



# Especificación de Requerimientos de Software

**Proyecto:** Adivina Adivinador

Juego por Terminal en Python

Revisión 1.2

Julio de 2023.

## Ficha del Documento

Fecha	Revisión	Autor	Verificación
15 - 07 - 2023	1.0	Jorge Mera.	Se crea el Anteproyecto
16 - 07 - 2023	1.1	Jorge Mera.	Se sube a repositorio
17 - 07 - 2023	1.2	Jorge Mera	Se actualiza la versión e incluye documentación.

### Imagen 1

Enlace QR al Repositorio del Proyecto en *Github*.



# Contenido

<b>Ficha del Documento .....</b>	<b>2</b>
Fecha .....	2
Revisión .....	2
Autor .....	2
Verificación.....	2
 <b>Contenido .....</b>	 <b>3</b>
 <b>1. Introducción .....</b>	 <b>5</b>
1.1 Propósito del documento.....	5
1.2 Alcance del sistema.....	5
1.3 Definiciones, Acrónimos y Abreviaturas .....	5
1.4 Referencias .....	5
1.5 Visión General del Documento .....	6
<b>2. Descripción General.....</b>	<b>7</b>
2.1 Perspectiva del Producto .....	7
2.2 Funciones del Producto .....	7
2.3 Características de los Usuarios.....	7
2.4 Restricciones Generales .....	7
2.5 Suposiciones y Dependencias .....	7
2.6 División del Sistema en Componentes.....	7
<b>3. Requisitos Específicos .....</b>	<b>8</b>
3.1 Requisitos Funcionales.....	9
3.2 Requisitos No Funcionales.....	9
<b>4. Otros Requisitos .....</b>	<b>10</b>
4.1 Requisitos de Implementación .....	10
4.2 Requisitos de Documentación.....	10
4.3 Requisitos de Pruebas .....	10
4.4 Requisitos de Mantenimiento.....	10
<b>5. Criterios de aceptación .....</b>	<b>11</b>
5.1. C1: Animación de inicio del juego.....	11
5.2. C2: Adivinar letra.....	11
5.3. C3: Adivinar palabra .....	11
5.4. C4: Mostrar palabra actualizada.....	11

5.5. C5: Finalizar el juego .....	11
<b>6. Diagramas.....</b>	<b>12</b>
6.1. Diagrama de casos de uso .....	12
<b>7. Conclusiones .....</b>	<b>14</b>
<b>Apéndice: Detalles de Funciones, Clases, Métodos y Ejecución de la aplicación en Terminal.</b>	<b>15</b>
1. Funciones: .....	15
2. Clases: .....	16
3. Métodos:.....	18
4. Ejecución del Programa: .....	19
5. Enlace a repositorio de Proyecto “Adivina Adivinador” .....	22

# 1. Introducción

El juego "Adivina Adivinador" es un juego de adivinanzas en el que los jugadores deben adivinar un color oculto en un número limitado de intentos. El juego se desarrolla en la terminal y proporciona una experiencia interactiva mediante el uso de caracteres ASCII y emoticones.

## 1.1 Propósito del documento

Este documento describe los requerimientos funcionales y no funcionales del juego "Adivina Adivinador" y define las características y funcionalidades esenciales para su implementación. También proporciona información sobre los criterios de rendimiento y usabilidad.

## 1.2 Alcance del sistema

El programa "Adivina Adivinador" es un juego de adivinanzas en el que los jugadores intentan adivinar un color oculto con un número limitado de intentos. El juego proporciona pistas y retroalimentación interactiva para guiar al jugador hacia la respuesta correcta.

## 1.3 Definiciones, Acrónimos y Abreviaturas

Color Oculto: El color que el jugador debe adivinar.

Intentos Restantes: La cantidad de intentos que le quedan al jugador para adivinar el color oculto.

Letras Adivinadas: La lista de letras ingresadas por el jugador como parte de su intento para adivinar el color oculto.

## 1.4 Referencias

[1] «The Python Standard Library», *Python documentation*. <https://docs.python.org/3/library/index.html>

[2] Manik, «¿Cómo funciona la codificación de caracteres? - ASCII / Unicode», *YouTube*. 6 de diciembre de 2020. [En línea]. Disponible en: [https://www.youtube.com/watch?v=M\\_yNoV3c8DY](https://www.youtube.com/watch?v=M_yNoV3c8DY)

[3] Envntor, «Como instalar PIP de Python en Windows 10/11 (ACTUALIZADO 2022/2023)», *YouTube*. 1 de diciembre de 2022. [En línea]. Disponible en: [https://www.youtube.com/watch?v=tQo\\_W7EboWw](https://www.youtube.com/watch?v=tQo_W7EboWw)

[4] Alberto Papirrin, «Como actualizar todos los paquetes con pip [Python 3]», *YouTube*. 1 de abril de 2017. [En línea]. Disponible en: <https://www.youtube.com/watch?v=rgkYufNYRFE>





[5] Tecno Tutoriales, «Cómo Actualizar PIP En Python (2023) // Actualización De Pip En Python Facil y Rapido», *YouTube*. 17 de noviembre de 2022. [En línea]. Disponible en: <https://www.youtube.com/watch?v=BDGXBWtRdx8>

[6] ALGORITMODETAREA, «Algoritmo del juego del ahorcado en pseint», *YouTube*. 15 de marzo de 2021. [En línea]. Disponible en: <https://www.youtube.com/watch?v=Q4UqwFEN2TA>

[7] Profe Dago, «DFD ahorcado 01», *YouTube*. 15 de octubre de 2016. [En línea]. Disponible en: <https://www.youtube.com/watch?v=IJgW2I3SwSk>

[8] Programador MX, «Como hacer barra de progreso en Python Terminal | python tutorial», *YouTube*. 26 de octubre de 2022. [En línea]. Disponible en: <https://www.youtube.com/watch?v=-Ut3jIX7dp0>

[9] El Pingüino Tech, «CURSO DE PYTHON - Cómo Crear y Definir FUNCIONES en Python 🐧», *YouTube*. 27 de abril de 2022. [En línea]. Disponible en: <https://www.youtube.com/watch?v=7KPTQIwcWpk>

- [10] Tecno Tutoriales, «Como Hacer Una Funcion En Python (2023) // Crear Funciones en Python», *YouTube*. 18 de noviembre de 2022. [En línea]. Disponible en: <https://www.youtube.com/watch?v=1U7zmcrrj-QM>
- [11] Programación Fácil, «El MÉTODO \_\_init\_\_ y SELF - Curso desde cero Programación Orientada a Objetos Python - Capítulo 3», *YouTube*. 25 de mayo de 2020. [En línea]. Disponible en: <https://www.youtube.com/watch?v=ID518qg86qg>
- [12] Fundación para la Equidad Educativa, «2.4 Saltos de linea - Aprende Python con Miniproyectos - Miniproyecto 2», *YouTube*. 9 de agosto de 2021. [En línea]. Disponible en: <https://www.youtube.com/watch?v=32cTrlnDqOI>
- [13] Cristian Henao, « Como crear un Repositorio y Subir Proyecto a  GITHUB  Paso a Paso », *YouTube*. 24 de marzo de 2021. [En línea]. Disponible en: <https://www.youtube.com/watch?v=eQMciGVc8N0>
- [14] nicosiored, «Diagrama de Casos de Uso I - 4 - Tutorial UML en español», *YouTube*. 16 de noviembre de 2017. [En línea]. Disponible en: [https://www.youtube.com/watch?v=yZWVx\\_esIq8](https://www.youtube.com/watch?v=yZWVx_esIq8)
- [15] El Potro Tecnologico, «NORMA IEEE830 - Especificacion de Requerimientos de Software», *YouTube*. 10 de febrero de 2023. [En línea]. Disponible en: <https://www.youtube.com/watch?v=ZUVDkSXrt5I>
- [16] DAYNER FELIPE ORDOÑEZ LOPEZ, «IEE830 Generalidades», *YouTube*. 3 de junio de 2020. [En línea]. Disponible en: <https://www.youtube.com/watch?v=FJIAQTiGjXI>
- [17] Facultad de Estudios a Distancia UMNG, «Elementos de la norma IEEE 830», *YouTube*. 24 de agosto de 2021. [En línea]. Disponible en: [https://www.youtube.com/watch?v=LjBOTZdd\\_iE](https://www.youtube.com/watch?v=LjBOTZdd_iE)
- [18] CamaradaYorch, «GitHub - CamaradaYorch/Adivina\_Adivinador: Juego de adivinanzas para adivinar un color oculto.», *GitHub*, julio de 2023. [https://github.com/CamaradaYorch/Adivina\\_Adivinador/tree/main](https://github.com/CamaradaYorch/Adivina_Adivinador/tree/main)

## 1.5 Visión General del Documento

Este documento se organiza en varias secciones que describen en detalle los requisitos del sistema. La sección 2 proporciona una descripción general del sistema y su contexto. La sección 3 enumera los requisitos funcionales y no funcionales del sistema. La sección 4 incluye otros requisitos relacionados con la implementación, documentación, pruebas y mantenimiento del sistema. Los apéndices contienen un glosario de términos y detalles adicionales sobre las funciones, métodos y clases del sistema.

## 2. Descripción General

El juego "Adivina Adivinador" permite a los jugadores intentar adivinar un color oculto seleccionado aleatoriamente dentro de una lista de colores predefinidos. Los jugadores pueden ingresar letras o palabras para adivinar el color, y el sistema proporciona retroalimentación en tiempo real sobre la validez de las entradas y si las letras o palabras adivinadas coinciden con el color oculto.

El sistema presenta una interfaz basada en texto en la terminal, sin requerir un diseño de interfaz gráfica compleja. Todos los comandos e interacciones con el juego se realizan a través de la línea de comandos.

### 2.1 Perspectiva del Producto

El sistema "*Adivina Adivinador*" se desarrollará como una aplicación independiente en el lenguaje de programación Python. Será ejecutado en un entorno de consola/terminal.

### 2.2 Funciones del Producto

El sistema permitirá a los jugadores:

- Ingresar su nombre para identificarse en el juego.
- Intentar adivinar el color oculto mediante el ingreso de letras o palabras.
- Recibir pistas y retroalimentación interactiva basada en las respuestas proporcionadas.
- Visualizar la representación actualizada del color oculto con las letras adivinadas correctamente.

### 2.3 Características de los Usuarios

El sistema está diseñado para ser utilizado por jugadores de todas las edades que deseen jugar un juego de adivinanzas interactivo por terminal.

### 2.4 Restricciones Generales

- El sistema se desarrolló utilizando el lenguaje de programación Python.
- El sistema funcionará en un entorno de consola/terminal.
- El sistema estará limitado a un número predefinido de intentos para adivinar el color oculto.

### 2.5 Suposiciones y Dependencias

- Se supone que los usuarios tienen conocimientos básicos de cómo utilizar una consola/terminal.
- El sistema depende de la generación aleatoria de un color oculto a partir de una lista predefinida de colores.

### 2.6 División del Sistema en Componentes

El sistema "*Adivina Adivinador*" estará compuesto por una clase principal "adivina\_adivinador\_v1.2" que encapsulará las funciones y métodos necesarios para el juego.



## Codigo Python de Adivina Adivinador

```

1 # Autor: [Nome aqui]
2 # Data: [Data aqui]
3 # Descrição: [Bom de programação para aprender a usar o Python]
4 # Versão: 1.0.0
5
6 #
7
8 # Importando bibliotecas
9 import random
10 import time
11
12 # Variáveis
13 nome = ""
14 idade = 0
15 altura = 0
16 peso = 0
17 sexo = ""
18 cor_pelo = ""
19 cor_olhos = ""
20 cor_cabelo = ""
21 cor_roupa = ""
22
23 # Função para gerar nome
24 def gerar_nome():
25     nomes = ["João", "Maria", "Pedro", "Ana", "Carlos", "Lucas", "Mariana", "Gabriel", "Isabella", "Rafael", "Amanda", "Felipe", "Miguel", "Larissa", "Vinicius", "Thiago", "Carla", "Rodrigo", "Juliana", "Ricardo", "Patrícia", "Diego", "Fernanda", "Rafaela", "Gabriel", "Mariana", "Felipe", "Miguel", "Larissa", "Vinicius", "Thiago", "Carla", "Rodrigo", "Juliana", "Ricardo", "Patrícia", "Diego", "Fernanda", "Rafaela"]
26     nome = random.choice(nomes)
27     return nome
28
29 # Função para gerar idade
30 def gerar_idade():
31     idade = random.randint(18, 30)
32     return idade
33
34 # Função para gerar altura
35 def gerar_altura():
36     altura = random.randint(1.6, 1.9)
37     return altura
38
39 # Função para gerar peso
40 def gerar_peso():
41     peso = random.randint(50, 90)
42     return peso
43
44 # Função para gerar sexo
45 def gerar_sexo():
46     sexo = random.choice(["M", "F"])
47     return sexo
48
49 # Função para gerar cor do cabelo
50 def gerar_cor_cabelo():
51     cores_cabelo = ["Preto", "Café", "Loiro", "Ruivo", "Cinza", "Verde", "Azul", "Roxo", "Laranja", "Amarelo", "Branco", "Cinza-escuro", "Cinza-claro", "Café-escuro", "Café-claro", "Loiro-escuro", "Loiro-claro", "Ruivo-escuro", "Ruivo-claro", "Cinza-escuro", "Cinza-claro", "Café-escuro", "Café-claro", "Loiro-escuro", "Loiro-claro", "Ruivo-escuro", "Ruivo-claro"]
52     cor_cabelo = random.choice(cores_cabelo)
53     return cor_cabelo
54
55 # Função para gerar cor dos olhos
56 def gerar_cor_olhos():
57     cores_olhos = ["Azul", "Verde", "Café", "Cinza", "Amarelo", "Roxo", "Laranja", "Branco", "Cinza-escuro", "Cinza-claro", "Café-escuro", "Café-claro", "Loiro-escuro", "Loiro-claro", "Ruivo-escuro", "Ruivo-claro"]
58     cor_olhos = random.choice(cores_olhos)
59     return cor_olhos
60
61 # Função para gerar cor da roupa
62 def gerar_cor_roupa():
63     cores_roupa = ["Preto", "Café", "Loiro", "Ruivo", "Cinza", "Verde", "Azul", "Roxo", "Laranja", "Amarelo", "Branco", "Cinza-escuro", "Cinza-claro", "Café-escuro", "Café-claro", "Loiro-escuro", "Loiro-claro", "Ruivo-escuro", "Ruivo-claro"]
64     cor_roupa = random.choice(cores_roupa)
65     return cor_roupa
66
67 # Função para gerar o personagem
68 def gerar_personagem():
69     nome = gerar_nome()
70     idade = gerar_idade()
71     altura = gerar_altura()
72     peso = gerar_peso()
73     sexo = gerar_sexo()
74     cor_cabelo = gerar_cor_cabelo()
75     cor_olhos = gerar_cor_olhos()
76     cor_roupa = gerar_cor_roupa()
77     return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
78
79 # Função para gerar o personagem
80 def gerar_personagem():
81     nome = gerar_nome()
82     idade = gerar_idade()
83     altura = gerar_altura()
84     peso = gerar_peso()
85     sexo = gerar_sexo()
86     cor_cabelo = gerar_cor_cabelo()
87     cor_olhos = gerar_cor_olhos()
88     cor_roupa = gerar_cor_roupa()
89     return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
90
91 # Função para gerar o personagem
92 def gerar_personagem():
93     nome = gerar_nome()
94     idade = gerar_idade()
95     altura = gerar_altura()
96     peso = gerar_peso()
97     sexo = gerar_sexo()
98     cor_cabelo = gerar_cor_cabelo()
99     cor_olhos = gerar_cor_olhos()
100    cor_roupa = gerar_cor_roupa()
101    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
102
103 # Função para gerar o personagem
104 def gerar_personagem():
105    nome = gerar_nome()
106    idade = gerar_idade()
107    altura = gerar_altura()
108    peso = gerar_peso()
109    sexo = gerar_sexo()
110    cor_cabelo = gerar_cor_cabelo()
111    cor_olhos = gerar_cor_olhos()
112    cor_roupa = gerar_cor_roupa()
113    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
114
115 # Função para gerar o personagem
116 def gerar_personagem():
117    nome = gerar_nome()
118    idade = gerar_idade()
119    altura = gerar_altura()
120    peso = gerar_peso()
121    sexo = gerar_sexo()
122    cor_cabelo = gerar_cor_cabelo()
123    cor_olhos = gerar_cor_olhos()
124    cor_roupa = gerar_cor_roupa()
125    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
126
127 # Função para gerar o personagem
128 def gerar_personagem():
129    nome = gerar_nome()
130    idade = gerar_idade()
131    altura = gerar_altura()
132    peso = gerar_peso()
133    sexo = gerar_sexo()
134    cor_cabelo = gerar_cor_cabelo()
135    cor_olhos = gerar_cor_olhos()
136    cor_roupa = gerar_cor_roupa()
137    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
138
139 # Função para gerar o personagem
140 def gerar_personagem():
141    nome = gerar_nome()
142    idade = gerar_idade()
143    altura = gerar_altura()
144    peso = gerar_peso()
145    sexo = gerar_sexo()
146    cor_cabelo = gerar_cor_cabelo()
147    cor_olhos = gerar_cor_olhos()
148    cor_roupa = gerar_cor_roupa()
149    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
150
151 # Função para gerar o personagem
152 def gerar_personagem():
153    nome = gerar_nome()
154    idade = gerar_idade()
155    altura = gerar_altura()
156    peso = gerar_peso()
157    sexo = gerar_sexo()
158    cor_cabelo = gerar_cor_cabelo()
159    cor_olhos = gerar_cor_olhos()
160    cor_roupa = gerar_cor_roupa()
161    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
162
163 # Função para gerar o personagem
164 def gerar_personagem():
165    nome = gerar_nome()
166    idade = gerar_idade()
167    altura = gerar_altura()
168    peso = gerar_peso()
169    sexo = gerar_sexo()
170    cor_cabelo = gerar_cor_cabelo()
171    cor_olhos = gerar_cor_olhos()
172    cor_roupa = gerar_cor_roupa()
173    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
174
175 # Função para gerar o personagem
176 def gerar_personagem():
177    nome = gerar_nome()
178    idade = gerar_idade()
179    altura = gerar_altura()
180    peso = gerar_peso()
181    sexo = gerar_sexo()
182    cor_cabelo = gerar_cor_cabelo()
183    cor_olhos = gerar_cor_olhos()
184    cor_roupa = gerar_cor_roupa()
185    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
186
187 # Função para gerar o personagem
188 def gerar_personagem():
189    nome = gerar_nome()
190    idade = gerar_idade()
191    altura = gerar_altura()
192    peso = gerar_peso()
193    sexo = gerar_sexo()
194    cor_cabelo = gerar_cor_cabelo()
195    cor_olhos = gerar_cor_olhos()
196    cor_roupa = gerar_cor_roupa()
197    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
198
199 # Função para gerar o personagem
200 def gerar_personagem():
201    nome = gerar_nome()
202    idade = gerar_idade()
203    altura = gerar_altura()
204    peso = gerar_peso()
205    sexo = gerar_sexo()
206    cor_cabelo = gerar_cor_cabelo()
207    cor_olhos = gerar_cor_olhos()
208    cor_roupa = gerar_cor_roupa()
209    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
210
211 # Função para gerar o personagem
212 def gerar_personagem():
213    nome = gerar_nome()
214    idade = gerar_idade()
215    altura = gerar_altura()
216    peso = gerar_peso()
217    sexo = gerar_sexo()
218    cor_cabelo = gerar_cor_cabelo()
219    cor_olhos = gerar_cor_olhos()
220    cor_roupa = gerar_cor_roupa()
221    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
222
223 # Função para gerar o personagem
224 def gerar_personagem():
225    nome = gerar_nome()
226    idade = gerar_idade()
227    altura = gerar_altura()
228    peso = gerar_peso()
229    sexo = gerar_sexo()
230    cor_cabelo = gerar_cor_cabelo()
231    cor_olhos = gerar_cor_olhos()
232    cor_roupa = gerar_cor_roupa()
233    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
234
235 # Função para gerar o personagem
236 def gerar_personagem():
237    nome = gerar_nome()
238    idade = gerar_idade()
239    altura = gerar_altura()
240    peso = gerar_peso()
241    sexo = gerar_sexo()
242    cor_cabelo = gerar_cor_cabelo()
243    cor_olhos = gerar_cor_olhos()
244    cor_roupa = gerar_cor_roupa()
245    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
246
247 # Função para gerar o personagem
248 def gerar_personagem():
249    nome = gerar_nome()
250    idade = gerar_idade()
251    altura = gerar_altura()
252    peso = gerar_peso()
253    sexo = gerar_sexo()
254    cor_cabelo = gerar_cor_cabelo()
255    cor_olhos = gerar_cor_olhos()
256    cor_roupa = gerar_cor_roupa()
257    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
258
259 # Função para gerar o personagem
260 def gerar_personagem():
261    nome = gerar_nome()
262    idade = gerar_idade()
263    altura = gerar_altura()
264    peso = gerar_peso()
265    sexo = gerar_sexo()
266    cor_cabelo = gerar_cor_cabelo()
267    cor_olhos = gerar_cor_olhos()
268    cor_roupa = gerar_cor_roupa()
269    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
270
271 # Função para gerar o personagem
272 def gerar_personagem():
273    nome = gerar_nome()
274    idade = gerar_idade()
275    altura = gerar_altura()
276    peso = gerar_peso()
277    sexo = gerar_sexo()
278    cor_cabelo = gerar_cor_cabelo()
279    cor_olhos = gerar_cor_olhos()
280    cor_roupa = gerar_cor_roupa()
281    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
282
283 # Função para gerar o personagem
284 def gerar_personagem():
285    nome = gerar_nome()
286    idade = gerar_idade()
287    altura = gerar_altura()
288    peso = gerar_peso()
289    sexo = gerar_sexo()
290    cor_cabelo = gerar_cor_cabelo()
291    cor_olhos = gerar_cor_olhos()
292    cor_roupa = gerar_cor_roupa()
293    return nome, idade, altura, peso, sexo, cor_cabelo, cor_olhos, cor_roupa
294
295 # Função para gerar o personagem
296 def gerar_personagem():
297    nome = gerar_nome()
298    idade = gerar_idade()
299    altura = gerar_altura()
300    peso = gerar_peso()
301    sexo = gerar_sexo()
302    cor_cabelo =
```

*Nota: El desarrollo del código Python se utilizó el entorno de desarrollo integrado Visual Studio Code y las pruebas de usabilidad se desarrollaron dentro de la terminal integrada.<sup>1</sup>*

*Nota: Se puede acceder al código en el repositorio del autor mediante un clic a la **Imagen 2***

### 3. Requisitos Específicos

<sup>1</sup> Microsoft. (2021). Visual Studio Code. [Software de computadora]. Microsoft. Recuperado de <https://code.visualstudio.com/>

Los requerimientos funcionales describen las acciones específicas que debe realizar el sistema. A continuación, se detallan los requerimientos funcionales del juego "*Adivina Adivinador*":

### **3.1 Requisitos Funcionales**

- **RF001:** El sistema debe mostrar una animación de inicio que consiste en una barra que se llena gradualmente para crear una sensación de anticipación antes de iniciar el juego.
- **RF002:** El sistema debe permitir a los jugadores ingresar su nombre para identificarse en el juego.
- **RF003:** El sistema debe permitir a los jugadores intentar adivinar el color oculto ingresando letras o palabras.
- **RF004:** El sistema debe proporcionar pistas y retroalimentación interactiva con emoticones basada en las respuestas proporcionadas por los jugadores.
- **RF005:** El sistema debe visualizar la representación actualizada del color oculto con las letras adivinadas correctamente.

### **3.2 Requisitos No Funcionales**

- **RNF001:** El sistema debe ser fácil de entender y utilizar, incluso para usuarios sin experiencia previa en juegos de adivinanzas.
- **RNF002:** El sistema debe ser rápido en su ejecución, proporcionando una experiencia de juego fluida.
- **RNF003:** El sistema debe ser capaz de manejar errores y situaciones inesperadas de manera adecuada.

## 4. Otros Requisitos

### 4.1 Requisitos de Implementación

- **RI001:** El sistema debe funcionar en la terminal, sin requerir una interfaz gráfica compleja. Debe proporcionar una experiencia de juego atractiva y fácil de usar utilizando solo comandos de texto.
- **RI002:** El sistema utilizará las bibliotecas estándar de Python para las funcionalidades necesarias. El juego debe ser fácil de instalar y configurar en diferentes sistemas operativos.
- **RI003:** El sistema debe proporcionar instrucciones claras para su ejecución sin problemas.
- **RI004:** El sistema debe responder rápidamente a las interacciones del usuario, proporcionando retroalimentación inmediata después de cada adivinanza.
- **RI005:** El sistema debe mostrar un código de caracteres ASCII si se gana o se pierde.

### 4.2 Requisitos de Documentación

- **RD001:** Se proporcionará documentación detallada sobre la estructura del código y el funcionamiento del sistema.

### 4.3 Requisitos de Pruebas

- **RP001:** Se realizarán pruebas para verificar su correcto funcionamiento del sistema en diferentes terminales.

### 4.4 Requisitos de Mantenimiento

- **RM001:** El sistema debe ser fácilmente de mantener y modificable para futuras actualizaciones o mejoras.

## 5. Criterios de aceptación

Los criterios de aceptación establecen los estándares que el sistema debe cumplir para ser considerado satisfactorio. A continuación, se presentan los criterios de aceptación asociados a los requerimientos principales del juego "Adivina Adivinador":

### 5.1. C1: Animación de inicio del juego

- La animación de inicio se muestra correctamente en la terminal.
- La barra se llena gradualmente, creando una sensación de carga de videojuego.

### 5.2. C2: Adivinar letra

- El sistema verifica si la letra ingresada es válida (minúscula y no un número).
- El sistema verifica si la letra ya ha sido adivinada previamente.
- Se muestra una retroalimentación adecuada en función de si la letra es correcta o incorrecta.
- Se resta un intento al jugador si la letra es incorrecta.

### 5.3. C3: Adivinar palabra

- El sistema verifica si la palabra ingresada coincide con el color oculto.
- Se muestra un mensaje de felicitaciones si la palabra es correcta y se finaliza el juego.
- Se resta un intento al jugador si la palabra es incorrecta.

### 5.4. C4: Mostrar palabra actualizada

- Después de cada intento de adivinanza, se muestra la palabra oculta actualizada, reemplazando las letras no adivinadas por guiones bajos.

### 5.5. C5: Finalizar el juego

- El juego se finaliza cuando se agotan los intentos del jugador.
- Se muestra un mensaje indicando que el jugador ha perdido y se revela el color oculto.

## 6. Diagramas

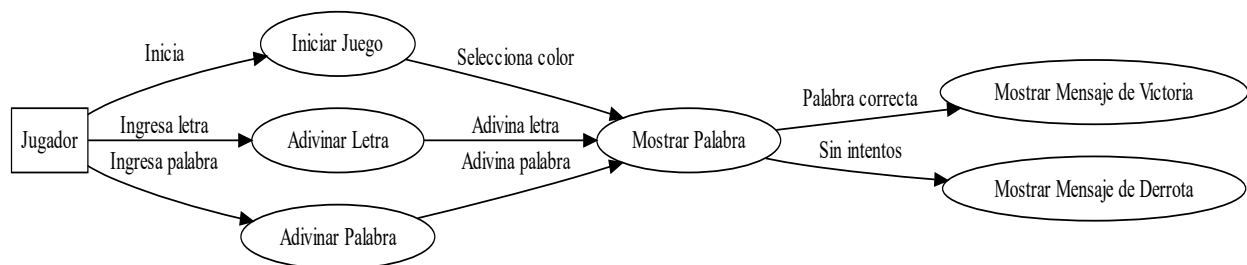
Los siguientes diagramas brindan una visión general de la interacción del usuario y la estructura interna del sistema:

### 6.1. Diagrama de casos de uso

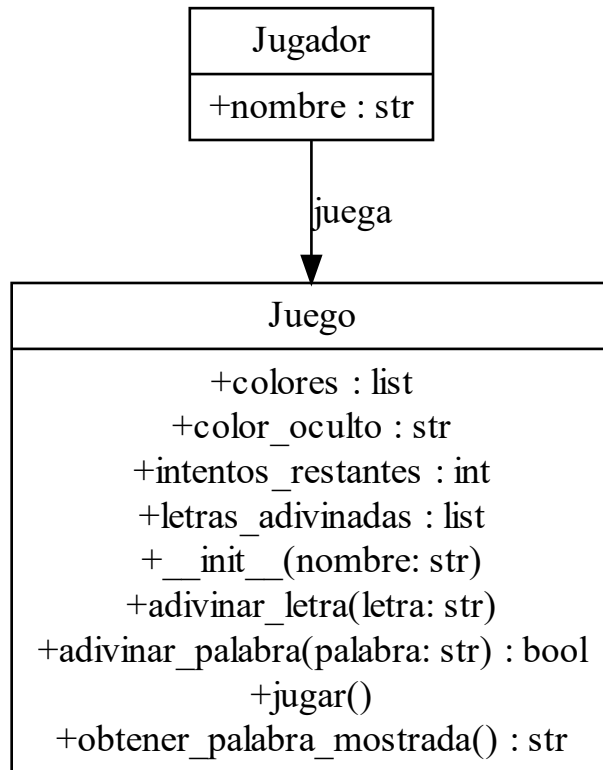
El diagrama representa la interacción del jugador con el juego "Adivina Adivinador", que es un juego de adivinanzas para adivinar un color oculto.

- **Usuario:** En el diagrama, el único actor es el "Jugador", que es quien interactúa con el sistema y realiza las acciones de juego.
- **Casos de Uso:** Los casos de uso representan las acciones o funcionalidades que el jugador puede llevar a cabo en el juego. Los casos de uso en este diagrama son:

1. **"Iniciar Juego":** Representa el caso de uso donde el jugador inicia el juego.
2. **"Adivinar Letra":** Indica que el jugador puede adivinar una letra del color.
3. **"Adivinar Palabra":** Muestra que el jugador puede adivinar toda la palabra del color.
4. **"Mostrar Palabra":** Refleja el caso de uso donde el sistema muestra la palabra oculta con las letras adivinadas por el jugador.
5. **"Mostrar Mensaje de Victoria":** Se activa cuando el jugador adivina correctamente toda la palabra.
6. **"Mostrar Mensaje de Derrota":** Se activa cuando el jugador se queda sin intentos y no adivina correctamente la palabra.



## 6.2. Diagrama de clases



## 7. Conclusiones

En conclusión, el juego "Adivina Adivinador" es una aplicación de adivinanzas que ofrece una experiencia de juego interactiva; a través de la implementación de los requerimientos funcionales y no funcionales descritos en este documento, se logrará un juego que cumple con las expectativas de los jugadores.

El juego se ejecuta en la terminal de cualquier sistema operativo, lo que lo hace accesible para una amplia gama de usuarios. El diseño de la interfaz de usuario y la interacción con el jugador se han cuidado para proporcionar una experiencia de juego fluida y agradable.

La implementación de los criterios de aceptación definidos asegurará que el juego funcione correctamente, sin errores ni fallos. Los jugadores podrán disfrutar de un juego estable y confiable, sin interrupciones inesperadas.

# Apéndice: Detalles de Funciones, Clases, Métodos y Ejecución de la aplicación en Terminal

En este apéndice, se proporcionan detalles adicionales sobre las funciones, métodos y clases del sistema "Adivina Adivinador".

## 1. Funciones:

### 1.1. adivinar\_letra(letra)

- **Descripción:** Verifica si la letra ingresada es válida y si ya ha sido adivinada anteriormente.

- Si es una letra válida y no repetida, se compara con el color oculto para determinar si es correcta.

```
def adivinar_letra(self, letra):
    if not letra.islower() or letra.isdigit():
        print("\n! Te dije que solo debes ingresar letras en minúscula ! ( →,←" )\n ")
        return

    if letra in self.letras_adivinadas:
        print("\n👤 Ya has usado esa letra ( ⬮ _ ⬮ )")
        return

    self.letras_adivinadas.append(letra)
    if letra in self.color_oculto:
        print("\n✅ ¡Adivinaste la letra! <( ^ _ ^ )>")
    else:
        self.intentos_restantes -= 1
        print("\n❌ ¡Letra incorrecta! Te quedan ▶ {} ◀ intentos. (⬮_⬮)".format(self.intentos_restantes))
```

### 1.2. adivinar\_palabra(palabra)

- **Descripción:** Verifica si la palabra ingresada por el jugador coincide con el color oculto.

- Si la palabra es correcta, el jugador ha adivinado el color oculto y gana el juego.

```
def adivinar_palabra(self, palabra):
    if palabra == self.color_oculto:
        print("😄 ★★ ★ ¡Felicidades, {}! Adivinaste el color. (ノ◡ノ)/ ͡°': ★★ ★".format(self.nombre))
        return True
    else:
        self.intentos_restantes -= 1
        print("😞 ❌ Palabra incorrecta.❌ Te quedan ▶ {} ◀ intentos.".format(self.intentos_restantes))
        return False
```

### 1.3. jugar()

- **Descripción:** Inicia el juego "Adivina Adivinador".



- [illegible]

```
def obtener_palabra_mostrada(self):
    palabra_mostrada = ""
    for letra in self.color_oculto:
        if letra in self.letras_adivinadas:
            palabra_mostrada += letra
        else:
            palabra_mostrada += "_ "
    return palabra_mostrada
```

- Atributos:
- nombre: Almacena el nombre del jugador.

## 2.2. Adivina Adivinador

- *Descripción:* Representa el juego "Adivina Adivinador".

- Al crear una instancia de esta clase, se define el nombre del jugador y se inicializan las variables necesarias para el juego,
- como la lista de colores disponibles, el color oculto seleccionado aleatoriamente, los intentos restantes y las letras adivinadas.

```
class adivina_adivinador_v12:
    def __init__(self, nombre):
        self.nombre = nombre
        self.colores = ["rojo", "azul", "verde", "amarillo", "naranja",
                        "rosa", "violeta", "negro", "blanco", "gris", "ocre", "sepia",
                        "cafe", "miel", "siena", "carmesi", "oro", "cian", "esmeralda",
                        "acua", "turquesa", "celeste", "morado", "plata"]
        self.color_oculto = random.choice(self.colores)
        self.intentos_restantes = 9
        self.letras_adivinadas = []

    def adivinar_letra(self, letra):
        # Código del método adivinar_letra

    def adivinar_palabra(self, palabra):
        # Código del método adivinar_palabra

    def jugar(self):
        # Código del método jugar

    def obtener_palabra_mostrada(self):
        # Código del método obtener_palabra_mostrada

# Bloque de código fuera de la clase
nombre_usuario = input("\n¿Como te llamas?: ")
juego = adivina_adivinador_v12(nombre_usuario)
juego.jugar()
```

### 3. Métodos:

- **\_\_init\_\_(nombre):** Constructor de la clase que recibe el nombre del jugador y realiza la inicialización de las variables.
- **adivinar\_letra (letra):** Verifica si la letra ingresada es válida y si ya ha sido adivinada anteriormente.
  - Si es una letra válida y no repetida, se compara con el color oculto para determinar si es correcta.
- **adivinar\_palabra (palabra):** Verifica si la palabra ingresada por el jugador coincide con el color oculto.
  - Si la palabra es correcta, el jugador ha adivinado el color oculto y gana el juego.
- **jugar():** Inicia el juego, solicita el nombre del jugador, presenta las instrucciones y gestiona los intentos del jugador
  - hasta que se agoten o se adivine el color oculto.
- **obtener\_palabra\_mostrada():** Genera una representación de la palabra oculta con espacios en blanco para las letras no adivinadas.
  - Esta representación se muestra al jugador para indicar qué letras ha adivinado correctamente.

```
class adivina_adivinador_v12:
    def __init__(self, nombre):
        # Código del constructor

    def adivinar_letra(self, letra):
        # Código del método adivinar_letra

    def adivinar_palabra(self, palabra):
        # Código del método adivinar_palabra

    def jugar(self):
        # Código del método jugar

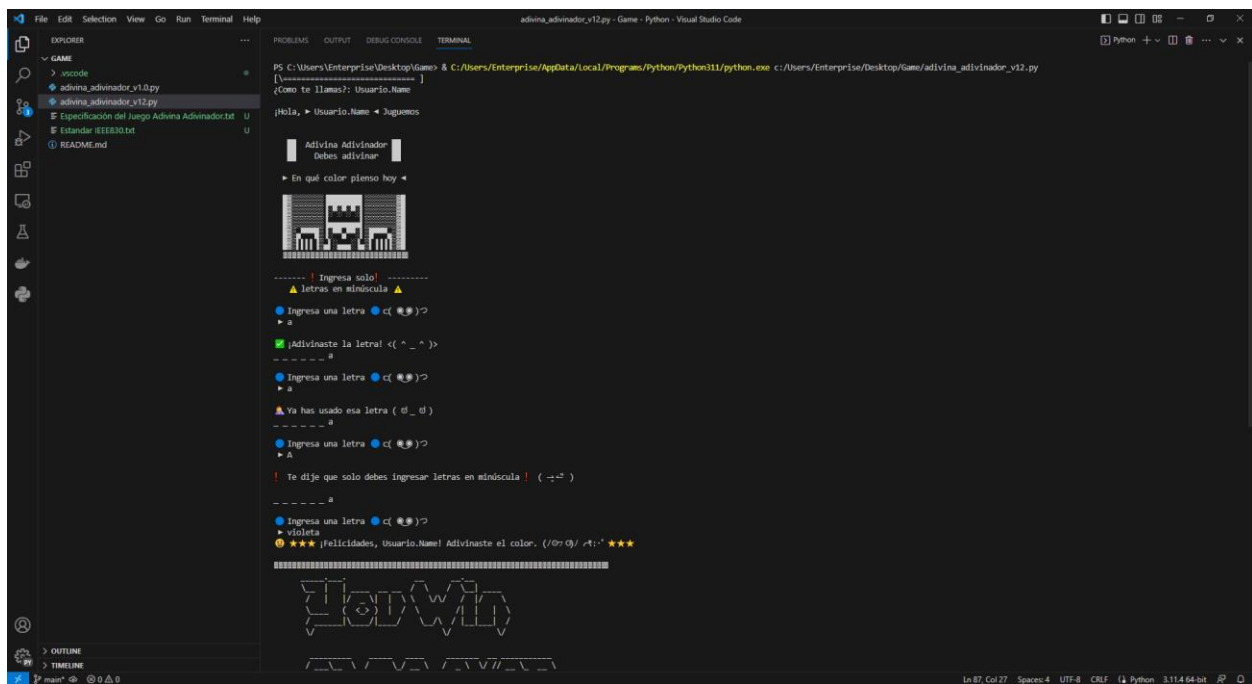
    def obtener_palabra_mostrada(self):
        # Código del método obtener_palabra_mostrada
```

## 4. Ejecución del Programa:

En esta sección, describiré los pasos para ejecutar el programa y cómo se puede ganar o perder la partida.

### 4.1. Ganar Partida Adivinando el Color

- El programa te pedirá que ingreses tu nombre para comenzar el juego.
- Una vez que hayas ingresado tu nombre, el juego generará un color oculto al azar de la lista de colores disponibles.
- Se mostrará un tablero con guiones bajos "\_" para representar las letras del color oculto.
- El objetivo es adivinar el color oculto ingresando una letra o adivinar la palabra completa del color.
- Si adivinas correctamente todas las letras del color oculto, el juego te felicitará y mostrará un mensaje de victoria junto con un mensaje especial.
- El juego finalizará y no se te pedirá más intentos.



```
PS C:\Users\Enterprise\Desktop\Game> & C:\Users\Enterprise\AppData\Local\Programs\Python\Python311\python.exe c:\Users\Enterprise\Desktop\Game\adivina_adivinator_v12.py
[Welcome message]
(Como te llamas?): Usuario_Name
¡Hola, Usuario_Name! Jugamos

Adivina Adivinador
Debes adivinar

En qué color pienso hoy

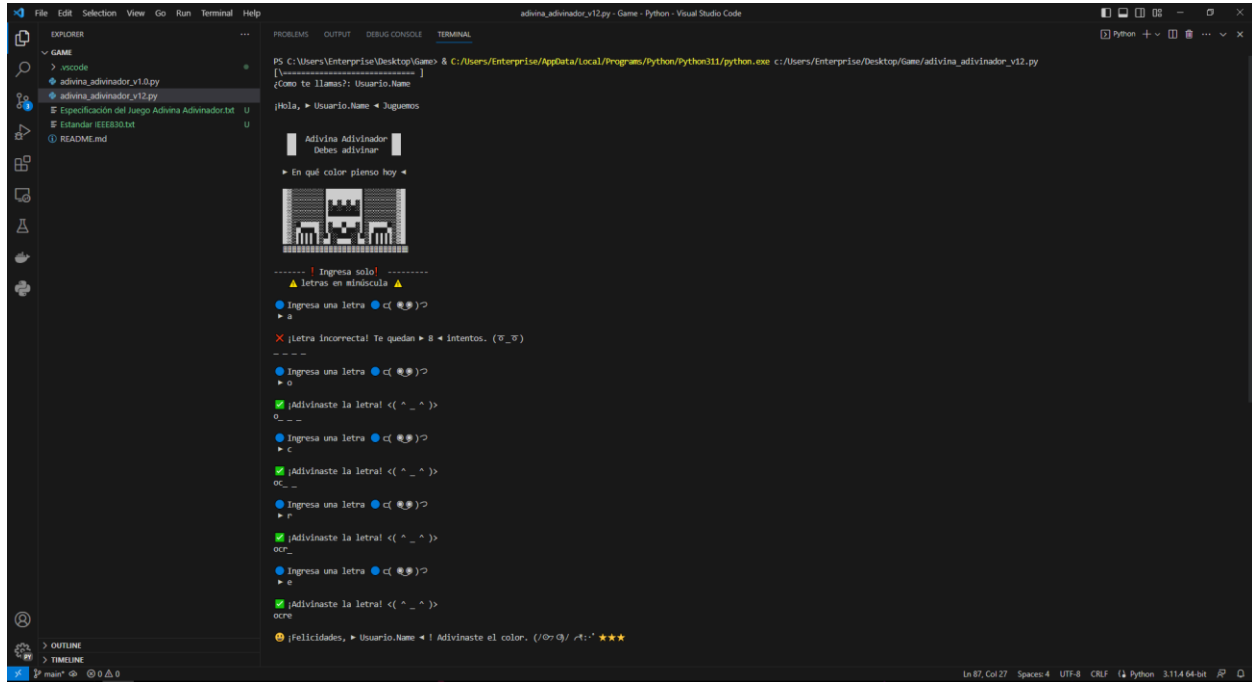
[Color selection interface]

----- ¡ Ingrese solo! -----
▲ letras en minúscula ▲
● Ingrese una letra ● c( )?
▶ a
■ Adivinaste la letra! <( ^ _ ^ )>
-----
● Ingrese una letra ● c( )?
▶ a
▲ Ya has usado esa letra ( d _ d )
-----
● Ingrese una letra ● c( )?
▶ A
! Te dije que solo debes ingresar letras en minúscula ! ( -.- )
-----
● Ingrese una letra ● c( )?
▶ violeta
● ★★★★★ (felicidades, Usuario_Name! Adivinaste el color. /O\O\ /!\: ★★★★★

[Word grid showing VIOLETA]
```

#### 4.1. Ganar Partida Adivinando las letras

- Si adivinas correctamente todas las letras del color oculto, el juego te notificará que adivinaste la letra y mostrará el color oculto hasta ese momento.
- El juego continuará y te permitirá seguir adivinando más letras para completar el color oculto.
- Una vez que hayas adivinado todas las letras del color, el juego te felicitará y mostrará un mensaje de victoria junto con un mensaje especial.
- El juego finalizará y no se te pedirá más intentos.



```
PS C:\Users\Enterprise\Desktop\Game> & C:/Users/Enterprise/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Enterprise/Desktop/Game/adivina_adivinator_v12.py
[User:Enterprise]
(Como te llamas?): Usuario.Name
{Hola, Usuario.Name < Usuario.Name}

Adivina Adivinador
Debes adivinar

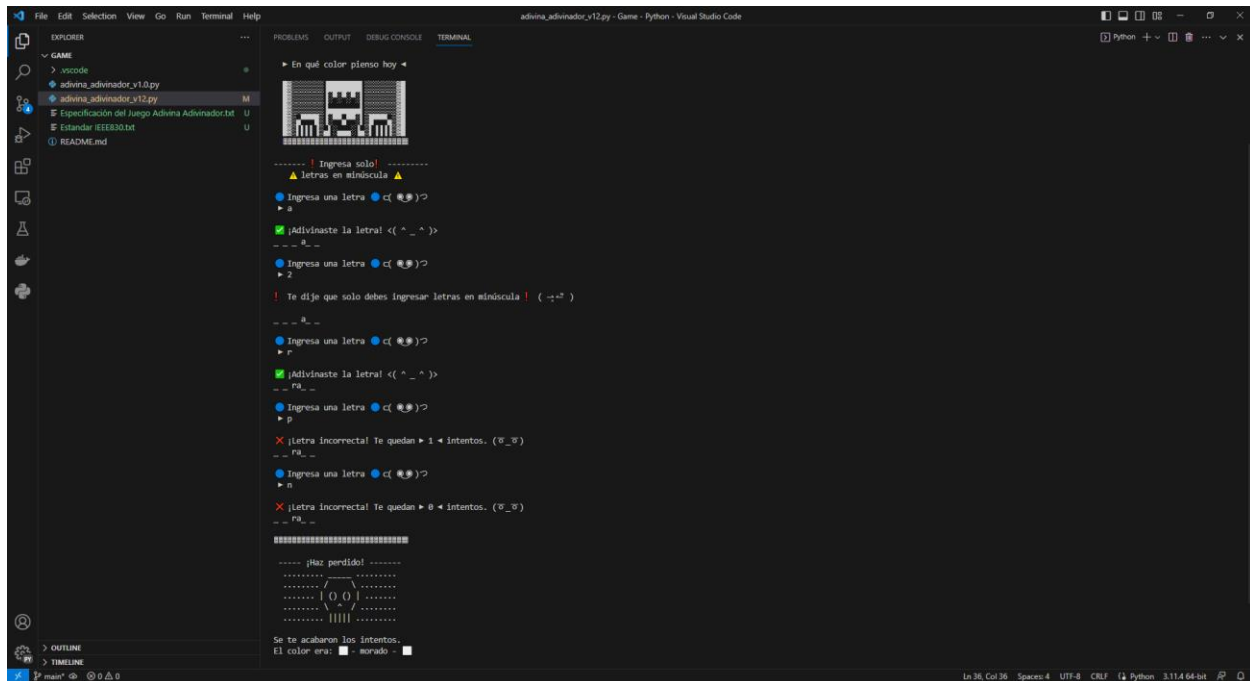
En qué color pienso hoy <

[Progress Bar: 4 bars filled, 1 bar empty]

----- Ingresar solo -----
▲ letras en minúscula ▲
● Ingresar una letra ● (c) (e) (o) (r)
> a
X ¡Letra incorrecta! Te quedan 8 intentos. (W_0)
-----
● Ingresar una letra ● (c) (e) (o) (r)
> o
✓ ¡Adivinaste la letra! < (^ _ ^ ) >
o _ _ _
● Ingresar una letra ● (c) (e) (o) (r)
> c
✓ ¡Adivinaste la letra! < (^ _ ^ ) >
oc _ _
● Ingresar una letra ● (c) (e) (o) (r)
> r
✓ ¡Adivinaste la letra! < (^ _ ^ ) >
ocr _
● Ingresar una letra ● (c) (e) (o) (r)
> e
✓ ¡Adivinaste la letra! < (^ _ ^ ) >
ocre
● ¡Felicidades, Usuario.Name < ! Adivinaste el color. (/O\ Q/ A:.' ★★★★★
```

## 4.1. Perder Partida

- Si agotas los intentos sin haber adivinado correctamente el color oculto, el juego te notificará que has perdido.
- Se mostrará un dibujo ASCII de un rostro triste junto con el mensaje de que se acabaron los intentos.
- Luego, se revelará el color oculto y se te informará cuál era el color que debías adivinar.



```
File Edit Selection View Go Run Terminal Help
adivina_adivinator_v12.py - Game - Python - Visual Studio Code

EXPLORER
> GAME
  adivina_adivinator_v12.py
  adivina_adivinator_v12.py
  Especificación del Juego Adivina Adivinator.txt
  [Standard: IEEE30.txt]
  README.md

TERMINAL
Python 3.11.4 64-bit

En qué color pienso hoy
----- ¡ Ingrese solo! -----
▲ letras en minúscula ▲
Ingresar una letra c( 読夢 )?
> a
¡Adivinaste la letra! <( ^ _ ^ )>
-- _ _ _
Ingresar una letra c( 読夢 )?
> 2
! Te dije que solo debes ingresar letras en minúscula ! ( -_- )
-- _ _ _
Ingresar una letra c( 読夢 )?
> p
¡Adivinaste la letra! <( ^ _ ^ )>
-- _ _ _
Ingresar una letra c( 読夢 )?
> p
X ¡Letra incorrecta! Te quedan 1 * intentos. ( @ _ @ )
-- _ _ _
Ingresar una letra c( 読夢 )?
> n
X ¡Letra incorrecta! Te quedan 0 * intentos. ( @ _ @ )
-- _ _ _

#####
----- ¡Perdido! -----
----- _ _ _ _ _
----- f _ _ _ _
----- | O O |
----- \ ^ /
----- |||||
Se te acabaron los intentos.
El color era: ■ morado ■
```

## 5. Enlace a repositorio de Proyecto “Adivina Adivinador”

En el siguiente enlace se puede acceder al repositorio del proyecto mediante el escaneo del código QR o dándole clic a la Imagen 1 o a la Imagen 2 para tener enlace directo al código del programa.

**Imagen 3**

Enlace QR al Repositorio del Proyecto en *Github*.



CamaradaYorch. (2023, julio 16). *GitHub - CamaradaYorch/Adivina\_Adivinador: Juego de adivinanzas para adivinar un color oculto.* GitHub.  
[https://github.com/CamaradaYorch/Adivina\\_Adivinador/tree/main](https://github.com/CamaradaYorch/Adivina_Adivinador/tree/main)