# Juego 20 preguntas en Google Assistant

EDyA 361075 Sánchez Torres, Carlos Eduardo

15 de julio de 2021

## 1. Resumen

Implementación del juego 20 preguntas en Google Assitant, Telegram y WhatsApp mediante DialogFlow, el backend sera Go y Firestore. En el cual el objetivo es preguntar «sí» y «no» en español sobre personas famosas.

## 2. Justificación

Los juegos forman parte de las necesidades según Maslow [3], por dar descanso. El juego utiliza principalmente árboles, base de datos para personalizar los resultados y una búsqueda de grafos para encontrar datos no previos, cumpliendo con el objetivo de la asignatura. Además, el autor sabe como implementar aplicaciones en dichas plataformas como puede ver en la página web [6], chatbot en WhatsApp [7] y en GitHub [8].

## 3. Estado del arte

Akinator, el juego popular y máximo exponente implementado en Android, Alexa y la mayoría de las plataformas [1], entra en la categoría del juego 20 preguntas. Otras investigaciones usan aprendizaje por refuerzo para mejorar su desempeño [4] y sistemas de recomendación [5], logrando reducir la cantidad de preguntas hasta 10. Siendo la principal diferencia que el proyecto funcionara correctamente en Google Assistant, tema que Akinator no resuelve del todo.

## 4. Objetivo General

Desplegar el juego de 20 preguntas a las plataformas más populares de chat, con alta escalabilidad ante las lecturas.

## 5. Objetivos específicos

Crear una prueba de concepto en un ambiente local mediante consola y lectura de archivos para entender el algoritmo central. Después, conectar el software a una base de datos -Firestore- para escalar el número de preguntas, servir la aplicación a una plataforma de lenguaje natural -Dialogflow- mediante HTTP usando Cloud Functions y finalmente, mediante los conectores que ofrezca el proveedor se despliega a Google Assitant, Telegram y WhatsApp, cuando pasemos el proceso de verificación y los beta testers.

## 6. Hipótesis

Se esperan mas lecturas que escrituras, debido a que es un juego con poco guardado de información por parte del usuario, 3 beta testers: los dos estudiantes participantes y la profesora. El proceso de verificación dura 3 semanas máximo y puede ser en paralelo. El proceso de migrar de local a la nube debería ser trivial. Se espera no gastar dinero. Esperamos el registro de 100 usuarios después de servir la aplicación, por compartir en redes sociales. La aplicación debe cumplir el triangulo de los proyectos: tiempo, dinero y alcance para lograr calidad.

# 7. Metodología

Como metodología ágil usaremos Scrum+XP, para el análisis y diseño usaremos UML, para la asignación de tareas se usara Trello, documentación en Notion, mediante TDD implementaremos el resto del desarrollo. La validación con beta testers será el primero de mayo de 2021. El sistema operativo en local sera Ubuntu 20.4. El lenguaje de programación, Go. Despliegue sobre Google Cloud Functions para servir HTTP. Dialogflow como plataforma de lenguaje natural. Firestore como base de datos.

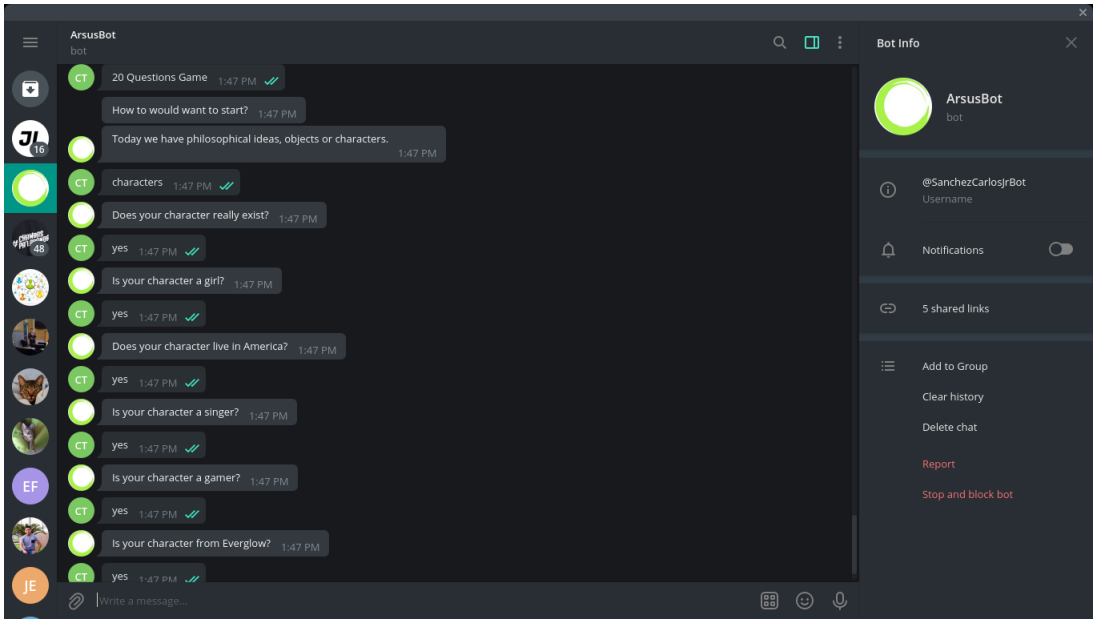


Figura 1: Trabajo en equipo.

# 8. Implementación

```go
type Discriminator string

const (
  QUESTION Discriminator = "Q"
  ANSWER   Discriminator = "A"
)

type DatabaseRepository interface {
  First()
  Previous()
  Next(response UserResponse)
  Actual() (string, Discriminator)
}
```

Figura 2: Interfaz para volver mantenible el código si cambiamos de base de dato.

```go
func (receiver *Question) Reply() {
    userResponse := infraestructure.StreamRepository().Interact(receiver.Response)
    for userResponse == domain.PREVIOUS && receiver.ancestor == nil {
        userResponse = infraestructure.StreamRepository().Interact(receiver.Response)
    }
    if domain.PREVIOUS == userResponse {
        infraestructure.DatabaseRepository().Previous()
        receiver.ancestor.Reply()
        return
    }
    infraestructure.DatabaseRepository().Next(userResponse)
    receiver.Children[userResponse] = factoryResponseType(receiver)
    child := receiver.Children[userResponse]
    child.Reply()
}
```

Figura 3: Replicar pregunta.

```go
    func factoryResponseType(ancestor domain.Game) domain.Game {
        response, discriminator := infraestructure.DatabaseRepository().Actual()
        switch discriminator {
        case domain.QUESTION:
            question := newQuestion(ancestor, response)
            return question
        case domain.ANSWER:
            return &Answer{Response: response}
        }
        return nil
    }
```

Figura 4: Crear cuestionario.

```go
func (receiver *Firestore) First() {
  snapshot, _ := receiver.client.Collection(receiver.collection).Doc(receiver.idFistDocument).Get(receiver.context)
  receiver.documentData = snapshot.Data()
}

func (receiver *Firestore) Previous() {
  receiver.documentData = receiver.previousData
}

func (receiver *Firestore) Next(response domain.UserResponse) {
  receiver.previousData = receiver.documentData
  newDocument := receiver.documentData[strconv.Itoa(int(response))].(string)
  snapshot, _ := receiver.client.Collection(receiver.collection).Doc(newDocument).Get(receiver.context)
  receiver.documentData = snapshot.Data()
}

func (receiver *Firestore) Actual() (string, domain.Discriminator) {
  return receiver.documentData["response"].(string), domain.Discriminator(receiver.documentData["discriminator"].(string))
}
```

Figura 5: Carga desde Firestore a DialogFlow.

```
  memory: '2GB',
})
.https.onRequest(async (request: https.Request, response: Response) => {
  const body: BodyTwilio = request.body;
  const telecommunicationCreator = new TelecommunicationCreator(new Dialogflow());
  const xml: string = await telecommunicationCreator.create(body.Body, body.From);
  response.setHeader('Content-Type', 'text/xml');
  response.send(xml);
  const firestoreCommunicationStoreRepository = new FirestoreCommunicationStoreRepository();
  return firestoreCommunicationStoreRepository.save({
    from: body.From,
    to: body.To,
    body: body.Body,
    extra: {
      ...body,
    },
  });
});
```

Figura 6: Responde a DialogFlow.

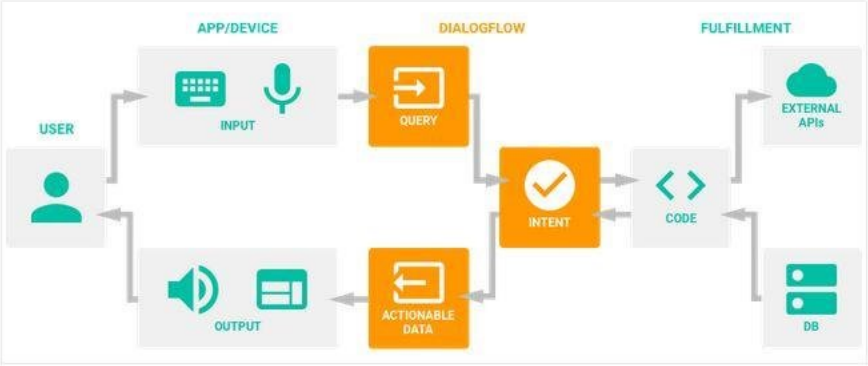


Figura 7: Arquitectura. Cargamos desde Firestore a Dialogflow, mediante un árbol.

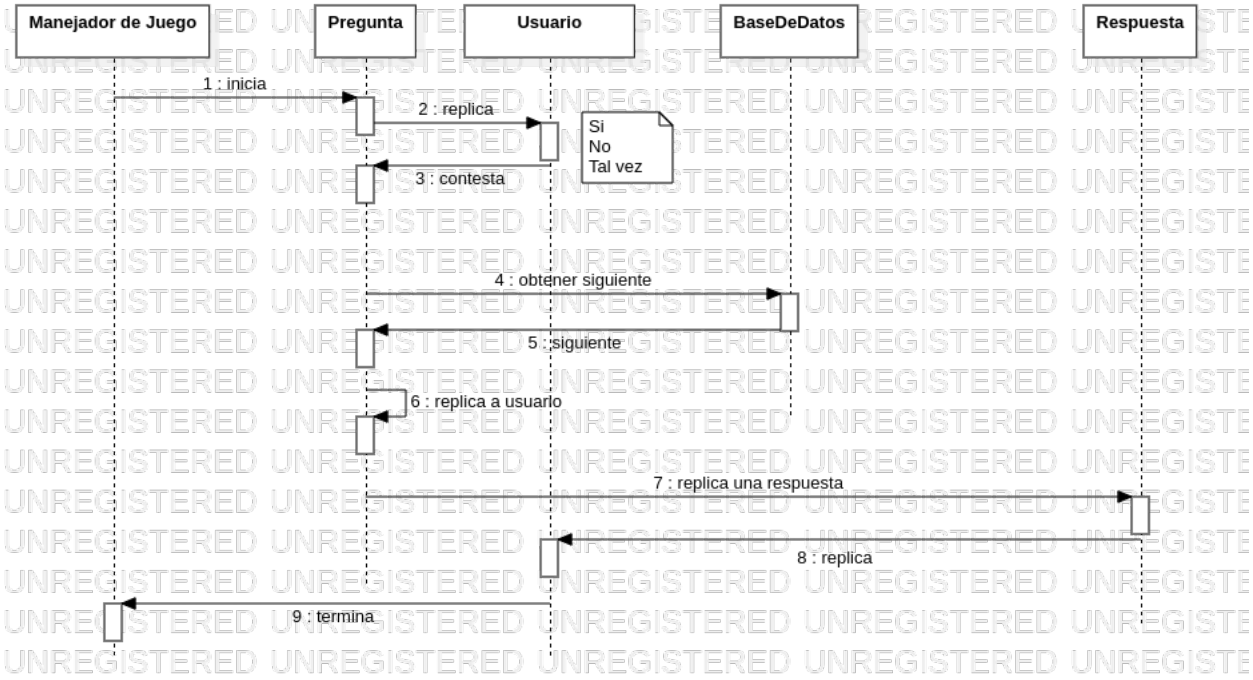


Figura 8: Secuencia del juego.

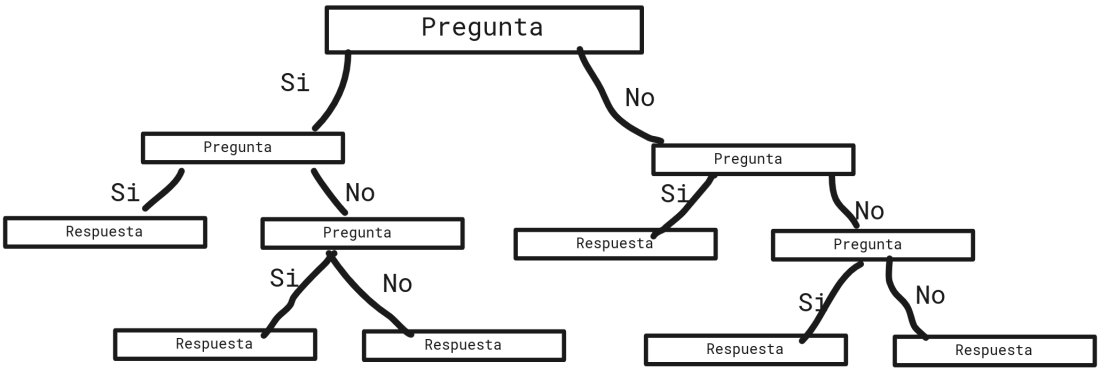


Figura 9: Árbol generado de manera perezosa.

# 9. Pruebas

Cuadro 1: Caso de prueba 1.

| Caso de prueba | Probar camino sí-sí-sí |
|---|---|
| Precondiciones | Base de datos preparada |
| Proceso | Resultado |
| Se inicia el software | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 1. |
| El usuario contesta sí | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 3. |
| El usuario contesta sí | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 5. |
| El usuario contesta no | Da indicaciones de como se juega, responde de acuerdo al archivo linea 9 y solicita retroalimentación para continuar. |
| El usuario termina. | El software se cierra. |

Cuadro 2: Caso de prueba 2.

| Caso de prueba | Probar camino sí-sí-no |
|---|---|
| Precondiciones | Base de datos preparada |
| Proceso | Resultado |
| Se inicia el software | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 1. |
| El usuario contesta no | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 4. |
| El usuario contesta no | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 6. |
| El usuario contesta no | Da indicaciones de como se juega, responde de acuerdo al archivo linea 8 y solicita retroalimentación para continuar. |
| El usuario termina. | El software se cierra. |

Cuadro 3: Caso de prueba 3.

| Caso de prueba | Probar camino no-no-no |
|---|---|
| Precondiciones | Base de datos preparada. |
| Proceso | Resultado |
| Se inicia el software | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 2. |
| El usuario contesta sí | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 6. |
| El usuario contesta sí | Da indicaciones de como se juega y pregunta de acuerdo al archivo linea 9. |
| El usuario contesta sí | Da indicaciones de como se juega, responde de acuerdo al archivo linea 15 y solicita retroalimentación para continuar. |
| El usuario termina. | El software se cierra. |

## 10. Resultados

Los distintas plataformas (la versión de consola con su respectivo código en español): https://sanchezcarlosjr.com/20-Questions-Game/5mm93vndqgBM4oP2MzbG


Figura 10: Consola. En esta versión se uso un archivo: https://firebasestorage.googleapis.com/v0/b/arsus-production.appspot.com/o/assets %2F20questionsGame$_{e}s_{M}X.txt?alt = mediatoken = e6652399 - d12a - 4d28 - a956 - dc4ad61f3fc4$

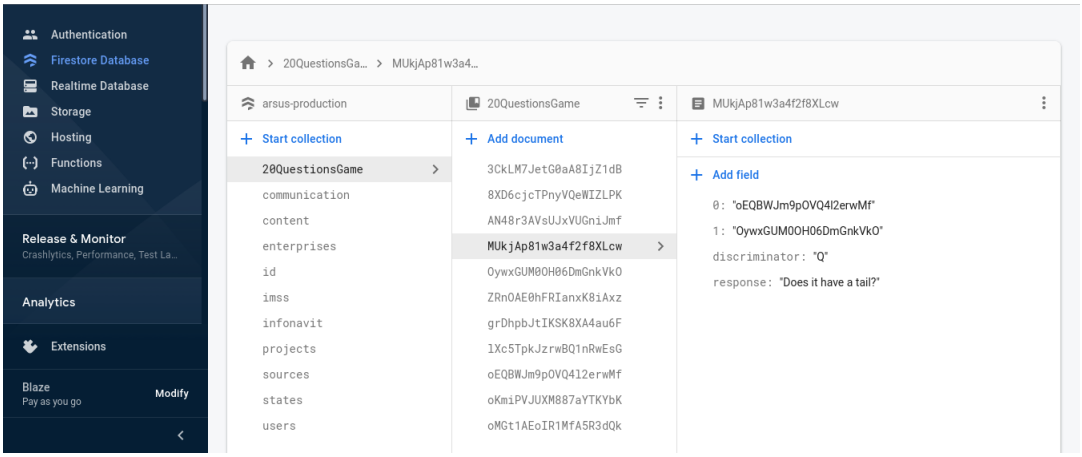


Figura 11: Firestore. Base de dato NO SQL. La cual se conecta a través de ID.

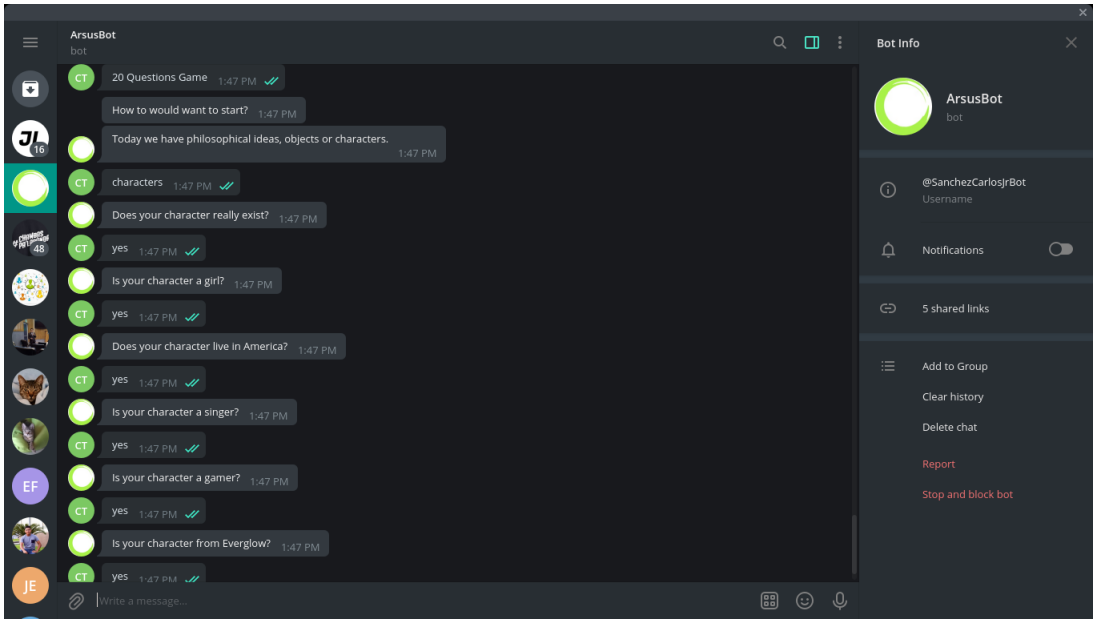


Figura 12: Telegram. En esta versión se uso Firestore y no un archivo. Y luego se lleva a Dialogflow.

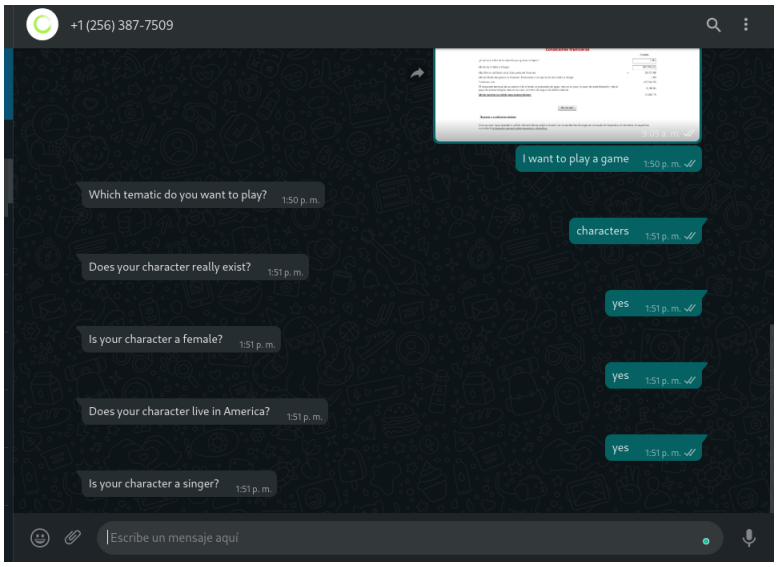


Figura 13: Whatsapp. En esta versión se uso Firestore y no un archivo. Y luego se lleva a Dialogflow.

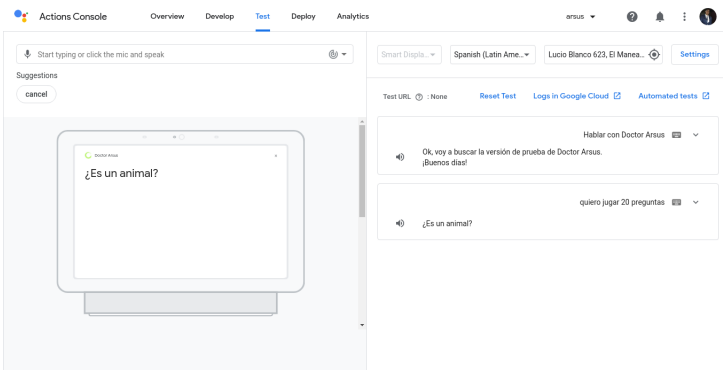


Figura 14: Google Assistant. En esta versión se uso Firestore y no un archivo. Y luego se lleva a Dialogflow.

# 11. Glosario

20 preguntas. Juego popular que mediante preguntas cuyas respuestas son «sí»,«no» y «tal vez», se intenta adivina lo que esta pensando.

Scrum+XP. Scrum como marco de trabajo para productos y XP como una implementación de las buenas practicas en el desarrollo de software.

Plataforma de lenguaje natural. Plataforma usada para diseñar e integrar conversaciones naturales con inteligencia artificial en móviles, web y otros dispositivos.

Beta testers. Primeros usuarios sobre versiones inestables.

TDD. Desarrollo guiado por pruebas.

Google Cloud Functions. Funciones en la nube sin la necesidad de admnistrar un servidor.

# Referencias

[1] Walsorth, Mansfield Tracy. Twenty questions: a short treatise on the game, Holt, 1882.

[2] Elokence. 2021. Akinator.com. Retrieved February 16, 2021 from https://en.akinator.com/

[3] Maslow, A.H. (1943). .ᴬ theory of human motivation". Psychological Review. 50 (4): 370–96. CiteSeerX 10.1.1.334.7586. doi:10.1037/h0054346 – via psychclassics.yorku.ca.

[4] Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, & Zhan Chen. (2019). Playing 20 Question Game with Policy-Based Reinforcement Learning.

[5] Alvin Dey and Harsh Kumar Jain and Vikash Kumar Pandey and Tanmoy Chakraborty (2019). All It Takes is 20 Questions!: A Knowledge Graph Based ApproachCoRR, abs/1911.05161.

[6] Arsus. 2020. Arsus - By Carlos Sanchez. Arsus. Retrieved February 16, 2021 from https://sanchezcarlosjr.com/

[7] 2021. Share on WhatsApp. WhatsApp.com. Retrieved February 16, 2021 from https://api.whatsapp.com/send?phone=+1 %20(256) %20387-7509/

[8] sanchezcarlosjr. 2021. sanchezcarlosjr/arsus. GitHub. Retrieved February 16, 2021 from https://github.com/sanchezcarlosjr/arsus