

Una arquitectura segura debe ser robusta frente ataques y debe comprender siguientes criterios [10]:

- Disponibilidad: se denomina disponibilidad a la disposición de los servicios a ser usados cuando sea necesario. La carencia de esta disponibilidad supone una interrupción del serivico y puede afectar directamente a la productivdad de las organizaciones. El asistente virtual funciona bajo demanda con un objetivo simple: recoger una opinión más detallada de los usuarios. Los usuarios suelen ser propensos a no proporcionar opiniones por su propia voluntad, a no ser que un agente externo les incite a participar. La disponibilidad de este servicio es clave en la recolección de estas opiniones, dado que su interrupción o mal funcionamiento puede derivar en una mala experiencia o desuso del asistente.
- Integridad: se corresponde con el mantenimiento de las características de completitud y corrección de los datos. Los mensajes enviados entre el asistente y los usuarios debe evitar ser manipulado por terceros, con el fin de evitar la manipulación de las opiniones.
- Confidencialidad: la información debe únicamente llegar a las personas autorizadas. Contra la confidencialidad o secreto pueden darse fugas y filtraciones de información, así como accesos no autorizados. En este caso el sistema no trabaja con datos personales de usuarios, no obstante la filtración de las opiniones de los usuarios, así como el acceso a usuarios no autorizados (usuarios que no accedieron al curso previamente) puede derivar en toma de datos que distorsionen la veracidad de las opiniones.
- Autenticidad: Propiedad o característica consistente en que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos. Se requiere mantener la autenticidad de los datos y la autenticidad de los usuarios, evitando suplantación de identidad.
- Trazabilidad: Aseguramiento de que en todo momento se podrá determinar quién hizo quué y en qué momento. El asistente virtual lleva un registro de información de todas las comunicaciones con el id del usuario.

## 2.2. Estado actual

En esta sección se pretende definir el estado actual del sistema.

- Alojamiento: El sistema se encuentra alojado en un servidor físico del CPD del Parque Científico de la Universidad de Valladolid que cuenta con la ISO27001 [1]. Dispone de configuración de firewall para protección de los servicios alojados. Características del servidor:
  - El sistema tiene un ancho de banda asegurado a internet Full Dúplex: 2 Mb.
  - Tiempo de respuesta en días laborales: 20 minutos.
  - Tiempo de resolución de incidencias críticas en días laborales: 40 minutos.
  - Tiempo de respuesta en días no laborales y festivos: 1 hora.
  - Tiempo de resolución de incidencias críticas en días no laborales y festivos: 2 horas.
  - Disponibilidad del servicio de alojamiento superior al 95% (salvo causas de fuerza mayor por intervenciones de mantenimiento y pruebas).
- Relación arquitectura de los componentes y arquitectura física: El asistente virtual es un web service HTTPS que se encuentra desplegado en una máquina virtual del CPD. El servicio cuenta con un proxy inverso para habilitar el cifrado SSL, instalado y configurado, y se puede acceder a partir del DNS del servicio (evitando errores de SSL). Para desplegar el servicio backend (API) de nodejs se utitliza el administrador de procesos PM2. El microservicio de Python se despliega mediante Flask, framework escrito en Python que permite crear aplicaciones web, en local, dentro de la propia máquina. Además, cuenta con la instalación de un WAF (Web Application Firewall) que añade una capa de seguridad, supervisando, filtrando y/o bloqueando el tráfico HTTP hacia y desde una aplciación web.
- Base de datos: El servicio cuenta con una base de datos MySQL Server 8.0 [2], instalada dentro del mismo CPD en una máquina virtual aparte. La administración de esta base de datos es dada por únicamente la cuenta administrador (equipo de soporte del CPD). La base de datos no contiene datos personales de los usuarios que establecen comunicación con el asistente virtual. Las conversaciones se asocian a un id, que es el código de acceso proporcionado a los usuarios de forma manual en el centro. Los códigos, a su vez, están relacionados con los formularios cumplimentados con las opiniones del curso.

- Servicios: El backend del sistema (API) se encuentra desarrollado en Node.js mediante el framework Express y el frontend está desarrollado en Vue.js. El sistema cuenta con un servicio web service y el frontend de la aplicación (interfaz del chatbot) se integra dentro del backend mediante la carpeta /dist (carpeta que contiene la versión minimizada del código fuente y se utiliza en las aplicaciones en producción). El servicio que trata la lógica del asistente virtual está desarrollado en Python y se despliega en local dentro de la máquina virtual del propio servidor. A través del puerto 443, se expone el servicio del asistente virtual a Internet (se utiliza DNS). El servidor de aplicaciones y la base de datos se encuentran en máquinas distintas suponiendo una gran ventaja desde un punto de vista de seguridad. No hay desventajas por latencia en las comunicaciones.

## 2.3. Problemas encontrados

A raiz del estudio de la arquitectura y del desarrollo del proyecto se han detectado posibles problemas que han de ser solucionados:

- El acceso a la aplicación no tiene un sistema de autenticación. Cualquier usuario podría acceder al asistente virtual si tuviera la dirección web. Aunque el asistente virtual para entablar comunicación con los usuarios en un primer lugar pide al usuario que introduza el código que les ha proporcionado el centro de forma manual, no hay ningún trato de intentos de introducción de código. Además el código es un código numérico de 6 dígitos, lo que podría presentar un problema de seguridad por ataques o suplantación de la identidad de los usuarios.
- El código de acceso es de un único uso, pero tiene duración ilimitada.
- Guías de estilo. Durante el desarrollo no se han seguido las guías de estilo en ninguno de los lenguajes utilizados (JavaScript y Python). En el desarrollo han intervenido tres desarrolladores (un desarrollador Frontend, un desarrollador Backend y un desarrollador backend Python).
- Archivos de logs: Hay logs del servicio pm2 en el servidor; no obstante si sucede un error durante la ejecución, no queda registrado.
- Tiempo de incidencias: A pesar de que el CPD ofrece un tiempo de respuesta todos los días del año, el equipo de desarrollo no ha definido ningún plan ante incidencias por parte del cliente. Por otra parte, el tiempo de respuesta ante las incidencias los fines de semana son altos dado que los cursos suelen finalizar entre el sábado y domingo y son esos días de la semana con más tráfico hacia el asistente.
- Control: Las conversaciones quedan registradas en la base de datos, pero no hay un control de acceso a la web de modo que no se conoce cuánto tráfico externo acceder a la web sin que haya establecido una comunicación con el asistente.

## 2.4. Soluciones propuestas

En esta sección se proponen las siguientes soluciones a los problemas encontrados:

- Acceso a la aplicación: Debido a que el sistema, como requisito del cliente, no recoge información personal de los usuarios, no se pueden autenticar por vía clásica (email o username y contraseña). Además, este sistema debe ser ligero y optimizar UX/UI de los usuarios para que no resulte pesado interactuar con el asistente. Es por ello por lo que se puede incidir en el código proporcionado, modificando su estructura y creando un código compuesto por las directrices que propone OWASP, [6], para el tratamiento de contraseñas de usuario.
- El código de acceso debe tener un periodo de caducidad. El periodo de caducidad propuesto debe ser hasta el tratamiento de los datos de las encuestas, que coincide con 1 mes antes del inicio de cada curso (los cursos se realizan dos veces por año). El centro tiene su disposición el contacto de los alumnos, por lo que podría envias un correo electrónico general recordando hacer uso del asistente previo a la caducidad del código.
- Guías de estilo. Se deberían utilizar guías de estilo de cada propio lenguaje de programación. Cada framework utilizado tiene su propia guía de estilos: Node.js [7], Vue.js [3] y Python [5].

- Archivos de logs: Se deberían incluir más puntos de recogida de logs dentro del código desarrollado del asistente para tener mayor precisión en el tratamiento de errores.
- Tiempo de incidencia: El equipo debe elaborar un plan de respuesta a incidencias.
- Control: En el estado actual, el sistema crea una nueva conversación en la base de datos cuando el usuario hace click en el plugin de la web e introduce su código de usuario. Se pierde mucho control del tráfico que puede existir en la web, por lo que se propone modificar la estructura de desarrollo y que se active el plugin del asistente cuando un usuario acceda a la web. Dado que es una web específica para el asistente virtual no tiene desventaja sobre UX/UI del usuario en la web. De esta manera se puede llevar un control más preciso.

# 3

# Diseño

## 3.1. Introducción

En este apartado se aborda el diseño del sistema del asistente virtual. Al igual que la arquitectura, el diseño es una de las secciones más importantes en el desarrollo de un proyecto. El diseño refleja las necesidades y requerimientos del cliente, además de una solución borrador del desarrollo que se debe llevar a cabo. Un mal diseño, y una mala arquitectura, pueden derivar en futuros fallos del sistema, desde fallos por parte de los usuarios finales, fallos en la implementación e incluso en desviaciones del proyecto.

Para analizar el diseño se han especificado las necesidades del proyecto, identificado los grupos stakeholders principales y definidos los requisitos de funcionalidad y de usuario. Además, se introduce una sección de realizabilidad que corresponde con posibles riesgos en la implementación.

## 3.2. Necesidades del proyecto

Una parte importante de la ingeniería de requisitos es establecer las necesidades de las partes interesadas, encargada de documentar y mantener los intereses de los stakeholders pertenecientes al sistema:

- ICU-0: Establecer comunicación con el asistente virtual en un servicio web.
- ICU-1: El asistente virtual debe ser proactivo y comenzar la conversación cuando un usuario acceda al servicio web.
- ICU-2: Los usuarios deben poder entablar conversación a partir de un código o bien accediendo mediante un código QR.
- ICU-3: Los usuarios deben ser anónimos.
- ICU-4: Las conversaciones deben ser almacenadas de forma anónima.
- ICU-5: El asistente virtual debe establecer flujos de preguntas en función de las respuestas de las encuentas de los usuarios.

Dentro del proyecto se pueden definir dos grupos interesados, aunque a nivel de uso del asistente, únicamente el grupo B es el interesado:

- Grupo A: Centro que proporciona el curso. Este tipo de grupo interesado necesita conocer opiniones de los alumnos más detalladas que con la encuesta para poder ofrecer mejores servicios.
- Grupo B: Usuario alumno. Este tipo de usuario corresponde con los usuarios que acceden a la aplicación y pueden conversar con el asistente virtual. A su vez, este grupo podría desagregarse en varios tipos de usuarios alumnos que varían en función de las encuestas que previamente habrían rellenado en el centro:
  - Usuario grupo nivel 0
  - Usuario grupo nivel 1

  - Usuario grupo nivel 2
  - Usuario grupo nivel 4

  Dada la identificación de este grupo, los usuarios se pueden dividir por niveles cuyas variaciones corresponden con las respuestas de la encuesta (grado de satisfacción). El asistente virtual identificará a qué grupo pertenece cada usuario previo a lanzar el flujo guiado de preguntas; no obstante, la forma de utilizar este asistente es igual para cualquier grupo de usuarios.

## 3.3. Requisitos del proyecto

En esta sección se exponen, evalúan y priorizan los requisitos de usuario del sistema del asistente virtual para transformarlos en una descripción funcional y técnica. El objetivo de esta fase es refinar los requisitos de usuario definidos en la sección anterior, eliminar ambigüedades y traducirlos al lenguaje técnico. Los requisitos se agrupan en categorías según el modelo FURPS [9]:

- Requisitos de funcionalidad (RF): cómo funciona el sistema.
- Requisitos de usabilidad (RU): conceptos de uso.
- Requisitos de fiabilidad (RFi): detallan la frecuencia y gravedad de los fallos, así como la capacidad de recuperación.
- Requisitos de rendimiento (RR): imponen condiciones a los requisitos funcionales.
- Requisitos de soporte (RS): extensibilidad y configurabilidad.

Los requisitos que afectan al sistema se pueden indicar mediante la etiqueta "Requisito crítico". En las tablas de la sección Requisitos de usuario se detallan los requisitos del sistema. Cada tabla, sin contar la tabla 3.1, corresponde a un nivel de requisitos. Así, en la tabla 3.2 están los requisitos de nivel 0 y en la tabla 3.3 los definidos como nivel 1:

- ID: Identificador único del requisito, siguiendo las reglas especificadas en la introducción y que permite hacer la relación entre requisitos de niveles diferentes.
- Nombre del requisito: nombre específico del requisito.
- Descripción: descripción completa del requisito.
- Justificación: causas que justifican el requisito.
- Prioridad (P):
  - Alta (A): debe completarse.
  - Media (M): si no entra en conflicto con otros requisitos y hay suficientes, debe cumplirse.
  - Baja (B): si es posible, se incluirá en el diseño, ya sea desde el principio o como una futura ampliación.
- Verificación (V): método de verificación que se utilizará para garantizar su cumplimiento.
  - Inspección (I): Examen visual (puede ser automático) que proporciona evidencia de que el requisito se ha cumplido, sin necesidad de ningún procedimieinto de ensayo específico.
  - Ensayo (E): Procedimiento específico diseñado para comprobar el funcionamiento del sistema o los parámetros constructivos y determinar el cumplimiento de los requisitos.
  - Demostración (D): El sistema se pone en funcionamiento para demostrar que cumple el requisito. Puede realizarse una o varias veces y documentarse.
  - Análisis (An): Suele realizarse cuando no es posible ningún otro análisis. Un experto analiza el sistema o parte, junto con los elementos con los que interactúa, y determina si cumplen los requisitos, emitiendo un informe final.
- Necesidad del usuario de la que se deriva dicho requisito (NU).

### 3.3.1. Requisitos de usuario

En esta sección se detallan los requisitos de usuario del sistema en función de la terminología expuesta.

Posteriormente se establecen las necesidades de usuario cuyos requerimientos particulares se encuentran descritos en la TABLA teniendo en cuenta los grupos interesados:

| ID | Necesidades del usuario | Grupo de pertenencia |
|---|---|---|
| NU-1 | El usuario debe poder establecer comunicación con el asistente virtual en un servicio web | Grupo B |
| NU-2 | El usuario debe poder acceder al servicio web desde PC y/o desde dispositivos móviles | Grupo A / Grupo B |
| NU-3 | El asistente virtual debe iniciar la conversación con el usuario cuando este acceda al servicio web | Grupo B |
| NU-4 | Los usuarios deben poder comenzar la conversación a través de un código proporcionado por el centro | Grupo A / Grupo B |
| NU-5 | El asistente virtual debe diferenciar el alumno y clasificarlo según tipo de flujo a seguir en la conversación | Grupo A |
| NU-6 | El asistente virtual debe poder conectarse con la base de datos en todo momento | - |
| NU-7 | De cara al sistema los usuarios serán anónimos | Grupo A |
| NU-8 | Todos los datos han de ser registrados en la base de datos | Grupo A |

Table 3.1: Identificación de las necesidades del usuario asociados a los grupos de referencia

### 3.3.2. Requisitos del sistema

Una vez detallados los requisitos generales en función de las necesidades del usuario, se detallan en la tabla 3.2 los requisitos de nivel 0 y la tabla 3.3 que corresponde con los requisitos más detallados de nivel 1. Las tablas se pueden observar al final del presente capítulo.

## 3.4. Realizabilidad

En esta sección se exponen los posibles riesgos que pueden llegar a surgir en un entorno de producción con el desarrollo expuesto. Esta parte es necesaria para entender la madurez de la propuesta del proyecto:

- La conversación del asistente virtual debe ser una conversación asistida de flujo guiado. Esto en gran parte se debe a la necesidad del usuario del grupo A (centro) porevitar errores de un asistente conversacional (conversación general, sin flujo) y rapidez y optimización en el flujo de preguntas para evitar generar rechazo. No obstante, otro motivo fundamental es la falta de datos históricos de conversaciones o de un dataset. Las conversaciones guiadas son cerradas y obliga al usuario a responder cuestiones acerca del tema tratado por el asistente. Como se ha mencionado y resultando ventajoso, esto permite minimizar los posibles fallos de entendimiento. Sin embargo, permitir que el usuario tenga un campo de texto para responder al usuario puede sentirse perdido. Es por ello por lo que se recomienda evitar en la mayoría de lo posible que el usuario utilice el campo de texto haciendo interactivo el chat del asistente y ofreciendo al usuario opciones de respuestas que con hacer click se pueda responder.

## 3.5. Problemas encontrados

En esta sección se tratarán de exponer los problemas encontrados o posibles riesgos existentes durante la definición y diseño del proyecto.

- Documentación: Falta de documentación del diseño del sistema. La documentación existente no está conectada en un libro de proyecto y no está actualizada.
- Nuevo integrante del equipo de desarrollo: Un problema encontrado es la integración de un nuevo miembro del equipo técnico como sustituto de otro técnico o como ampliación del equipo. Este puede suponer un problema desde un punto de vista de implementación, desarrollo y mantenimiento del software, dado que no hay documentación. Las consecuencias pueden ser desde empeoramiento de la calidad del código, hasta largos plazos de entrega de nuevas versiones y/o modificaciones, además de existir la posibilidad de generar errores involuntarios.

- Librerías de terceros: Se ahondará en este tema en el capítulo 4, pero existe un riesgo de uso de librerías no actualizadas.
- Equipo técnico desidentificado: El equipo técnico participante del proyecto puede llegar a no identificarse con el proyecto en sí, concluyendo en un desconocimiento de para qué sirve lo que están desarrollando. Esto puede originar en errores, largos periodos de desarrollo y/o modificaciones por equivocaciones y no entendimiento del proyecto, etc.
- Archivo de logs: Este riesgo existe por la falta de un arachivo de logs y en caso de existir un problema dentro del sistema, no quedaría registrado.

## 3.6. Soluciones propuestas

Dados los problemas y/o riesgos encontrados en la sección 3.5, se proponen las siguientes soluciones:

- Documentación: Redactar una documentación detallada del proyecto, así como de su desarrollo y modificaciones. También se puede redactar un libro del proyecto que agrupe todos estos informes y las actualizaciones que se llevan a cabo.
- Librerías de teceros: Al igual que se propone en el capítulo 4, una solución es revisar de manera periódica la versión de las librerías y su documentación por si existen vulnerabilidades.
- Equipo técnico desidentificado: Se debería tener reuniones de inicio de proyecto para explicar en qué consiste el proyecto y reuniones periódicas para que todo el equipo participante esté enterado de las actualizaciones/cambios y desarrollo del mismo.
- Nuevo integrante en el equipo de desarrollo: Con el punto de la documentación, este aspecto dejaría de ser un posible riesgo, dado que el nuevo integrante tendría documentación de apoyo.
- Archivo de logs: Añadir un sistema (archivo) de logs para recoger la historia de la aplicación web. Este es un punto importante en el momento de puesta en un entorno de producción.

| ID | Nombre del requisito | Descripción | Justificación | P | V | NU |
|---|---|---|---|---|---|---|
| RF-0 | (Requisito crítico) Comunicación bidireccional entre los usuarios y el asistente virtual | La aplicación web debe permitir comunicación bidireccional | Es un sistema de pregunta y respuesta | A | T - Realizar conversaciones entre ambas partes para probar que la comunicación se realiza correctamente | NU-0 |
| RF-1 | El acceso al asistente virtual debe ser mediante un web service | Web service | Acceso a través de internet | A | I - Utilizar un navegador web para acceder una vez sea desplegado | NU-1 |
| RF-3 | El asistente virtual debe poder activarse a partir de la lectura de QRs | A los alumnos del centro se les proporcionará un código en formato texto y en QR con acceso al servicio web | Mejora de la experiencia de usuario | A | An - Realizar un proceso de prueba para verificar la correlación del alumno | |
| RU-4 | Se requiere proactividad por parte del asistente virtual | Cuando el usuario acceda el asistente virtual iniciará la conversación | Se evita dejar al usuario el control de la conversación y se minimizan errores y/o pérdida del mismo | A | | NU-2 |
| RF-5 | El asistente virutal debe ser personalizado | El asistente virtual identificará las respuestas de cada usuario y ofrecerá un flujo de conversación con cada uno en función de sus respuestas previas | Los flujos de preguntas deben ser en función de cada posibilidad en el cuestionario | A | T - Realizar cuestionarios distiintos para comprobar los flujos de preguntas | NU-4 |
| RF-6 | (Requisito crítico) El asistente virtual (el sistema) debe registrar las conversaciones | El sistema debe ser capaz de recoger datos de conversaciones y de usuarios | Con estos datos, a futuro, el centro desea poder realizar un pequeño análisis y extraer conclusiones de las opiniones | A | | NU-8 |

Table 3.2: Primer nivel de la tabla de requisitos

| ID | Nombre del requisito | Descripción | Justificación | P | V | Metodología de verificación |
|---|---|---|---|---|---|---|
| RR-0.1 | La comunicación debe ser privada entre el usuario y el asistente virtual | Cada comunicación es privada y unipersonal | | A | T | Realizar comunicaciones simultáneas para evitar que se filtren datos entre distintos usuarios |
| RF-1.1 | Los usuarios podrán acceder e interactuar con el asistente mediante ordenador y dispositivos móviles | Los usuarios podrán acceder desde cualquier tipo de dispositivo al servicio web | | A | | |
| RF-4.1 | Mensaje de bienvenida del asistente virtual e identificación del código o la procedencia del QR | Cuando el usuario accede al asistente, este emitirá un primer mensaje de bienvenida. A partir de aquí, el asistente virtual guiará la conversación | | M | T | Realizar una simulación y probar el asistente virtual con distintos códigos para comprobar su fiabilidad |
| RF-4.2 | Los usuarios podrán valorar el servicio del asistente mediante una rápida encuesta | A partir de la selección de un número de estrellas (entre 0 y 5) y un comentario opcional, los usuarios podrán valorar el asistente y su funcionalidad | | | | |

Table 3.3: Segundo nivel de la tabla de requisitos

# 4

# Implementación

## 4.1. Introducción

En el presente capítulo se describen los puntos más importantes de la implementación del asistente virtual: tecnologías utilizadas, pseudocódigo, código fuente y librerías utilizadas. Posteriormente se mostrarán los posibles riesgos y problemas encontrados y las soluciones propuestas.

## 4.2. Tecnologías utilizadas

El asistente virtual se materializa en un plugin que inicialmente se iba a instalar en la web del centro, aunque posteriormente y con el fin de reducir la exposición del asistente al público en general, se ha desplegado en una web estática destinada únicamente para el asistente virtual. A esta web se puede entrar por búsqueda en internet, a partir de un link dentro de la web del centro o bien por el código QR que se les proporcionó a los estudiantes. El asistente virtual consta de varias partes: backend del asistente, microservicio de python, base de datos y frontend.

### 4.2.1. Asistente virtual

Tal y como se ha comentado recientemente, el plugin del asistente es una aplicación web que se divide en dos partes: backend y frontend. El frontend es el diseño web que el usuario visualiza a través de su dispositivo de acceso, mientras que el backend es donde se realizan toda la lógica de negocio y/o procesos necesarios para establecer un correcto funcionamiento y comunicación.

#### 4.2.1.1. Backend

El backend del asistente virtual se ha desarrollado en el entorno de ejecución Node.js v18.2.0. Node.js es un entorno de tiempo de ejecución de JavaScript de backend que se ejecuta en el motor JavaScript V8. Se ha utilizado Node.js debido a que:

- Gran comunidad open-source.
- El equipo de desarrollo backend es el lenguaje al que se encuentran más acostumbrados.
- Node.js tiene una arquitectura basada en eventos capaz de E/S asíncrona lo que permite optimizar el rendimiento y la escalabilidad en aplicaciones web con muchas operaciones de entrada/salida. Esta característica es muy útil en comunicaciones en tiempo real (punto clave en el proyecto).

#### 4.2.1.2. Frontend

El frontend del asistente se ha realizado mediante el framework de JavaScript de código abierto Vue.js v3.0.1 (última versión estable que corresponde con Octubre de 2020). Los componentes de Vue.js extienden los elementos básicos de HTML para encapsular el código reutilizable. En un nivel alto, los componentes son elementos persoanlizados a los que el compilador de Vue adjunta comportamiento. Vue.js utiliza una sintaxis de plantilla basada en HTML que permite vincular el DOM (Documento Object Model - interfaz de plataforma) a los datos de la instancia subyacente. Vue.js permite que en un mismo fichero se pueda incluir el código de HTML, CSS y JavaScript. El uso de este framework está muy extendido (al igual que la pareja

node.js para el backend y vue.js para el frontend) y principalmente se ha utilizado por su rápida curva de aprendizaje.

### 4.2.2. Microservicio Python

La lógica del asistente virtual viene dada a partir del lenguaje de programación Python v3.10.4. En Python se ha creado un microservicio que, mediante socket se comunica con la API de Node.js. Python recibe la petición del usuario a través de Node.js, procesa la petición y envia una respuesta para que Node.js se comunique con el frontend. Python es un lenguaje de programación de propósito general y alto nivel, muy extendido para crear asistentes virtuales y el desarrollo de algoritmos de Procesamiento Natural del Lenguaje o NLP. En este proyecto no fue requerido crear o desarrollar un algoritmo de NLP dado que las respuestas del asistente viene preconfigurada por el flujo de seguimiento.

## 4.3. Pseudocódigo

La ejecución del asistente virtual sigue la siguiente secuencia:

1. Se lee el código introducido por el usuario en la web del asistente virtual.
2. Se comprueba si el código proporcionado coincide con algún código de la base de datos y si hay respuestas asociadas al código introducidas por los profesores del centro a través de otro software.
3. Si el código no existe o el código existe, pero no hay respuestas asociadas:
   (a) El asistente virtual avisa que sus respuestas no están introducidas en la base de datos y finaliza el proceso.
4. Si el código existe, pero ya existe una conversación previa entre el asistente virtual y el usuario:
   (a) El asistente virtual avisa que sus opiniones han sido registradas con éxito y finaliza el proceso.
5. Si el código proporcionado por el alumno existe, las respuetas están registradas y tiene asociado un tipo de flujograma:
   (a) El asistente virtual proporciona un mensaje de bienvenida y comienza el flujo de preguntas y respuestas por parte del usuario.
   (b) El backend envía al microservicio de Python el mensaje recibido por parte del usuario.
   (c) El microservicio de Python procesa la consulta del usuario y devuelve el siguiente nodo del flujograma.
   (d) El backend lo envía al frontend y se visualiza en el plugin de la ventana de chat del asistente.
   (e) Este proceso se repite hasta que se finalice el flujo de preguntas y respuestas.
6. El usuario tiene la opción de evaluar la labor del asistente virtual.
7. El asistente virtual se despide y se cierra el proceso.

Si el usuario volviera a repetir el proceso de evaluación, el asistente virtual le avisaría, por lo que este proceso únicamente se puede ejecutar una vez en su totalidad. De este modo se evita que un usuario replique sus respuestas y altere la veracidad de las opiniones.

## 4.4. Código fuente

En esta sección se desglosa el código fuente del programa. Se dividen en tres partes aunque en el entorno de producción los archivos de Python se encuentran en el mismo esqueleto que los archivos del backend de node.js. Por otro lado, al utilizar archivo de compilación del frontend, el frontend se reduce a la carpeta /dist. Se expondrá la estructura utilizada en el desarrollo de la aplicación:

### 4.4.1. Backend

En el backend se encuentran los siguientes directorios compuestos de los ficheros:

- ./config: Directorio donde se encuentra la configuración del proyecto.
  - db.config.js: Archivo que tiene la configuración de la base de datos y sus credenciales.
- ./controllers: Directorio donde se encuentran los controladores (métodos de la API), encargados de unir el Model y View.
  - chatbot.controller.js: Archivo que se compone de la lógica para discernir si el código introducido por el usuario está registrado, tiene conversación asociada o tiene que llamar al Python para comenzar con el flujo de la conversación.
  - tipo.controller.js: Archivo que en base a los resultados de las encuestas escritos en clase y registradas por el centro, calcula el tipo de flujograma que debe seguir el usuario.
- ./dist: Compilación del frontend
- ./models: Directorio donde se encuentran los modelos utilizados para la subida de datos a la base de datos:
  - chat.model.js: Archivo que tiene la estructura de la tabla de la base de datos correspondiente con el chat.
  - tipo.model.js: Define el tipo de los usuarios
  - user.model.js: Define el usuario correspondiente por el código proporcionado
- ./routes: Directorio con los métodos HTTP (GET, POST, PUT, DELETE) para la comunicación.
- ./node-modules: Directorio con las librerías utilizadas.
- ./python: Directorio con los archivos para montar el microservicio de Python, véase 4.4.2.
- server.js: Archivo que crea el servidor.
- app.js: Archivo que contiene la conexión con la base de datos y se definen los modelos y las rutas.
- socket.js: Archivo que establece la comunicación con el microservicio de Python (lógica) mediante socket.io.
- package.json: Información del proyecto así como enumeración de las dependencias utilizadas.

### 4.4.2. Python

En el microservicio de Python se encuentran los siguientes archivos:

- chatbot.py: Archivo "__main__" que ejecuta el servicio de Python. Contiene los métodos para la comunicación con Node.js.
- chatbot_h.py: Archivo que tiene la clase utilizada para definir el flujograma y las funciones lógicas del asistente.
- input_data.pyt: Archivo que tiene la secuencia del flujograma y las respuestas preprogramadas.

### 4.4.3. Frontend

Por parte del frontend, se mencionan los archivos que dan vida al plugin del asistente y a su web estática:

- ./public: Directorio donde se encuentran imágenes utilizadas (icono del plugin del asistente, index.html).
- ./src: Directorio donde se encuentran los componentes de vue.
  - ./assets: Directorio donde se encuentran los assets utilizados.
  - ./components: Directorio con los componentes de Vue.js
    - iconChatbot.vue: Archivo que constituye el plugin del asistente virtual y su ventana de chat.
  - App.vue: Template que lanza la aplicación.
  - main.js: Archivo JavaScript que comunica con la API por socket y renderiza App.vue

## 4.5. Librerías utilizadas

| Librería | Versión utilizada en el proyecto | Versión actual |
|---|---|---|
| bcryptjs | 2.4.3 | 2.4.3 |
| cors | 2.8.5 | 2.8.5 |
| express | 4.18.1 | 4.18.2 |
| jsonwebtoken | 8.5.1 | 9.0.0 |
| jssha | 3.2.0 | 3.3.0 |
| mysql2 | 2.3.3 | 3.0.1 |
| request-promise | 4.2.6 | 4.2.6 |
| sequelize | 6.24.0 | 6.28.0 |
| socket.io | 2.1.1 | 4.5.4 |

Table 4.1: Tabla con las librerías de Node.js utilizadas en este proyecto

Hay que tener en cuenta que el paquete: 'request-promise' se encuentra 'deprecated'.

| Librería | Versión utilizada en el proyecto | Versión actual |
|---|---|---|
| @codekraft-studio/vue-record | 0.0.3 | 0.0.3 |
| @mdi/font | 6.7.96 | 7.1.96 |
| axios | 0.26.0 | 3.5.2 |
| core-js | 3.1.3 | 3.27.2 |
| express | 4.18.1 | 4.18.2 |
| jquery | 3.6.0 | 3.6.3 |
| p5 | 1.4.1 | 1.5.0 |
| vue-icon | 2.2.0 | 2.3.0 |
| vue-p5 | 0.8.4 | 0.8.4 |
| vue-socket.io | 3.0.10 | 3.0.10 |
| vue-sound | 0.0.4 | 0.0.4 |
| vuetify-audio | 0.3.3 | 0.3.3 |

Table 4.2: Tabla con las librerías de Vue.js utilizadas en este proyecto

| Librería | Versión utilizada en el proyecto | Versión actual |
|---|---|---|
| Flask | 2.2.2 | 2.2.2 |
| numpy | 1.21.5 | 1.24.1 |
| joblib | 1.1.0 | 1.2.0 |
| ujson | 5.4.0 | 5.7.0 |

Table 4.3: Tabla con las librerías de Python utilizadas en este proyecto

## 4.6. Problemas encontrados

Con respecto a la implementación, estructura del código y uso de librerías del presente proyecto se han señalado los siguientes problemas:

- Librerías 'deprecated': Hay librerías que se han quedado en una versión, como por ejemplo request-promise de los paquetes utilizados para el backend de Node.js que se encuentran deprecated y ya no está soportada dicha versión.
- Librerías no actualizadas: A pesar de que las versiones utilizadas no se alejan de las versiones actuales dado que el proyecto es actual, hay alguna librería que se utiliza una versión muy antigua, como por ejemplo en el backend 'socket.io' cuya versión actual es 4.5.4 y la utilizada es 2.1.1 y no se encuentra documentado de por qué razón puede ser.

- Librerías no utiliazadas: A lo largo del código hay librerías que se han declarado, pero no se han utilizado. Esto denota poco control sobre el código que se ha desarrollado.
- Ficheros con contraseñas: El fichero db.config.js contiene la contraseña (usuario y contraseña) de la base de datos. Es verdad que se requieren declarar, pero se encuentran dentro del fichero hardcodeadas.
- Código: El código no tiene una estructura seguida por la guía de estilos de cada lenguaje de programación.

## 4.7. Soluciones propuestas

Para los anteriores problemas encontrados, se proponen las siguientes soluciones:

- Librerías 'deprecated': Hay librerías que no están en uso. Se deberían buscar alguna alternativa de estas librerías.
- Librerías no actualizadas: Aquellas librerías que estén desactualizadas deberían actualizarse de forma periódica.
- Librerías no utiliazadas: Se deberían eliminar las referencias de aquellas librerías que están definidas en el código y no se hace uso de ellas.
- Ficheros con contraseñas: Se deberían evitar hardcodear las contraseñas de unión a la base de datos dentro de los ficheros de código.
- Código: El código no tiene una estructura seguida por la guía de estilos de cada lenguaje de programación.

# 5

# Operaciones

## 5.1. Introducción

En la presente sección se realiza un análisis de las operaciones enfocado en el asistente virtual. Para ello se reevisa el estado actual de esta herramienta y se indican buenas prácticas que han de ser tomadas para mejorar la seguridad de la aplicación.

## 5.2. Estado actual

Actualmente, la herramienta se encuentra desplegada aunque no ha sido entregada al cliente. El servicio web está montado en un entorno de producción aunque está en fase de aceptación (pruebas) por parte del cliente.

- Entorno de preproducción: Entorno idéntico al entorno de producción donde se realizan pruebas previas a su despliegue. En este caso no existe dicho entorno y las pruebas se ejecutan en un entorno de desarrollo.
- Entorno de desarrollo: El entorno de desarrollo es un entorno donde los desarrolladores realizan pruebas en local antes de desplegar la aplicación. Se utiliza este entorno como fase previa al entorno de producción.
- Entorno de producción: Entorno real donde se va a desplegar la aplicación. Se está utilizando este entorno cuando se comprueba que la aplicación funciona en el entorno de desarrollo.
- Entorno de pruebas: No existe un entorno destinado a las pruebas. Todas las pruebas se realizan en el entorno de desarrollo a partir de la última versión del entorno de producción.
- Lanzamiento, despliegue y monitorización: No existe ninguna guía para realizar ninguna de las tres acciones. Existe monitorización del asistente, pero no una guía para proceder.

## 5.3. Problemas encontrados y buenas prácticas

En esta sección se comentarán prácticas aconsejables que han sido detectadas a partir de los problemas encontrados en la operatividad de este proyecto:

- Auditorías: Se recomendaría realizar autorías de la aplicación de forma periódica, por los propios participantes o equpo de proyecto y por un agente profesional externo para tener una visión global del proyecto.
- Entorno de preproducción: El entorno de preproducción es inexistente. Se está utilizando el entorno de desarrollo como paso previo a producción y debería existir un entorno de condiciones similares al entorno de producción, de modo que se pueda testear la aplicación previo a su despliegue.

- Manual de programador y guías de despliegue: El proyecto carece de una extensa documentación desde un punto de vista de desarrollo y de despliegue. Si surge un error postproyecto y debe solucionarlo un nuevo integrante del equipo, tendría muy complicada esta labor. No tendría suficiente información de cómo poder desplegar una nueva versión. Como método urgente, se debe crear una guía de despliegue.

# 6

# Automatización

## 6.1. Introducción

En este capítulo se describen procesos de automatización y se realizará un análisisi del código de todas las partes del asistente virtual (backend, frontend y python). Se realizarán test unitarios

## 6.2. Estado actual

Actualmente, el sistema del asistente virtual no tiene ningún tipo de automatización u otro mecanismo de análisis de código.

- Análisis de código: No se ha realizado ningún análisis de código durante el desarrollo ni la la puesta en el entorno de producción para las pruebas del cliente. No se ha barajado incluir una herramienta de análisis de código dentro del proceso de desarrollo.
- Pruebas (integración y funcionales): No se han tenido en cuenta ningún tipo de pruebas, ni funcionales, ni de integración.
- CI/CD: No se hace uso de ningún tipo de softwares o plataformas de integración continua y despliegue continuo. El equipo trabaja con Bitbucket, pero no ha utilizado ningún plugin o software relativo para ello.

## 6.3. Problemas encontrados

Analizando el estado actual de la herramienta se pueden observar los siguientes problemas:

- Pruebas: Dado que el equipo no hace uso de un entorno de pruebas o preproducción, si no que directamente despliega la aplicación del entorno de desarrollo al entorno de producción, no se realizan pruebas sobre el despliegue del sistema. Además, se ha observado que no se realizan pruebas funcionales ni análisis de código durante el desarrollo.
- Malas prácticas: El equipo utiliza Bitbucket como repositorio pero carece de una guía o de un sistema para subir al entorno de produccion. No se hace uso de herramientas de integración continua o despliegue continuo.
- Calidad del software: Se ha realizado un análisis estático del repositorio con el sistema del asistente mediante SonarCloud. Este análisis nos ofrece una visión global del estado actual del proyecto.

Para el análisis estático y tests se ha utilizado SonarCloud. En la figura 6.1 se puede observar el resumen del análisis y, teniendo en cuenta que es un proyecto que se encuentra desplegado en un entorno de producción, destaca que no se han encontrado ningún error de seguridad (si revisiones), pero si se han encontrado 2 bugs importantes (en el icono del frontend del asistente virtual y en un método de /controller).

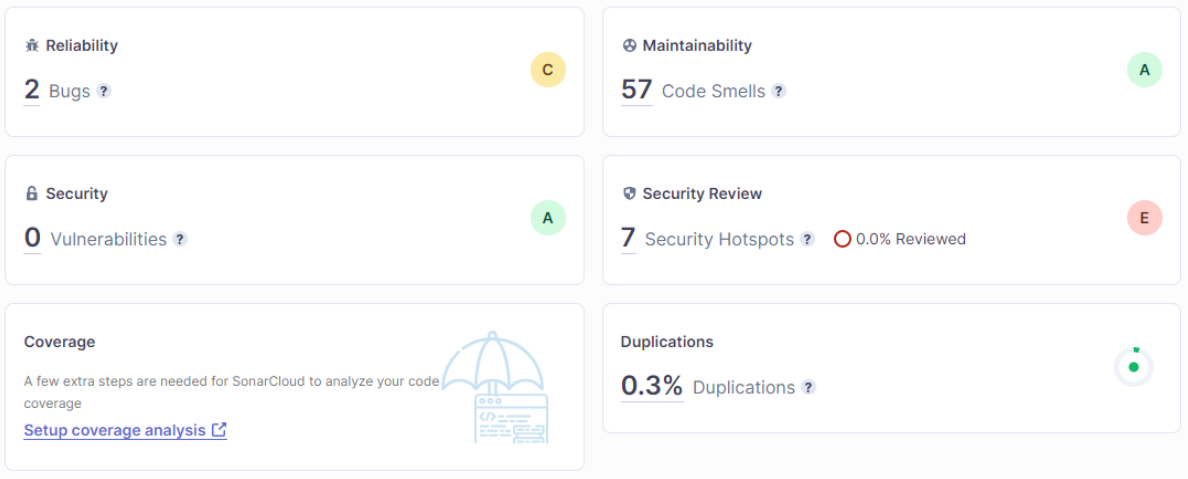


Figure 6.1: Resumen del análisis realizado por SonarCloud

En la figura 6.2 se pueden ver en detalle los dos bugs mencionados:

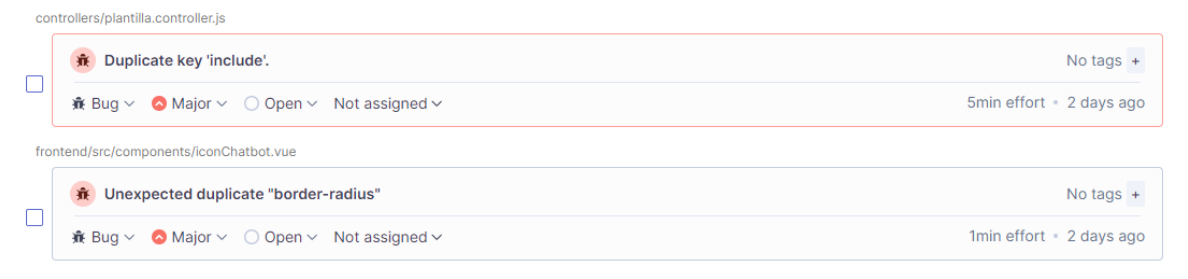


Figure 6.2: Ejemplos de bugs en el proyecto

A pesar de que el análisis en SonarCloud no ha encontrado grandes problemas o vulenerabilidades en el código desde un punto de vista de seguridad, es importante destacar que todo este desarrollo se ha realizado sin haber tenido en cuenta estas herramientas y por errores humanos se pueden generar vulnerabilidades. Este tipo de análisis prorpociona un apoyo.

En una vista más detallada por carpetas (estructura) del proyecto, se pueden observar la distribución de estas vulnerabilidades detectadas en el código. Destaca la existencia de código duplicado sobre todo en el backend.

| | Lines of Code | Bugs | Vulnerabilities | Code Smells | Security Hotspots | Coverage | Duplications |
|---|---|---|---|---|---|---|---|
| dps-project | | | | | | | |
| config | 16 | 0 | 0 | 0 | 0 | — | 0.0% |
| controllers | 880 | 1 | 0 | 29 | 1 | — | 7.0% |
| dist | 1 | 0 | 0 | 0 | 1 | — | 0.0% |
| frontend | 829 | 1 | 0 | 11 | 3 | — | 0.0% |
| models | 180 | 0 | 0 | 1 | 0 | — | 0.0% |
| python | 117 | 0 | 0 | 3 | 0 | — | 0.0% |
| routes | 73 | 0 | 0 | 4 | 0 | — | 0.0% |
| tests | 368 | 0 | 0 | 2 | 1 | — | 0.0% |
| app.js | 66 | 0 | 0 | 2 | 1 | — | 0.0% |
| package-lock.json | — | 0 | 0 | 0 | 0 | — | 0.0% |
| package.json | — | 0 | 0 | 0 | 0 | — | 0.0% |
| server.js | 6 | 0 | 0 | 0 | 0 | — | 0.0% |
| socket.js | 131 | 0 | 0 | 5 | 0 | — | 18.4% |

Figure 6.3: Detalle del código con las vulnerabilidades obtenidas por SonarCloud

## 6.4. Soluciones propuestas

Teniendo en cuenta los problemas mencionados y el estado actual, se proponen las siguientes medidas:

- Pruebas (integración y funcionales): El equipo de desarrollo debería realizar pruebas a corto plazo ya que ahora mismo no se desarrollan.
- Análisis de la calidad del código: Se propone que se realicen buenas prácticas de análisis de código para comprobar vulnerabilidades que puedan surgir durante el desarrollo y despliegue del proyecto.
- Integración y Despliegue continuo (CI/CD): Se propone utilizar herramientas destinadas a este propósito. Estas herramientas ayudan al equipo de desarrollo y son un apoyo a la hora de detectar vulnerabilidades, además que son buenas prácticas.

# 7

# Conclusión

Teniendo en cuenta el presente documento de análisis de seguridad del asistente virtual, se pueden observar varios puntos a tener en cuenta para mejorar el desarrollo realizado. En el presente documento se ha analizado desde un punto de vista de la seguridad del desarrollo: Arquitectura, Diseño, Implementación, Operaciones y Automatización y Tests. En cada sección se han observado posibles riesgos y soluciones propuestas para subsanar y minimizar el riesgo del sistema. Entre los riesgos se destacan:

- Correcto desarrollo de código. Se ha visto la necesidad de utilizar guías de estilo para desarrollo de código.
- Revisión de librerías de terceros: Hay librerías que no están actualizadas y se encuentran lejos de la última versión. Evitar este tipo de errores mejora la seguridad del proyecto.
- Falta de archivos logs. Es un sistema que se encuentra en producción y carece de archivos logs para comprobar si hay errores o no en producción.
- Análisis de código y pruebas.
- No hay entorno de preproducción.

A nivel personal, el informe de seguridad ha ayudado para comprobar la seguridad del proyecto, así como utilizar los conceptos aprendidos en un proyecto real. La valoración es muy positiva y se volverá a aplicar en el resto de proyectos que tengo dentro de la empresa. Para concluir, el objetivo de la práctica ha sido completado: tanto analizar el código como conocer y aplicar esta metodología a un trabajo real (no es simulación).

# Appendices

# A

# Anexos

## A.1. Cuestionario de seguridad

Se adjunta el cuestionario de evaluación de seguridad para aplicaciones web, VSAQ [8] que ha sido rellenado.

# VSAQ

## Cuestionarios de evaluación de seguridad de proveedores

Cargar respuestas desde archivo:

Elegir archivos Ninguno archivo selec.

### Cuestionario de seguridad de aplicaciones web

#### Aplicación

##### Metadatos de la aplicación

**El nombre de la aplicación:**

Virtual assistant

**Una breve descripción:**

La aplicación web se compone de un asistente virtual desplegado sobre una web estática que permite la recogida de las opiniones de los alumnos de un curso de una manera interactiva, rápida y más detallada. Este proceso del asistente forma parte de un sistema mayor donde:
1. Los alumnos acuden al centro y realizan el curso.
2. Tras la finalización del curso, los profesor dan un cuestionario a los estudiantes para que evalúen el curso, su grado de satisfacción, al instructor y el centro. Los profesores les proporcionan un código QR con un enlace y un código para que los alumnos puedan acceder a la web del asistente.
3. En función de las respuestas de los alumnos, los profesores clasifican a estos alumnos (de forma anónima, a través del código proporcionado) en un tipo de alumno (tipo 0, 1, 2 y 3). Estos tipos representan el camino del flujograma que debe seguir el asistente virtual cuando el alumno acceda a conversar.
4. Los alumnos acceden a la web y conversan con el asistente. El asistente únicamente se inicia si el alumno ha introducido su código y es correcto.
5. El asistente detecta el tipo del alumno que es y lanza un flujograma específico para extraer una opinión más detallada.

**¿Qué marcos (si los hay) requiere esta aplicación?**

Para el backend se utiliza Node.js (JavaScript) y microservicio de Python (mediante el framework Flask). Como base de datc

##### Informes y gestión de vulnerabilidades

**Debido a que ningún sistema está completamente libre de problemas de seguridad, es importante proporcionar formas para que los usuarios externos ofrezcan información y reporten vulnerabilidades.**

**¿Tiene una forma fácil de descubrir para que los investigadores externos informen las vulnerabilidades de seguridad en sus sistemas?**

○ Sí, tenemos un contacto de correo electrónico de seguridad publicado o proporcionamos otra forma para que los usuarios informen problemas de seguridad. Los informes entrantes se revisan y clasifican oportunamente.

◉ No, actualmente no ofrecemos una forma de informar vulnerabilidades de seguridad para el manejo prioritario.

**Advertencia: posible problema de riesgo medio**

Facilite que otros le informen sobre problemas de seguridad en sus productos. De esa manera, aprenderá antes sobre las vulnerabilidades y podrá responder a ellas rápidamente. Además, sin una manera fácil de informarle directamente sobre los problemas, los investigadores externos podrían publicar los problemas ampliamente.

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación. Si está trabajando para abordar este problema, incluya una estimación de cuándo se*

*resolverá.*

### Riesgos de HTTPS y contenido mixto

**Seleccione la opción que mejor describa su aplicación web:**

- (●) La aplicación web es accesible exclusivamente a través de HTTPS. Incluso si el usuario edita manualmente la URL para comenzar con `http://`, no funcionará o se redirigirá a `https://`.
- ( ) La aplicación web es flexible: los usuarios pueden acceder a ella a través de HTTP o HTTPS.
- ( ) La aplicación web solo admite HTTP y no se puede acceder a través de HTTPS, incluso si edita la URL.

### Configuración de SSL/TLS

**¿Revisó recientemente su configuración SSL para asegurarse de que solo se ofrezcan cifrados y protocolos seguros a los clientes?**

- (●) Sí, revisamos periódicamente el paquete de cifrado anunciado por el servidor y los protocolos que utiliza.
- ( ) No estamos seguros de si nuestra configuración SSL/TLS es segura.

**¿Su servidor ofrece confidencialidad directa para los clientes que lo admiten?**

- (●) Sí, el servidor admite cifrados ECDHE y DHE que ofrecen confidencialidad directa.
- ( ) No, no se habilitan cifrados que proporcionen confidencialidad directa.

**¿Están sus claves privadas SSL/TLS debidamente protegidas en sus servidores web?**

- (●) Sí, hemos tomado todas las medidas necesarias para proteger nuestras claves privadas.
- ( ) No estoy seguro de lo bien protegidos que están.

**¿Dónde finaliza la conexión SSL entre el usuario y su aplicación?**

- (●) En el servidor de aplicaciones
- ( ) En el balanceador de carga
- ( ) En algún otro lugar

**Las aplicaciones servidas a través de SSL aún pueden ser vulnerables a los ataques si los recursos (a menudo JavaScript, hojas de estilo u otro contenido activo) se incluyen a través de HTTP simple. Esto anula el propósito de SSL, porque el contenido activo cargado a través de HTTP simple tendrá acceso al DOM del contenido protegido por SSL. Asegúrese de que no se incluyan recursos de sitios HTTP simples. Por lo general, los navegadores ayudarán a identificar los casos en los que se incluyen recursos de sitios que no son SSL, mostrando advertencias de contenido mixto .**

**Para evitar estos problemas, ¿tiene comprobaciones para garantizar que todas las referencias a los recursos apunten a SSL o sean relativas al protocolo?**

- (●) Sí, somos muy cuidadosos y contamos con controles específicos para evitar problemas de contenido mixto.
- ( ) No sería muy difícil que algo pasara desapercibido e introdujera errores de contenido mixto.

**Para mejorar aún más la seguridad de sus usuarios, ¿ha implementado HTTP Strict Transport Security (HSTS) en su servidor?**

- ( ) Sí, tenemos HSTS configurado con un valor de edad máxima de al menos 6 meses.
- (●) No, no usamos HTTP Strict Transport Security, o tenemos configurado un período máximo corto.

**Advertencia: posible problema de alto riesgo**

Strict Transport Security es un encabezado de respuesta HTTP que les dice a los clientes que inicien conexiones solo a través de HTTPS. Este mecanismo de defensa en profundidad ayuda a mitigar ciertos problemas, como la inclusión accidental de (o la vinculación a) recursos a través de HTTP simple. En HSTS, el encabezado debe configurarse con un valor de 'edad máxima'; esto les dice a los clientes cuánto tiempo deben respetar la directiva.

Las aplicaciones web confidenciales deben habilitar HSTS para garantizar que los usuarios estén siempre protegidos por SSL/TLS. Para que esta protección sea efectiva, el valor de edad máxima debe estar en el rango de más de 6 meses.

**Si su aplicación admite la autenticación, ¿las cookies de autenticación están marcadas con el `secure`atributo?**

- ○ Sí, las cookies de autenticación están marcadas `secure`.
- ◉ No estoy seguro de si la aplicación hace eso.

**Advertencia: posible problema de riesgo medio**

A menos que una cookie tenga el `secure`atributo establecido, el navegador agregará la cookie a las solicitudes HTTP de texto sin formato, incluso si la aplicación es solo SSL. Un ataque man-in-the-middle puede usar esta vulnerabilidad para robar cookies de autenticación.

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación.*

## Autenticación y Autorización

### Información básica

**Para comenzar, cuéntenos un poco acerca de su aplicación para que podamos hacerle las preguntas correctas.**

- ☐ Nuestra aplicación requiere que los usuarios regulares inicien sesión. La mayoría de las funciones no están disponibles sin iniciar sesión.
- ☐ Además de una interfaz para usuarios habituales, nuestra aplicación proporciona una interfaz de administración.
- ☐ Nuestra aplicación presenta una gestión de usuarios compleja. Se pueden asignar varios roles a las cuentas de usuario.

## Vulnerabilidades web comunes

**Ciertas funciones pueden generar problemas de seguridad si se usan incorrectamente. Para ayudarnos a identificar problemas potenciales, seleccione las declaraciones que describen su aplicación:**

- ☑ La aplicación utiliza un back-end de base de datos o cualquier otro back-end de persistencia que se pueda consultar con SQL o un lenguaje relacionado (p. ej., GQL, FQL, SOQL, etc.).
- ☐ La aplicación requiere un complemento, como Java, Flash, Silverlight, etc.
- ☐ La aplicación tiene una función de carga de archivos.
- ☐ La aplicación carga contenido activo, como secuencias de comandos, subprogramas u hojas de estilo, desde servidores de terceros (es decir, cualquier servidor que no esté bajo su control directo).
- ☐ La aplicación procesa o manipula el XML proporcionado por el usuario.
- ☐ La aplicación utiliza criptografía para cifrar datos o proteger su integridad.

### Secuencias de comandos entre sitios



**Las secuencias de comandos entre sitios (o XSS para abreviar) se producen cuando una aplicación vuelve a mostrar una entrada de usuario insuficientemente desinfectada en el contexto del origen de la aplicación**

**(como se define en la política del mismo origen ). Si la entrada del usuario contiene ciertos tipos de código de secuencias de comandos, puede leer o alterar el DOM de la página actual cuando se vuelve a mostrar. En muchos casos, XSS se usa para robar las cookies de los usuarios u otros datos relacionados con la aplicación, pero también se puede usar para ataques de phishing o incluso para desfigurar la página web. Desafortunadamente, XSS es uno de los problemas de seguridad más comunes en las aplicaciones web y, debido a las peculiaridades del navegador y otros factores, es bastante difícil protegerse. Seleccione las declaraciones que describen su estrategia:**

- ☐ Utilizamos un sistema de plantillas que escapa automáticamente a todas las entradas del usuario antes de volver a mostrarlo.
- ☐ Nuestra aplicación tiene un cuello de botella central donde todas las entradas del usuario se validan y escapan, según el contexto en el que se interpretarán.
- ☐ Algunas de las páginas (o todas ellas) escapan a la entrada del usuario.
- ☐ Estamos utilizando alguna otra técnica para proteger contra XSS.
- ☑ Parte de la aplicación trata con HTML proporcionado por el usuario que se desinfecta y se vuelve a mostrar al usuario.

**Advertencia: posible problema de riesgo medio**

Desafortunadamente, debido al comportamiento de corrección de errores y las peculiaridades del navegador, es muy difícil obtener el saneamiento de HTML correcto. El código de desinfección tiene que crear una representación en memoria del DOM y luego serializarla en un formato seguro conocido. Algunas bibliotecas hacen esto correctamente, por lo que recomendamos usar una biblioteca bien probada.

*Describa cómo su aplicación trata con el saneamiento de HTML:*

**Además de aplicar las estrategias que ha identificado, ¿establece la aplicación un tipo de contenido y un conjunto de caracteres válidos y apropiados para cada página (en el `Content-Type`encabezado HTTP)?**

- ◉ Sí, tenemos mucho cuidado al establecer esto, sabiendo que de lo contrario podríamos estar introduciendo vulnerabilidades XSS.
- ○ No estoy seguro de que todas las páginas establezcan un tipo de contenido adecuado.

**Algunas vulnerabilidades XSS funcionan exclusivamente en el lado del cliente, en el código de secuencias de comandos de una aplicación. Este tipo de XSS se conoce comúnmente como XSS basado en DOM . Debido a que el escape del lado del servidor de la entrada del usuario no protege contra XSS basado en DOM, necesita una estrategia para manejar el código de secuencias de comandos del lado del cliente que maneja la entrada del usuario, así como partes del DOM que pueden contener la entrada del usuario (como documento .localización).**

- ○ Conocemos XSS basado en DOM y tomamos medidas específicas para protegernos contra este tipo de vulnerabilidad.
- ◉ Es posible que algo se haya escapado y nuestra aplicación tenga vulnerabilidades XSS basadas en DOM.

**Advertencia: posible problema de riesgo medio**

Recomendamos que audite minuciosamente su código en busca de vulnerabilidades XSS basadas en DOM y establezca procedimientos para que el código futuro también esté protegido.

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación. Si está trabajando para abordar este problema, incluya una estimación de cuándo se resolverá:*

## Back-ends de persistencia y consultas

**Debido a que su aplicación utiliza una base de datos o un back-end similar para conservar los datos, debemos asegurarnos de que no sea vulnerable a los ataques de inyección, como la inyección SQL .**

**Una aplicación es vulnerable a la inyección SQL cuando la base de datos interpreta una parte de la entrada del usuario como parte de una consulta. Cuando esto ocurre, un atacante puede leer o incluso escribir datos directamente desde o hacia la base de datos.**

- [ ] Nuestra aplicación utiliza un marco de mapeo relacional de objetos (ORM). Cuando necesitamos construir manualmente consultas o condiciones, usamos uno de los mecanismos seleccionados a continuación:
- [x] Usamos declaraciones preparadas y dejamos que el marco se encargue de escapar correctamente de la entrada del usuario.
- [ ] Pasamos la entrada del usuario a la base de datos a través de procedimientos almacenados.
- [x] Escapamos manualmente de la entrada del usuario cada vez que necesitamos usarla en una consulta de base de datos.
- [ ] Hacemos otra cosa.

**Advertencia: posible problema de riesgo medio**

El escape manual de consultas SQL (o relacionadas) es propenso a errores y muy difícil de hacer de manera consistente. Aquí hay un par de ejemplos:

- PHP proporciona la función `mysql_escape_string`para escapar de la entrada del usuario que se utilizará en una consulta de base de datos. Desafortunadamente, esa función no tiene en cuenta el conjunto de caracteres de la conexión, por lo que aún es posible pasar de contrabando la entrada del usuario que se interpretará como parte de la consulta SQL. Las aplicaciones deberían usar en su `mysql_real_escape_string`lugar.
- En SQL, los números no necesitan estar entre comillas cuando se usan en una declaración. Por ejemplo, `SELECT username WHERE id=123`es perfectamente válido. Pero si no se confirma que la entrada proporcionada por el usuario que se usará como un número sea realmente numérica, el código resultante será vulnerable a la inyección SQL (incluso si la entrada tiene escape).

Recomendamos encarecidamente usar algo como declaraciones preparadas o usar una capa ORM de manera consistente en toda la aplicación. Asegúrese de tener procedimientos establecidos para hacer cumplir su enfoque (como pruebas estáticas cuando el código se registra en el repositorio).

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación. Si está trabajando para abordar este problema, incluya una estimación de cuándo se resolverá:*

## Pruebas, QA y Monitoreo

**Las pruebas de seguridad pueden ser parte de las pruebas de aplicación estándar. Aquí hay unos ejemplos:**

- **Pruebas unitarias simples: las pruebas unitarias se utilizan normalmente para confirmar que los componentes básicos de la aplicación funcionan como se esperaba. Las pruebas unitarias son fáciles de repetir: se pueden ejecutar cada vez que se registra un nuevo código en el repositorio, para confirmar que el código aún se comporta como se esperaba. Las pruebas unitarias también pueden verificar las características de seguridad. Por ejemplo, se pueden usar para confirmar que las solicitudes fallan sin tokens XSRF; que se requiere autenticación para acceder a los datos del usuario; o que las etiquetas HTML inesperadas no pueden atravesar los filtros de entrada o escapar de las rutinas.**

- **Pruebas de lanzamiento: antes de que se lance una nueva versión de un producto, los evaluadores humanos generalmente revisan la aplicación, prueban las nuevas funciones y se aseguran de que las funciones anteriores aún funcionen correctamente (pruebas de regresión). Las pruebas de seguridad también deben incluirse en este proceso. Por ejemplo, las pruebas de lanzamiento son un buen momento para verificar que el usuario A no puede acceder a los datos del usuario B.**
- **Monitoreo: una vez que se implementa la aplicación, el enfoque generalmente cambia de la prueba al monitoreo. Tenga cuidado con los picos inesperados en las tasas de error, las infracciones de sandbox y otros comportamientos irregulares o inexplicables (incluidos los errores de prueba intermitentes), y antes de descartar una anomalía, consulte con su equipo de seguridad. Los bloqueos y la descamación pueden indicar una condición de carrera o un error de corrupción de memoria.**

**Las siguientes preguntas evalúan las pruebas y el seguimiento de su aplicación.**

**¿Está utilizando pruebas unitarias o métodos similares?**

○ Sí
◉ No

**¿Sus ingenieros y su equipo de control de calidad buscan posibles problemas de seguridad durante las pruebas de lanzamiento y han sido capacitados para hacerlo?**

○ Sí, nuestro proceso de control de calidad incluye explícitamente pruebas de problemas de seguridad que podrían haberse introducido en la nueva versión.
◉ Esta es un área en la que tenemos margen de mejora.

**Advertencia: posible problema de riesgo medio**

Sus equipos de ingeniería y control de calidad están en la mejor posición para comprender cómo funcionan todas las partes de la aplicación, qué ha cambiado desde la iteración anterior y cómo los cambios pueden introducir vulnerabilidades de seguridad. Las pruebas de lanzamiento generalmente se realizan bajo una presión de tiempo considerable, pero es la última oportunidad de detectar vulnerabilidades de seguridad internamente. Cuando sus equipos ya están enfocados en las pruebas, agregar algunas pruebas de seguridad no aumentará mucho el esfuerzo.

Los ingenieros y probadores que han sido capacitados para buscar problemas de seguridad pueden marcar la diferencia entre un producto seguro y una vulnerabilidad grave.

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación. Si está trabajando para abordar este problema, incluya una estimación de cuándo se resolverá:*

**Monitoreo Post-Lanzamiento**

**¿Cómo describiría su seguimiento posterior al lanzamiento?**

○ **Sólido** : contamos con procedimientos para registrar y monitorear bloqueos inesperados, excepciones y otras condiciones de error. Si algo parece sospechoso, un ingeniero preocupado por la seguridad lo evalúa.
○ **Débil** : si algo sale terriblemente mal, como picos masivos en las tasas de accidentes u otras anomalías a gran escala, probablemente nos demos cuenta. Pero nuestro monitoreo es bastante tosco y hay espacio para mejorar.
◉ **Inexistente** : por el momento, no estamos realizando ningún tipo de seguimiento posterior al lanzamiento que busque signos de explotación o aumentos en bloqueos/excepciones.

**Advertencia: posible problema de riesgo medio**

Las excepciones y los bloqueos a menudo indican un problema de seguridad subyacente. El monitoreo de la aplicación implementada puede contribuir en gran medida a identificar rápidamente y corregir posteriormente las vulnerabilidades.

En productos de software cuidadosamente diseñados, las excepciones deberían ser una ocurrencia bastante rara, por lo tanto, generalmente no introduce una sobrecarga significativa para monitorearlos.

*Si tiene controles de compensación establecidos o cree que este problema no constituye un riesgo en sus circunstancias específicas, explique a continuación. Si está trabajando para abordar este problema, incluya una estimación de cuándo se resolverá:*

**Notas adicionales**

**Proporcione cualquier información adicional sobre la seguridad de su aplicación:**

No se realizan Tests en la aplicación.
No se utilizan herramientas de CI/CD.

**Contactos de seguridad**

**Enumere las direcciones de correo electrónico de las personas a las que debemos contactar sobre cualquier problema de seguridad en la aplicación:**

**Comentarios**

**¡Felicidades! Has llegado al final de este cuestionario. Si tiene otro minuto, háganos saber cómo podemos mejorarlo. Su retroalimentación es muy apreciada.**

Estado:Borrador guardado

**Descargar respuestas** **Restablecer cuestionario**

# Bibliography

[1] Normas iso: Iso27001, –. URL https://normaiso27001.es/.

[2] Mysql community downloads, –. URL https://dev.mysql.com/downloads/mysql/.

[3] Vue.js style guide, –. URL https://v2.vuejs.org/v2/style-guide/?redirect=true.

[4] Security architecture and design, 2017. URL https://upload.wikimedia.org/wikipedia/commons/5/51/Security_Architecture_and_Design.pdf.

[5] Guido van Rossum <guido at python.org> David Goodger <goodger at python.org>. Pep 257 - docstring conventions, 2001. URL https://peps.python.org/pep-0257/.

[6] The OWASP Foundation. Owasp secure coding practices - quick reference guide, 2010. URL https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf.

[7] Felix Geisendörfer. Node.js style guide, 2007. URL https://felixge.de/.

[8] Google. Vsaq - vendor security assessment questionnaires, –. URL https://vsaq-demo.withgoogle.com/vsaq.html?qpath=questionnaires/webapp.json.

[9] Wang S. H. & Chen D. Samadhiya D. Quaility models - role and value in software engineering. *IEEE*, 2010. URL https://ieeexplore.ieee.org/abstract/document/5608852/?casa_token=o_Pa9-kJAUsAAAAA:Y12uYQZxCV-Pww7eUtBzRhJw8I7OwC8kMXBCYLpHTa3-HCmrE6rWFgBGqEuBepN3Kf3UAz10hNQ.

[10] Juan Delfín Peláez Álvarez. *Análisis de Riesgos - Universidad de Leon.* 2022.