Chatbot para docencia: herramienta de obtención de información y realización de cuestionarios para la docencia

César Valcarce Pagán

Resumen: En este proyecto se va a realizar el desarrollo de un chatbot para el alumnado de la UAB. Este chatbot permitirá al alumno buscar información y realizar cuestionarios sobre las asignaturas de su interés. Esta aplicación está encarada, inicialmente, para el alumnado de la UAB y con ella podrán tener dos principales herramientas: el seguimiento de los eventos de las asignaturas y cuestionarios a modo test con el objetivo de mejorar el rendimiento del alumno. Todo ello mediante una aplicación que permitirá conectarse [5] a una base de datos o bien obtener información de una URL donde se aloja todo el material de interés para el alumno. Como posibles funcionalidades secundarias se valora desarrollar: un sistema de logros para la motivación del usuario (gamificación), el registro de cada usuario mediante tres letras y un sistema de verificación de identificación para garantizar que los asistentes del bot son alumnos de la asignatura en cuestión.

Palabras clave. chatbot, restdb, flow.ai, cuestionario, información, get, post, put.

Abstract: In this project, the development of a chatbot for UAB students will be carried out. This chatbot will allow the student to search for information and take questionnaires about the subjects of their interest. This application is aimed, first, for the UAB students, with it you can have two main tools: the monitoring of the events of the subjects and quizzes in order to improve the student's performance that will be able to connect [5] to a database or obtain information from a URL where all the material of interest to the student is housed. As possible secondary functionalities, it is valued to develop: an achievement system for user motivation (gamification), the registration of each user through three letters and an identification verification system to guarantee that the bot assistants are students of the subject in question .

Index Terms: chatbot, restdb, flow.ai, quiz, information, get, post ,put.

1 INTRODUCCIÓN

Los chatbots son herramientas que tratan de proveer al usuario de la información que éste solicita. Mediante la reacción a los inputs que el usuario introduce, el chatbot va guiando al usuario con instrucciones y ofrece opciones que ayuden a encontrar la información que se solicita. Esta aplicación está encarada, inicialmente, al alumnado de la UAB, con ella podrán tener dos principales herramientas: el seguimiento de los eventos de las asignaturas y cuestionarios a modo test con el objetivo de mejorar el rendimiento del alumno.
Todo ello mediante una aplicación que permitirá la conexión [5] a una base de datos o bien obtener información de una URL donde se aloja todo el material de interés para el alumno.

Como posibles funcionalidades secundarias se valora desarrollar: un sistema de logros para la motivación del usuario (gamificación), el registro de cada usuario mediante tres letras y un sistema de verificación de identificación para garantizar que los usuarios que hacen uso del bot son alumnos de la asignatura en cuestión.

- *E-mail de contacto: cesareo.valcarce@e-campus.uab.cat*
- *Mención realizada: Enginyeria del Software*
- *Trabajo tutorizado por: Felipe Lumbreras*
- *Curs 2020/21*

1.1 Plataformas de interés

A la hora de elegir una plataforma debemos tener en cuenta diversos factores: las herramientas disponibles, el precio de estas y el método de trabajo, ya sea únicamente código o mediante interfaz gráfica. A continuación se citan algunas plataformas interesantes a tener en cuenta.

1.1.1 DialogFlow

Dialog Flow: herramienta proporcionada por Google en la que podremos crear, de forma gratuita, un agente con el que el usuario interactúa. Este chatbot consiste en asignarle "Intents" o acciones que el chatbot tendrá registradas para poder devolver una respuesta a la petición del usuario mediante chat de voz.

Por ejemplo: hay un conjunto de palabras que Google tiene registradas que el agente interpreta como un saludo de bienvenida agrupadas en una lista llamada "Welcome". Deberemos asignarle un conjunto de respuestas que serán las que el agente devuelve al usuario para devolverle el saludo.

1.1.2 BotFather

BotFather: bot configurable desde la plataforma de Telegram, de fácil accesibilidad. El bot mostrará información de interés según la asignatura escogida por el usuario. Este usuario, además, deberá poder consultar información mediante comandos que podrán ser consultados mediante "/help".

1.1.2 Flow.ai

Flow.ai: servicio de desarrollo de flujos con soporte para empresas externas como Whatsapp, Telegram, Facebook, entre otros. Mediante una interfaz gráfica se pueden crear flujos que simularán una conversación entre el usuario y el programa.

1.2 Ventajas y desventajas

Dialog Flow: es una herramienta muy potente capaz de analizar y desmenuzar una frase del usuario de forma que puede extraer horas, lugares, días, entre otros para obtener con exactitud la petición del usuario. Esto puede resultar muy útil a la hora de realizar una query a una base de datos ya que fácilmente podemos obtener datos de relevancia que nos ayuden a filtrar.

Por otra parte, veo que la integración del uso de los comandos de voz puede resultar incómodo para el objetivo de este trabajo, por lo que me interesaría encontrar una herramienta que no necesite de gran aportación por parte del usuario para que este pueda conseguir toda la información que necesita, además, el usuario debe estar dispuesto a descargar mi aplicación, lo cual implica una dificultad añadida a la accesibilidad.

BotFather: herramienta que informará automáticamente al usuario de eventos cercanos a la fecha en la que se encuentre. Adicionalmente, el usuario podrá, mediante un listado de comandos, solicitar información al bot, que este consultará en la base de datos o bien directamente desde la URL donde esté contenida la información solicitada. Destacar la accesibilidad ya que un gran porcentaje de alumnos disponen de Telegram, lo que facilita la aceptación de este proyecto. Tiene la capacidad de implementación de cuestionarios para que el alumno pueda mejorar sus conocimientos de forma cómoda, sin necesidad de acceder a sus apuntes.

Por otro lado, no se contempla la posibilidad de utilizar el bot mediante chat de voz, lo que limita la usabilidad.

Flow.ai: gracias a su interfaz gráfica y a su capacidad para desarrollar flujos de forma muy personalizable, se puede mostrar de forma cómoda para el usuario la información que necesite. Una de las principales ventajas del bot es que al poder crear tantos flujos como se desee podemos darle varias funcionalidades en un mismo bot, lo que se podría utilizar para cumplir los objetivos de informar al usuario y proponerle cuestionarios.

Como desventaja, el precio para obtener unas prestaciones adecuadas para escalar a nivel docente es de 19€/mes.

1.3 Justificación de la elección

1.3.1 Flow.ai

Al ser [2] una plataforma compatible con diversos programas de terceros como WhatsApp, Telegram o Twitter, podría llegar fácilmente a más público.

La interfaz gráfica es intuitiva, lo cual ayuda en la curva de aprendizaje para reducir el tiempo del conocimiento de la herramienta y maximizar el tiempo de desarrollo.

Hay una amplia sección de tutoriales [6] para poder obtener conocimientos sobre la implementación de ciertos puntos como puede ser la conexión a la base de datos.

1.3.2 Restdb.io

Haremos uso de la herramienta restdb.io[3], con la cual podemos crear y administrar de forma local (workbench) y remota (phpmyadmin) una base de datos mysql.

2 Definición de objetivos

2.1 Objetivo de la aplicación

Esta aplicación permitirá al alumnado de la UAB efectuar el seguimiento de los eventos de las asignaturas y realizar cuestionarios test con el objetivo de mejorar el rendimiento del alumno en las diversas asignaturas.

Todo ello mediante una aplicación que permitirá conectarse [5] a una base de datos o bien obtener información de una URL donde se aloja todo el material de interés para el alumno.

Como posibles funcionalidades secundarias se valora

desarrollar: un sistema de logros para la motivación del usuario (gamificación), el registro de cada usuario mediante tres letras y un sistema de verificación de identificación para garantizar que los asistentes del bot son alumnos de la asignatura en cuestión.

2.2 Árbol de objetivos y subobjetivos

Objetivos mínimos para completar el proyecto, prioridad alta:

1- Creación e implementación de una base de datos.
2- Correcta conexión entre la base de datos y la aplicación.
3- Flujos de cuestionarios para el usuario [4].
4- Flujos de información para el usuario.

Objetivos secundarios para complementar el proyecto, prioridad media:

1- Visualización por parte de la administración o profesorado del resultado de los cuestionarios realizados de forma masiva .

2- Control de sesiones con el objetivo de restringir el acceso a usuarios no relacionados con la asignatura.

Objetivos opcionales para añadir atractivo al proyecto, prioridad baja:

1- Sistema de gamificación mediante trofeos.
2- Ranking mediante el registro por iniciales.

2.3 Análisis de requerimientos

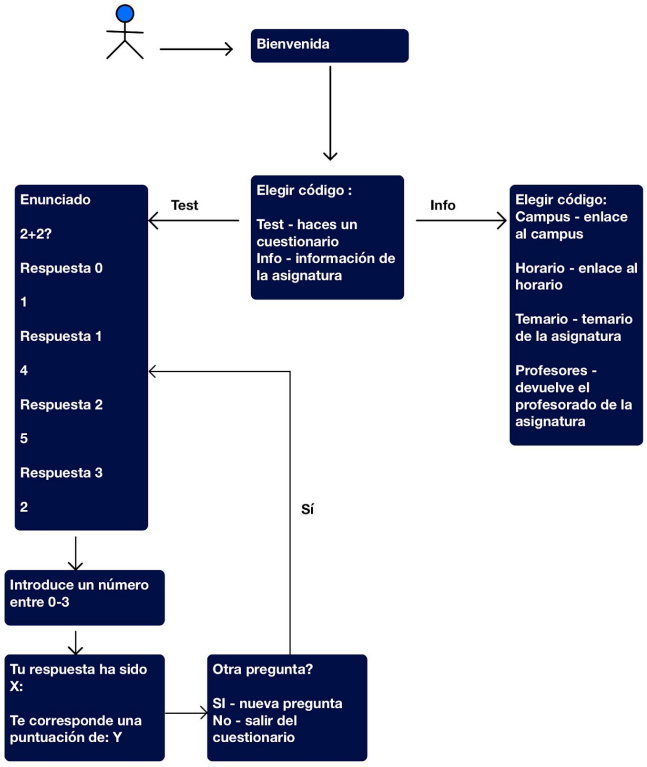


Figura 1. Esqueleto

Herramienta informativa:
- El usuario podrá consultar el horario de las diferentes asignaturas.
- El usuario podrá consultar las entregas de las diferentes asignaturas.
- El usuario podrá consultar las fechas de los exámenes para las diferentes asignaturas.

Herramienta de cuestionarios:
- El alumno puede realizar pequeños cuestionarios que vaya añadiendo el profesor, posteriormente podrá realizar el cuestionario de la sesión completa.
- El alumno puede escoger el tema del que quiere realizar el cuestionario.
- Se ha de mostrar un mensaje explicativo de la respuesta correcta independientemente de la respuesta escogida.

3 Metodología: Scrum

Planificación de iteraciones: se elabora una serie de objetivos a cumplir semanalmente con la finalidad de cumplir con los requisitos citados anteriormente.

Ejecución de las iteraciones: se llevan a cabo las tareas acordadas y se adaptan los requisitos en función de la resolución de estas.

Inspección: el último día de la iteración se revisan las tareas acordadas. Las que no han podido ser completadas son adaptadas desde otro punto de vista que permita alcanzar los requisitos preestablecidos.

3.1 Plataforma utilizada para registrar iteraciones: Trello

Se ha creado un tablero en la aplicación Trello registrando las iteraciones e eventos relevantes y añadiendo una descripción con el contenido de cada uno de estos eventos.

https://trello.com/b/mgvXgFUM/chatbot-para-docencia

4 Planificación

Dado que hay ciertas actividades que no dependen de anteriores, se puede trabajar en paralelo en la creación de la base de datos ([1],[3]) y en la creación de flujos [4].

4.1 Base de datos

Para la creación de la base de datos necesitaremos un medio que nos proporcione el host para alojarla, en este caso se está valorando un servicio gratuito con unas prestaciones suficientes para un uso de demostración.
Una vez creada la base de datos, se procederá al desarrollo del diseño de sus clases y relaciones.
Posteriormente se añadirán los datos necesarios para una

correcta demostración.
Finalmente se procederá a suministrar esta información al usuario dependiendo de la información que se requiera.

4.2 Flujos de información

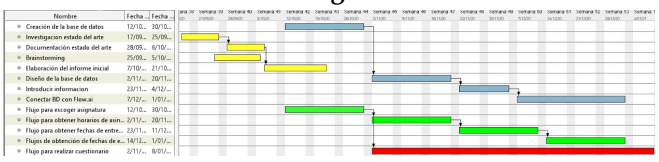
Para aportar la información que el usuario necesita, inicialmente se procederá a obtener la asignatura de interés.

Una vez obtenida la asignatura en cuestión, se preguntará al usuario si quiere información sobre ella o quiere realizar un cuestionario.

En el caso de que quiera realizar un cuestionario se le preguntará sobre qué tema y se procederá con una batería de preguntas que formarán un cuestionario completo.

Por otra parte, si el usuario solicita información, se le proporcionará una serie de opciones para que elija entre: horarios de clase, horarios de entregas y fechas de exámenes.

Tabla 1. Diagrama de Gantt



La Tabla 1 muestra un diagrama en el que se muestra un conjunto de tareas con la finalidad de definir cuales son los milestones o trabajos y en qué margen de tiempo deben cumplirse para no perjudicar a las tareas que dependen de sus predecesores.

5 ESTADO DEL ARTE

Analizaremos las herramientas existentes que nos puedan facilitar el trabajo de desarrollo y ayudar en la captación de ideas para abordar los requerimientos de la aplicación.

5.1 Bot de información

En este apartado se citarán aplicaciones disponibles que su función sea aportar información de interés para el usuario.

Feed Reader Bot

Este bot tiene en parte el objetivo que se persigue en este trabajo: lograr informar al usuario de forma automática de un tema de interés.
Feed Reader Bot permite añadir a canales de Twitter, Youtube e Instagram mediante una URL para mostrar las últimas noticias al usuario.
De esta forma el usuario está informado de cualquier noticia relacionada con un tema de interés propio, pudiendo personalizarse individualmente.

Figura 2. Feed Reader Bot

Mediante el comando /add añadimos el canal de youtube de la escuela de ingeniería de la UAB y este notificará al usuario cada vez que haya una actualización mediante el bot.

5.2 Bot para cuestionarios/encuestas

Se mostrarán ejemplos de chatbots para la generación y ejecución de cuestionarios o encuestas que puedan servir como idea para el desarrollo de este requisito.

Quiz Bot

Este bot permite crear cuestionarios para que los usuarios puedan mejorar sus nociones sobre la asignatura. Estos cuestionarios también pueden ser útiles para el profesorado pues pueden obtener información de los puntos donde deben hacer más hincapié si en un cuestionario falla una cantidad notable de usuarios.

Dado que este bot tiene unas funcionalidades interesantes para la implementación de nuestro proyecto, se han cogido ideas para implementarlas a posteriori, como el registro de test de los usuarios.

5.3 Bases de datos

La implementación de una base de datos para el desarrollo de este proyecto es necesaria ya que en esta se guardan: los enunciados y respuestas con los que los usuarios realizarán los tests, los enlaces y el conjunto de profesores para aportar información sobre la asignatura y el registro de los resultados de los cuestionarios por cada usuario.

Además, es una buena solución para dar escalabilidad al proyecto, ya que con una base establecida se puede, por ejemplo, ampliar la batería de tests. Todo ello de forma segura al no depender de la aplicación (puedes acceder a la base de datos aunque el chatbot no esté operativo).

6 DESARROLLO

En este apartado se van a detallar los procedimientos llevados a cabo para completar los puntos que se mencionan a continuación, estos puntos incluyen partes esenciales de código para simplificar la comprensión de los conceptos.

6.1 Creación e implementación de una base de datos

Para alojar colecciones de datos se hace uso de la herramienta restdb.io [7] una base de datos en la nube que, de forma gratuita, nos permite alojar hasta 100 MB y como máximo cada fichero debe ser de 1 MB, suficiente para cumplir con los requisitos marcados.

Realizamos la importación de un cuestionario proporcionado por el profesor, convirtiéndolo previamente a formato CSV.

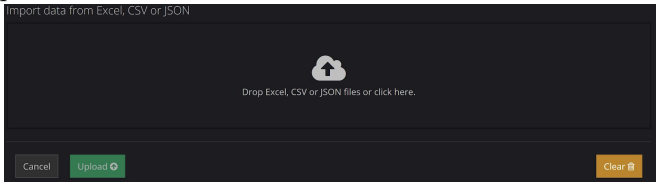


Figura 3 : Importación de .csv

Una vez importado podemos visualizar la colección simplemente haciendo clic en ella.

Figura 4 : Colección cuestionarios

Podemos observar que cada una de las filas de la colección corresponde a un cuestionario que, entre otros parámetros, tiene un enunciado y cuatro posibles respuestas, siendo únicamente una la correcta. Cada una de las columnas viene titulada de forma que podemos entender que contendrá cada uno de estos ítems, siendo por ejemplo "question text text" la columna que guarda los enunciados de cada cuestionario.

6.2 Correcta conexión entre la base de datos y la aplicación

Para establecer una conexión estre la aplicacion Flow.ai y la base de datos restdb.io [7] se han utilizado métodos GET para mostrar información, POST para introducir un determinado valor en una colección de la base de datos y PUT para actualizar una información ya existente, todo esto mediante lenguaje Javascript NodeJS.

Para cualquier interacción con la base de datos, inicialmente hay que especificar a qué base de datos y a qué colección nos vamos a conectar. Todo esto se especifica mediante una URL.

URL:
https://chatbot-efcd.restdb.io/rest/preguntes-vc-2020-tema-1

y le pasaremos la x-apikey que es la "contraseña" para poder hacer modificaciones y obtener contenido de la base de datos:

'x-api key': '51a9ce7dc1f4f5397804be4214eaff58f34d6'

Todos estos parámetros se declaran dentro de una variable que luego utilizaremos para hacer la petición mediante un request.

```
//COGER TODAS LAS PREGUNTAS DEL FORMULARIO
var options = { method: 'GET',
url: `https://chatbot-efcd.restdb.io/rest/preguntes-vc-2020-tema-1`,
headers:
 { 'cache-control': 'no-cache',
   'x-apikey': '51a9ce7dc1f4f5397804be4214eaff58f34d6' } };
```

Figura 5 : Conexión

6.3 Flujos de cuestionarios para el usuario

El objetivo de esta funcionalidad es enviar al usuario una pregunta aleatoria de la colección, que la responda y se le devuelva una respuesta indicando si su respuesta es correcta.

El flujo de este requisito es el siguiente:

Figura 6 : Flow test

Los recuadros de color rosa significan un input por parte del usuario, el programa espera a que se introduzca un valor registrado para comenzar un determinado proceso, en este caso, el valor "test".

Los recuadros oscuros significan procesos automáticos del bot para devolver información al usuario, el primer

caso implica devolver el texto que contiene el primer recuadro.

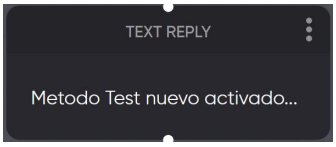


Figura 7 : Nodo salida texto

Seguidamente nos encontramos con otro recuadro oscuro que contiene el nombre de la función RANDOMOK 2.0.

Figura 8 : Nodo llamada función

RANDOMOK 2.0:

El primer paso de esta función es recoger en nodos JSON cada una de las filas de la colección de cuestionarios mediante el método GET.

Una vez recogidos, contamos cuántas filas ha recogido con la extensión .length y creamos un número aleatorio entre 1 y el número total de filas con el objetivo de que ese número aleatorio sea el número de la fila que corresponde a la cuestión que se planteará al usuario.

A continuación obtenemos el nombre del usuario que está haciendo la petición y comprobamos si este existe en una colección llamada "Userandnumber" de nuestra base de datos.

Esta colección "Userandnumber" guarda dos parámetros: el nombre del usuario y el último número que ha obtenido correspondiente a la pregunta que le ha tocado.

Si el usuario existe, se ejecuta un método PUT para sustituir el anterior número por el actual. En caso de que el usuario no exista, se ejecuta un método POST, que añadirá el nombre del usuario, el número aleatorio"${num}" que se le ha generado e inicializará otros atributos que nos servirán para tener una impresión sobre el desempeño que el alumno tiene en la realización de los tests, este punto lo comentaremos más adelante.

PUT para modificar una fila existente

```
var put = { method: 'PUT',
  url: `https://chatbot-efcd.restdb.io/rest/stats/${id}`,
  headers:
   { 'cache-control': 'no-cache',
     'x-apikey': '51a9ce7dc1f4f5397804be4214eaff58f34d6',
     'content-type': 'application/json' },
  body: {  file: `${num}` },
  json: true };
```

Figura 9 : Método PUT

POST para crear una nueva fila

```
var zero = 0
var o = { method: 'POST',
  url: 'https://chatbot-efcd.restdb.io/rest/stats',
  headers:
   { 'cache-control': 'no-cache',
     'x-apikey': '51a9ce7dc1f4f5397804be4214eaff58f34d6',
     'content-type': 'application/json' },
  body: { name: `${username}`, file: `${randomquestion}`, ntests: `${zero}`,
    ncorrect : `${zero}`, nfailed : `${zero}`, overall : `${zero}` },
  json: true };

r(o, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

Figura 10 : Método POST

Finalmente obtenemos el enunciado y las posibles respuestas del número de fila que corresponda con el número random anteriormente obtenido y se los presentamos al usuario.

Una vez ejecutada la función se le mostrará al usuario un mensaje correspondiente a la tercera caja oscura donde le indica que elija un número para escoger su respuesta.

Figura 11 : Nodo salida texto

Después el programa esperará a que el usuario introduzca un valor que será tratado como una posible respuesta.

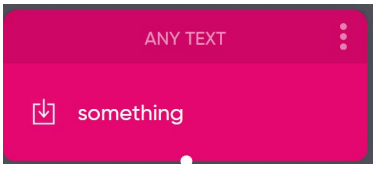


Figura 12 : Nodo de obtención de variable

Finalmente entrara la función verify randomquestion 2.0

verify randomquestion 2.0:

Inicialmente se obtienen todas las filas de la colección "Userandnumber" que contiene todos los usuarios y números correspondientes que han sido registrados.

Se realiza un bucle que recorre fila a fila comparando el nombre del usuario que está ejecutando esta función con los usuarios que ha devuelto la colección "Userandnumber", cuando coincida, obtendremos el número que corresponde a la fila de las preguntas de la colección de cuestionarios.

Seguidamente, guardaremos en nodos JSON todos los cuestionarios de la colección de cuestionarios.

Habiendo obtenido el número correspondiente a la fila de la cual se está haciendo el cuestionario, sacamos el valor

de cada una de las posibles respuestas pudiendo ser o bien -25 para respuestas incorrectas o 100 para la correcta.

Asignamos una nota para cada una de las posibles opciones y devolvemos la nota al usuario según la respuesta que haya escogido.

Respuesta incorrecta:

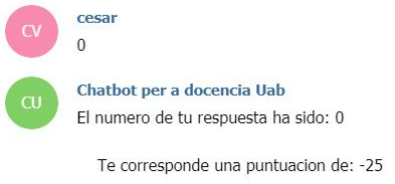


Figura 12 : Traza test incorrecto

Respuesta correcta:

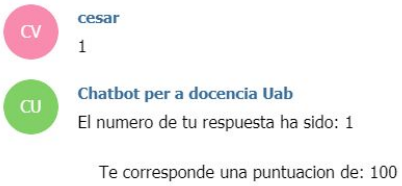


Figura 13 : Traza test correcto

En caso de que no se haya introducido un valor correcto se le indica al usuario.

6.4 Flujos de información para el usuario

El objetivo de esta funcionalidad es aportar información al usuario sobre la asignatura, mostrándole las siguientes opciones:

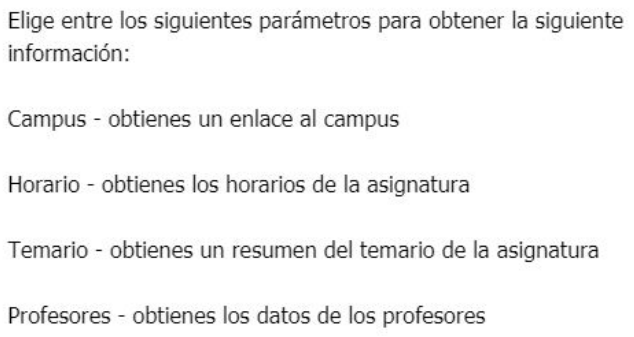


Figura 14 : Flujo info

Para comprobar que se introduce un valor entre los existentes se comprueba mediante una condición "if()" donde comprueba que el valor introducido sea uno de los disponibles

```
if (answer == "Campus" ||  answer == "Horario"|| answer == "Temario" || answer ==
  "Profesores") {
```

Figura 15 : Comprobación respuesta

Implementación opciones: Campus, Horario, Temario:

Se ha guardado en una única colección los valores pertinentes según el dato que se esté solicitando.

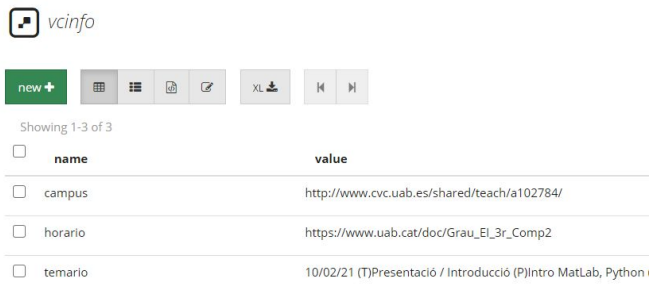


Figura 16 : Colección de información

El proceso de obtención de datos se hace utilizando la variable que ha enviado el usuario ya que coincide con el valor de la columna "name", convirtiendo esta variable siempre a minúsculas para evitar un error.

Se procede a realizar el metodo GET a la base de datos

```
  var options = { method: 'GET',
url: `https://chatbot-efcd.restdb.io/rest/vcinfo?q={"name": "${answer}"}`,
headers:
 { 'cache-control': 'no-cache',
   'x-apikey': '51a9ce7dc1f4f5397804be4214eaff58f34d6' } };

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

Figura 17 : Método GET

Y se guarda el contenido en una variable donde extraemos el contenido de la columna "value" para mostrarlo al usuario.

Campus: devuelve un enlace al campus de la asignatura de visión por computador

Horario: devuelve un enlace hacia una imagen con los horarios de la mención.

Temario: devuelve una lista de temarios con sus correspondientes modalidades de teoría, laboratorios y prácticas.

Profesores: devuelve, por cada fila que exista en la colección de la base de datos, un listado de cada uno de los profesores con sus cargos, correos y disponibilidad para tutorías.

En esta colección se guarda un profesor por cada línea.

Figura 18: Selección de profesores

Se obtiene el objeto resultante del método GET

Se realiza un tratamiento de datos en el que se guarda, por cada posición de un array, el número de profesor con cada uno de sus atributos (cargo, tutorías, mail, etc).

```
let profesores = new Array(result.length);
for (var i = 0; i < result.length ; i++) {
  delete result[i]._id;

  num = `Professor ${i + 1}`

  nombre = Object.values(result[i].nom)
  nombre = nombre.join("")
  nombre = `Nombre : ${nombre}`

  carrec = Object.values(result[i].carrec)
  carrec = carrec.join("")
  carrec = `Carrec : ${carrec}`

  mail = Object.values(result[i].mail)
  mail = mail.join("")
  mail = `Mail : ${mail}`

  tutories = Object.values(result[i].tutories)
  tutories = tutories.join("")
  tutories = `Tutories : ${tutories}`

  profesores[i] = `${num} \n\n ${nombre}  \n ${carrec}  \n ${mail}  \n ${tutories
    \n \n `
  //profesores[i] = `${Object.values(result[i])}  \n `

}
```

Figura 19: Tratamiento datos profesores

6.5 Flujos de datos para el profesorado

Esta funcionalidad sirve para que el profesor pueda tener un reporte sobre el desempeño de cada uno de los usuarios que han realizado tests. Esta funcionalidad requiere una contraseña para comprobar que es un usuario con privilegios (un profesor), la contraseña es "1234".

El procedimiento para la elaboración de esta funcionalidad es la siguiente:

La colección guarda un alumno con los atributos de interés para el profesorado por cada fila.

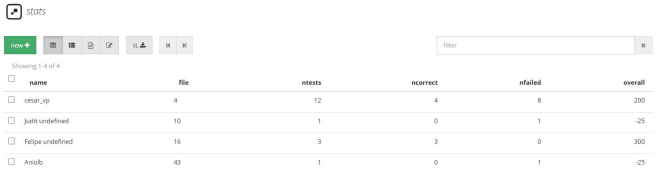


Figura 20: Colección de alumnos

Para la obtención de estos datos se efectúa un método GET

Se realiza un tratamiento de datos para que el usuario pueda ver de forma agradable el contenido de cada una de las filas de la colección.

```
for (var i = 0; i < result.length ; i++) {
  delete result[i]._id;

  num = `Alumno ${i + 1}`

  nombre = Object.values(result[i].name)
  nombre = nombre.join("")
  nombre = `Nombre : ${nombre}`

  ultima = Object.values(result[i].file)
  ultima = ultima.join("")
  ultima = `Ultima Pregunta : ${ultima}`

  fallados = result[i].nfailed
  fallados = `Test fallados : ${fallados}`

  acertados = result[i].ncorrect
  acertados = `Test acertados : ${acertados}`

  totales = result[i].ntests
  totales = `Test totales : ${totales}`

  global = result[i].overall
  global = `Valoracion global  : ${global}`


  datos[i] =`${num} \n\n  ${nombre} \n ${ultima} \n ${fallados} \n ${acertados} \n
    ${totales} \n ${global} \n\n `


}
```

Figura 21: Tratamiento datos alumnos

7 RESULTADOS

A continuación se muestran las salidas que el chatbot devuelve dependiendo de las entradas del usuario.

7.1 Pantalla inicial

La pantalla inicial se activa en cuanto saludamos al bot. Éste nos da la bienvenida y nos muestra las tres principales opciones a elegir: test, info y datos.

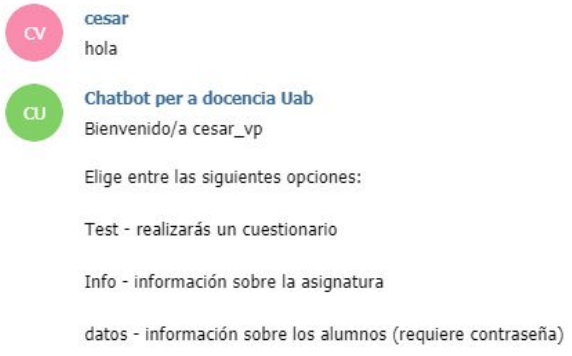


Figura 22: Pantalla de bienvenida

7.2 Pantalla Test

Se activa una vez introducimos el comando "Test" en la pantalla de bienvenida. Este apartado nos muestra el enunciado de una pregunta y cuatro posibles respuestas, dejándonos elegir la respuesta que veamos más adecuada.

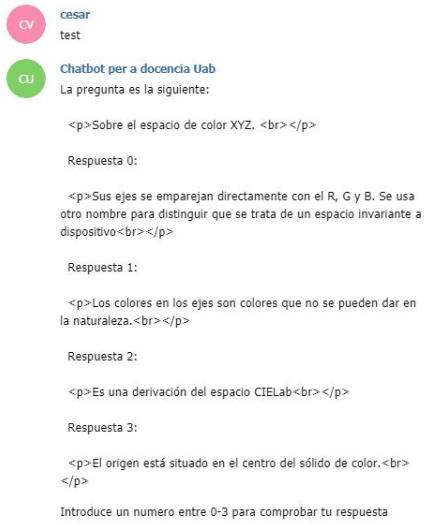


Figura 23: Comando test

7.2.1 Respuesta correcta

Se enviará un texto al usuario indicando que ha obtenido una puntuación de 100 para esta pregunta y se incrementará el número de respuestas acertadas en su registro.

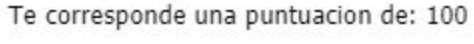


Figura 24: Respuesta correcta

7.2.2 Respuesta incorrecta

Se enviará un texto al usuario indicando que ha obtenido una puntuación de -25 para esta pregunta y se incrementará el número de respuestas erróneas en su registro.

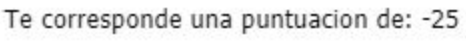


Figura 25: Respuesta incorrecta

7.2.3 Iteración

Una vez completado un test, el chabot preguntará al usuario si quiere volver a realizar otro test, si el usuario escribe "Si" se iniciara otro test de forma aleatoria, si marca que "No" habrá terminado la funcionalidad de test.

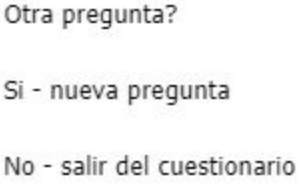


Figura 26: Reinicio de test

7.3 Pantalla Info

Se activa una vez introducimos el comando "Info" en la pantalla de bienvenida. Esta sección muestra una nueva lista de opciones que aportan información sobre la asignatura. Las opciones son : campus, horario, temario, profesores.

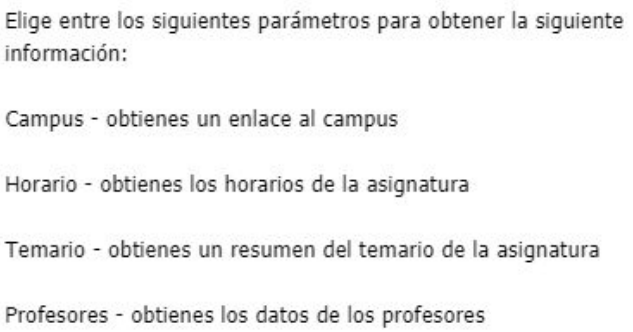


Figura 27: Comando info

7.3.1 Campus y Horario

Al escribir la opción "Campus" o "Horario" el chatbot muestra un mensaje que contiene un enlace que redirige al contenido deseado.

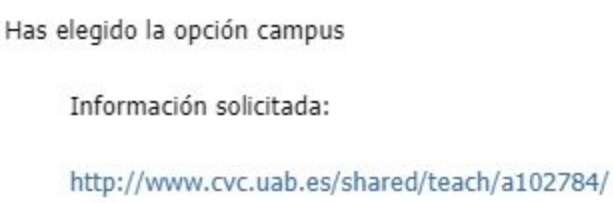


Figura 28: Acceso campus

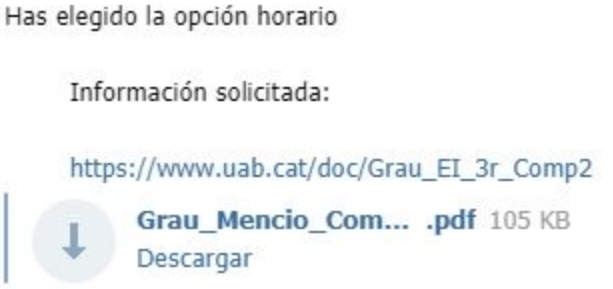


Figura 29: Acceso horario

7.3.2 Temario

Introduciendo la opción "Temario" obtendremos un texto que contiene el temario relacionado con las clases de teoría, prácticas y de laboratorio.

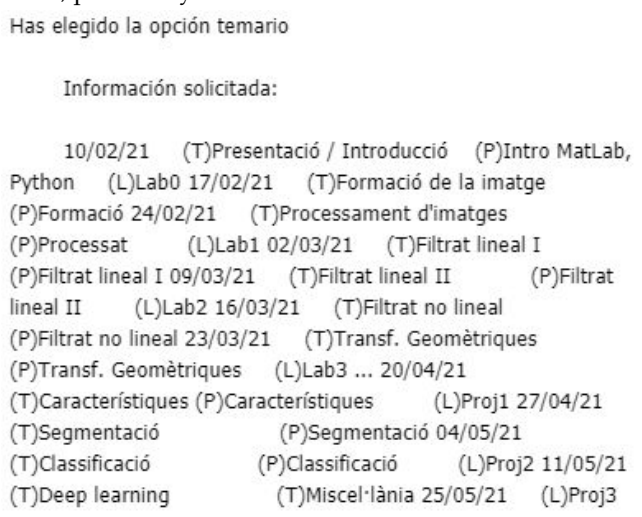


Figura 30: Temario

7.3.3 Profesores

Con la opción "Profesores" se mostrará la información relevante de cada uno de los profesores encargados de la asignatura.

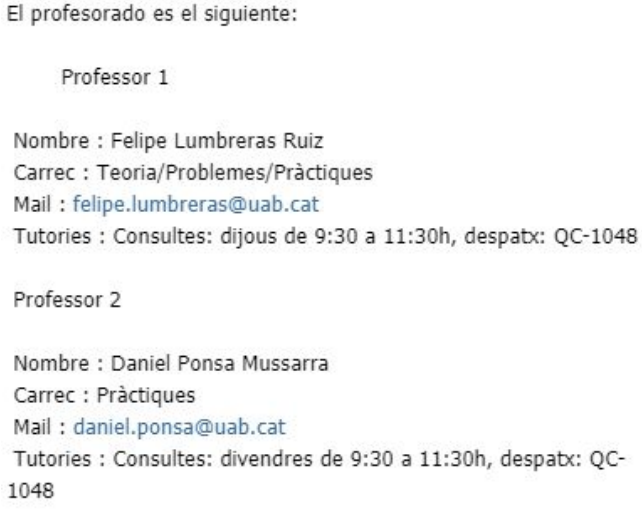


Figura 31: Profesores

7.4 Pantalla datos

Mediante la introducción de la opción "datos" en la pantalla de bienvenida, se mostrará al usuario que posea la contraseña (1234) una lista con los usuarios que han realizado test y datos relevantes para poder valorar su desempeño.

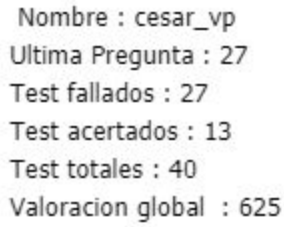


Figura 32: Datos

7.5 Resumen resultados

Los resultados cumplen con los requisitos especificados al inicio del proyecto, aportando al usuario información de interés sobre la asignatura. La información se muestra de una forma intuitiva para que el usuario pueda avanzar entre las diversas opciones que el proyecto ofrece y los resultados de las peticiones que hace el usuario sean fácilmente entendibles. Como parte negativa, no se ha hecho un tratamiento de datos en cuanto a los enunciados y las respuestas del apartado de tests para eliminar los tags, ya que vienen en formato html, lo cual es menos entendible visualmente.

8 CONCLUSIONES

Se ha conseguido cumplir con los requisitos establecidos como principales. Tanto la funcionalidad de aportar información al usuario, como la de realización de cuestionarios funcionan de forma correcta y son visualmente intuitivas para el usuario. Además, se ha cumplido con objetivos secundarios tales como mostrar información sobre el alumnado al profesorado para que estos puedan tener un mayor feedback del desempeño de sus alumnos. Ha quedado pendiente realizar un sistema de gamificación que fue propuesto en la entrevista con los profesores, pero dada la naturaleza de esta aplicación, considero que es suficientemente gratificante el hecho de poder preparar de forma cómoda el temario mediante la realización de test con el dispositivo móvil.

Este proyecto tiene potencial para ser explotado en el ámbito del Big-Data, ya que con la suficiente cantidad de muestras, se pueden hacer estudios en base a una cantidad de datos obtenidos. Como por ejemplo: donde fallan más los alumnos, la media de a partir de cuantos test un usuario empieza a conocer el temario, etc.

En cuanto a conocimientos adquiridos, he mejorado considerablemente en cuanto a la interacción entre una aplicación y una base de datos, ya que se han tocado todos los métodos existentes: GET, POST, PUT, DELETE. Y también se ha mejorado en cuanto a conocimientos de Javascript. 11 mar 2019

BIBLIOGRAFIA

[1] Hostinger tutoriales, agosto 13, 2020 https://www.hostinger.es/tutoriales/conectar-php-mysql/
[2] YouTube, 11 mar 2019 https://www.youtube.com/watch?v=Cw4TgJIHaXI
[3] DB4free , 2020 https://www.db4free.net/signup.php
[4] Flow.ai,2019 https://flow.ai/blog/how-to-build-a-chatbot-quiz-for-WhatsApp
[5] Flow.ai,2019 , 2019https://flow.ai/blog/kb-how-to-integrate-your-chatbot-with-a-database
[6] Flow.ai,2019 https://flow.ai/docs/tutorial#other-tutorials
[7] Restdb.io, 2015 https://restdb.io/docs/quick-start

APÉNDICE

A1. Exploratory Testing

En este apartado se realizan ataques para forzar la aparición de errores.

A1.1 Valor inválido

Ataque 1: Introducir un valor inválido
En la sección de información nos pide introducir una serie de valores:
Valores válidos: Campus, Horario, Temario, Profesores
Valor introducido: 9
Respuesta:

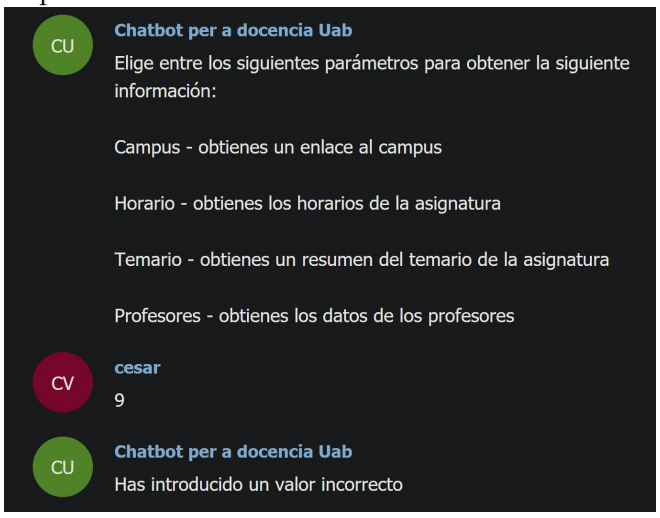


Figura 33: Valor incorrecto

A1.2 Llenar buffer

Ataque 2: Llenar el buffer
Introducir una gran cantidad de valores con el objetivo de que no se pueda procesar la información
Valor introducido: 9999…
Respuesta:

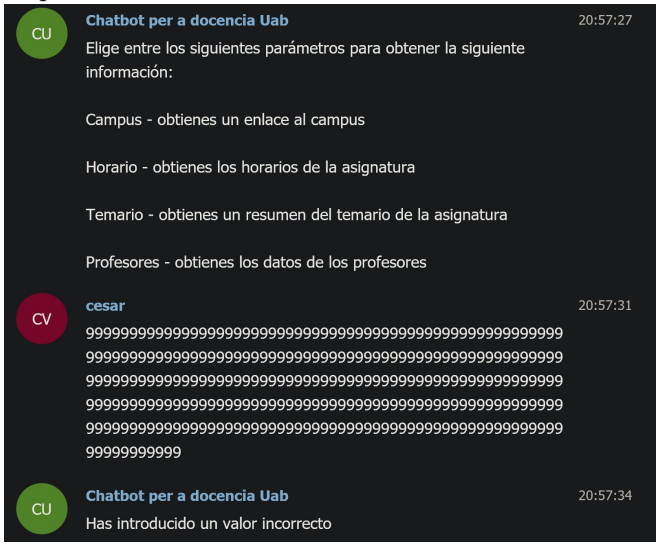


Figura 33: Exceder buffer

A1.3 Valores aritméticos

Ataque 3: Introducir valores de forma aritmetica
En la sección de test, introducir un valor válido envuelto por paréntesis
Valores válidos: 0, 1, 2, 3
Valor introducido: (1)
Respuesta:

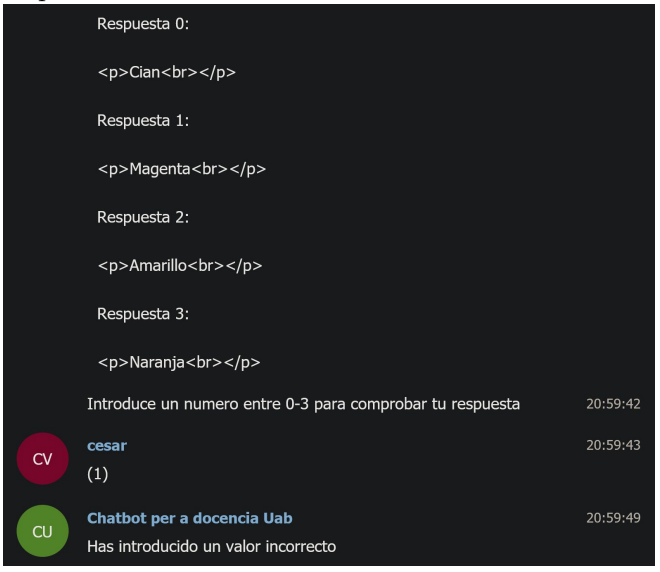


Figura 33: Valores aritméticos