



DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS CUÁNTICOS PARA PROBLEMAS DE APRENDIZAJE AUTOMÁTICO SUPERVISADO

GRADO EN INGENIERÍA INFORMÁTICA DEL SOFTWARE

TRABAJO FIN DE GRADO

AUTOR

Alba Aparicio Pérez

TUTOR

Elías Fernández-Combarro Álvarez

Julio 2022
Versión 1.0.

Agradecimientos

Gracias a mi familia y amigos por todo el apoyo, en especial a Marta y Diana.

Índice general

1. Introducción	11
1.1. Resumen	11
1.2. Palabras Clave	13
1.3. Abstract	14
1.4. Keywords	15
2. Aspectos Teóricos	16
2.1. Introducción	16
2.2. La computación cuántica	18
2.2.1. Introducción a las matemáticas detrás de la computación cuántica	18
2.2.2. Introducción a la física detrás de la computación cuántica .	22
2.2.3. Teoría de la computación cuántica	25
2.2.4. Hardware de los ordenadores cuánticos	31
2.3. Aprendizaje Automático	33
2.3.1. Introducción	33
2.3.2. Modelos de Aprendizaje Automático	33
2.3.3. Cómo construir un sistema inteligente	34
2.4. Aprendizaje Automático Cuántico	38
2.4.1. Introducción	38
2.4.2. QSVM	38
2.4.3. Redes neuronales cuánticas	42
3. PLANIFICACIÓN DEL SISTEMA DE INFORMACIÓN	46
3.1. INICIO DEL PLAN DE SISTEMAS DE INFORMACIÓN	47
3.1.1. Análisis de la Necesidad del PSI	47
3.1.2. Identificación del Alcance del PSI	47
3.1.3. Determinación de Responsables	48
3.2. DEFINICIÓN Y ORGANIZACIÓN DEL PSI	49
3.2.1. Especificación del Ámbito y Alcance	49
4. DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA	52
4.1. Identificación de las Necesidades de Infraestructura Tecnológica . .	53
4.2. Selección de la Arquitectura Tecnológica	54

5. ESTUDIO DE VIABILIDAD DEL SISTEMA	55
5.1. ESTUDIO Y VALORACIÓN DE ALTERNATIVAS DE SOLUCIÓN. SELECCIÓN DE ALTERNATIVA FINAL	56
5.1.1. PennyLane vs Qiskit	56
5.1.2. Google vs IBM	57
5.1.3. Java vs C# vs Python	57
6. PLANIFICACIÓN Y GESTIÓN DEL TFG	58
6.1. PLANIFICACIÓN DEL PROYECTO	59
6.1.1. Identificación de Interesados	59
6.1.2. OBS y PBS	59
6.1.3. Planificación Inicial. WBS	60
6.1.4. Riesgos	60
6.1.5. Presupuesto Inicial	69
6.2. EJECUCIÓN DEL PROYECTO	70
6.2.1. Plan Seguimiento de Planificación	70
6.2.2. Bitácora de Incidencias del Proyecto	70
6.2.3. Riesgos	71
6.3. CIERRE DEL PROYECTO	73
6.3.1. Planificación Final	73
6.3.2. Informe Final de Riesgos	73
6.3.3. Presupuesto Final de Costes	74
6.3.4. Informe de Lecciones Aprendidas	75
7. ANÁLISIS DEL SISTEMA DE INFORMACIÓN	77
7.1. DEFINICIÓN DEL SISTEMA	78
7.1.1. Determinación del Alcance del Sistema	78
7.2. ESTABLECIMIENTO DE REQUISITOS	79
7.2.1. Obtención de los Requisitos del Sistema	79
7.2.2. Identificación de Actores del Sistema	82
7.2.3. Especificación de Casos de Uso	83
7.3. ANÁLISIS DE LOS CASOS DE USO	86
7.3.1. Caso de Uso - Obtener un ordenador cuántico de IBM	86
7.3.2. Caso de Uso - Cargar la cuenta de IBM guardada en disco	89
7.3.3. Caso de Uso - Activar cuenta de IBM	90
7.3.4. Caso de Uso - Ejecutar QSVM en un simulador cuántico	93
7.3.5. Caso de Uso - Ejecutar QSVM en un ordenador cuántico	94
7.3.6. Caso de Uso - Ejecutar QNN en un simulador cuántico	97
7.3.7. Caso de Uso - Ejecutar QNN en un ordenador cuántico	98
7.4. ANÁLISIS DE CLASES	101
7.4.1. Diagrama de Clases	101
7.4.2. Descripción de las Clases	101
7.5. DEFINICIÓN DE INTERFACES DE USUARIO	106
7.5.1. Descripción de la Interfaz	106
7.5.2. Descripción del Comportamiento de la Interfaz	111
7.5.3. Diagrama de Navegabilidad	112

7.6.	ESPECIFICACIÓN DEL PLAN DE PRUEBAS	114
7.6.1.	Pruebas unitarias	114
7.6.2.	Pruebas de sistema	114
7.6.3.	Pruebas de integración	115
8.	DISEÑO DEL SISTEMA DE INFORMACIÓN	116
8.1.	DISEÑO DE CASOS DE USO REALES	117
8.1.1.	Caso de Uso - Obtener un ordenador cuántico de IBM . . .	117
8.1.2.	Caso de Uso - Cargar la cuenta de IBM guardada en disco .	118
8.1.3.	Caso de Uso - Activar cuenta de IBM	119
8.1.4.	Caso de Uso - Ejecutar QSVM en un simulador cuántico . .	120
8.1.5.	Caso de Uso - Ejecutar QSVM en un ordenador cuántico .	122
8.1.6.	Caso de Uso - Ejecutar QNN en un simulador cuántico . .	123
8.1.7.	Caso de Uso - Ejecutar QNN en un ordenador cuántico . .	124
8.2.	DISEÑO DE CLASES	126
8.2.1.	Diagrama de Clases	126
8.3.	DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SIS- TEMA	128
8.3.1.	Diseño de Módulos del Sistema	128
8.3.2.	Diseño de Comunicaciones entre Módulos	129
8.4.	DISEÑO FÍSICO DE DATOS	132
8.4.1.	Diagrama E-R	132
8.5.	ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS	133
8.5.1.	Pruebas Unitarias	133
8.5.2.	Pruebas de Integración	133
8.5.3.	Pruebas de Sistema	134
9.	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN	136
9.1.	PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONS- TRUCCIÓN	137
9.1.1.	Estándares y normas seguidos	137
9.1.2.	Lenguajes de programación	137
9.1.3.	Herramientas y programas usados para el desarrollo	137
9.2.	GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES Y PRO- CEDIMIENTOS	139
9.2.1.	QMLAlgorithm	139
9.2.2.	QuantumModel	140
9.2.3.	Executor	141
9.3.	EJECUCIÓN DE LAS PRUEBAS	143
9.4.	ELABORACIÓN DE LOS MANUALES DE USUARIO	149
9.4.1.	Manual de Instalación	149
9.4.2.	Manual de Ejecución	151
9.4.3.	Manual de Usuario	152
9.4.4.	Manual del Programador	153

10.CONCLUSIONES Y AMPLIACIONES	155
10.1. CONCLUSIONES	156
10.2. AMPLIACIONES	157
ANEXOS	158
PLAN DE GESTIÓN DE RIESGOS	159
PRESUPUESTO	162
Definición de empresa	162
Investigación	163
Desarrollo del prototipo	166
Documentación	170
Otros costes	175
Presupuesto de costes	175
PRESUPUESTO FINAL	176
Definición de empresa	176
Investigación	176
Documentación	176
Otros costes	180
Presupuesto de costes	180
CONTENIDO ENTREGADO EN LOS ANEXOS	181

Índice de figuras

2.1. Representación cartesiana de números complejos	19
2.2. Representación de números complejos en forma polar	19
2.3. Experimento de la doble rendija	23
2.4. Esfera de Bloch	26
2.5. Regresión lineal	34
2.6. SVM	39
2.7. SVM. Margen blando.	40
2.8. Proceso seguido en problemas de QML	41
2.9. SVM. Feature map.	41
2.10. Perceptrón Multi-Capa	43
2.11. Arquitectura de una QNN	44
4.1. Esquema de la infraestructura tecnológica	53
6.1. OBS	60
6.2. PBS General	61
6.3. PBS Investigacion	61
6.4. PBS Desarrollo	62
6.5. PBS Documentacion	62
6.6. WBS. General	63
6.7. WBS. Investigación	63
6.8. WBS. Desarrollo	64
6.9. WBS. Documentación	64
6.10. Hitos del proyecto (1)	65
6.11. Hitos del proyecto (2)	65
6.12. Hitos del proyecto (3)	66
6.13. Presupuesto de Costes	69
6.14. Presupuesto de Cliente detallado	70
6.15. Presupuesto de Cliente resumido	71
6.16. Planificación Final	73
6.17. Presupuesto final de Costes	74
6.18. Presupuesto final de cliente detallado	75
6.19. Presupuesto final de cliente resumido	76
7.1. Casos de uso	84
7.2. Diagrama de robustez - Obtener un ordenador cuántico de IBM . .	87
7.3. Diagrama de robustez - Cargar la cuenta de IBM guardada en disco	88

7.4. Diagrama de robustez - Activar cuenta de IBM	91
7.5. Diagrama de robustez - Ejecutar QSVM en un simulador cuántico	92
7.6. Diagrama de robustez - Ejecutar QSVM en un ordenador cuántico	95
7.7. Diagrama de robustez - Ejecutar QNN en un simulador cuántico .	96
7.8. Diagrama de robustez - Ejecutar QNN en un ordenador cuántico .	99
7.9. Diagrama de Clases	102
7.10. Boceto - Pantalla principal	106
7.11. Boceto - Pantalla del algoritmo QSVM	107
7.12. Boceto - Pantalla del algoritmo QNN	108
7.13. Boceto - Pantalla de configuración de la ejecución en un simulador	109
7.14. Boceto - Pantalla de configuración de la ejecución en un ordenador de IBM	109
7.15. Boceto - Pantalla de salida	110
7.16. Boceto - Pantalla de documentación	110
7.17. Boceto - Pantalla de contacto	111
7.18. Boceto - Pantalla de mensaje de error	113
7.19. Diagrama de navegabilidad	113
8.1. Diagrama de interacción - Obtener un ordenador cuántico de IBM	117
8.2. Diagrama de estados - Cuenta IBM	118
8.3. Diagrama de interacción - Cargar la cuenta de IBM guardada en disco	119
8.4. Diagrama de interacción - Activar cuenta de IBM	120
8.5. Diagrama de interacción - Ejecutar QSVM en un simulador cuántico	121
8.6. Diagrama de estados - Ejecución	122
8.7. Diagrama de interacción - Ejecutar QSVM en un ordenador cuántico	123
8.8. Diagrama de interacción - Ejecutar QNN en un simulador cuántico	124
8.9. Diagrama de interacción - Ejecutar QNN en un ordenador cuántico	125
8.10. Diseño - Diagrama de clases	127
8.11. Diagrama de paquetes	129
8.12. Diagrama de despliegue	130
8.13. Diagrama de componentes	131
8.14. Diseño - Diagrama de Entidad-Relación	132
9.1. Ejemplo de ejecución en local. QSVM	145
9.2. Ejemplo de ejecución en local. QNN	146
9.3. Pantalla de error	147
9.4. Cobertura de pruebas	148
9.5. Instalación de dependencias	150
9.6. Instalación de dependencias en el entorno virtual	151
9.7. Ejecución del entorno Flask	152
9.8. Pantalla principal	153
9.9. Obtención del TOKEN de IBM	154
10.1. Clasificación de riesgos	160
10.2. Matriz de probabilidad e impacto	161
10.3. Presupuesto. Resumen.	162

10.4. Presupuesto. Costes indirectos.	162
10.5. Presupuesto. Medios de producción.	162
10.6. Presupuesto. Personal.	163
10.7. Presupuesto. Personal. Precio por Hora.	163
10.8. Presupuesto. Personal Externo. Precio por Hora.	164
10.9. Presupuesto. Investigación	165
10.10. Presupuesto. Desarrollo del prototipo (1)	167
10.11. Presupuesto. Desarrollo del prototipo (2)	168
10.12. Presupuesto. Desarrollo del prototipo (3)	169
10.13. Presupuesto. Documentación (1)	171
10.14. Presupuesto. Documentación (2)	172
10.15. Presupuesto. Documentación (3)	173
10.16. Presupuesto. Otros costes.	174
10.17. Presupuesto de Costes	175
10.18. Presupuesto final. Documentación (1)	177
10.19. Presupuesto final. Documentación (2)	178
10.20. Presupuesto final. Documentación (3)	179
10.21. Presupuesto final de Costes	180

Índice de tablas

2.1. Representación Puerta I	28
2.2. Representación Puerta X	28
2.3. Representación Puerta Y	28
2.4. Representación Puerta Z	29
2.5. Representación Puerta H	29
2.6. Representación Puerta S	29
2.7. Representación Puerta SWAP	30
2.8. Representación Puerta CNOT	30
6.1. Ponderación de Costes	69
6.2. Ponderación final de Costes	75
7.1. Especificación Caso de Uso - Obtener un ordenador cuántico de IBM	83
7.2. Especificación Caso de Uso - Cargar la cuenta de IBM guardada en disco	84
7.3. Especificación Caso de Uso - Activar cuenta de IBM	84
7.4. Especificación Caso de Uso - Ejecutar QSVM en un simulador cuántico	85
7.5. Especificación Caso de Uso - Ejecutar QSVM en un ordenador cuántico	85
7.6. Especificación Caso de Uso - Ejecutar QNN en un simulador cuántico	85
7.7. Especificación Caso de Uso - Ejecutar QNN en un ordenador cuántico	85
7.8. Análisis del Caso de Uso - Obtener un ordenador cuántico de IBM	89
7.9. Análisis del Caso de Uso - Cargar la cuenta de IBM guardada en disco	90
7.10. Análisis del Caso de Uso - Activar cuenta de IBM	93
7.11. Análisis del Caso de Uso - Ejecutar QSVM en un simulador cuántico	94
7.12. Análisis del Caso de Uso - Ejecutar QSVM en un ordenador cuántico	97
7.13. Análisis del Caso de Uso - Ejecutar QNN en un simulador cuántico	98
7.14. Análisis del Caso de Uso - Ejecutar QNN en un ordenador cuántico	100
7.15. Descripción de diseño de QMLAlgorithm	101
7.16. Descripción de diseño de Executor	103
7.17. Descripción de diseño de IBMExecutor	103
7.18. Descripción de diseño de LocalExecutor	103
7.19. Descripción de diseño de Validator	104
7.20. Descripción de diseño de Dataset	104

7.21. Descripción de diseño de QuantumModel	105
7.22. Descripción de diseño de QSVCModel	105
7.23. Descripción de diseño de QNNModel	105
8.1. Pruebas unitarias	133
8.2. Pruebas de integración	134
8.3. Pruebas de sistema	135
9.1. Descripción de diseño de QMLAlgorithm	140
9.2. Descripción de diseño de QuantumModel	141
9.3. Descripción de diseño de Executor	142

Capítulo 1

Introducción

1.1. Resumen

La computación cuántica es un nuevo paradigma de computación que ha cogido mucha fuerza en estos últimos años. Este paradigma es una fusión entre la mecánica cuántica y la ciencia de la computación, que aprovecha las características de la mecánica cuántica y de las partículas subatómicas, tales como la superposición o el entrelazamiento cuántico, para lograr obtener modelos computacionales totalmente diferentes a los modelos clásicos.

Este tipo de computación nos resulta extremadamente útil en muchos ámbitos diferentes, pero en este trabajo se profundizará acerca de su uso en el ámbito de la inteligencia artificial (IA).

Como la computación cuántica, la inteligencia artificial es una disciplina que está en pleno auge. Entendemos como IA cualquier sistema o pieza de software que sea capaz de percibir su entorno y tomar decisiones para cumplir algún objetivo o tarea.

La rama de la IA en la que se centra este trabajo es el aprendizaje automático, el cual crea sistemas que tengan la capacidad de aprender a resolver una tarea, sin estar programados explícitamente para ella.

El objetivo de este trabajo es explorar la unión de ambos campos, dando lugar a un estudio sobre aprendizaje automático cuántico. Se aplicarán técnicas y modelos del aprendizaje automático usando el paradigma de la computación cuántica, además de realizar ejecuciones tanto en simuladores como en computadores cuánticos reales.

En concreto, se resolverán problemas de aprendizaje supervisado, que son aquellos en los que tenemos un conjunto de datos de entrada etiquetados a partir de los cuales nuestro computador aprenderá lo necesario para resolver el problema planteado.

Se espera que la computación cuántica nos permita resolver problemas que con la computación clásica resultan inviables. Por ello, con este trabajo se busca

conseguir resolver problemas de aprendizaje automático, contribuyendo así a la comprensión de estas áreas de estudio tan novedosas.

1.2. Palabras Clave

Computación cuántica, Aprendizaje Automático, QSVM, QNN, Inteligencia Artificial, Aprendizaje Supervisado

1.3. Abstract

Quantum computing is a new computing paradigm that has gained a lot of momentum in recent years. This paradigm is a fusion between quantum mechanics and computer science, which takes advantage of the characteristics of quantum mechanics and subatomic particles, such as superposition or quantum entanglement, to obtain computational models totally different from classical ones.

This type of computation is extremely useful in many different fields, but this paper will focus on its use in the field of artificial intelligence (AI).

Like quantum computing, artificial intelligence is a discipline that is booming. We understand AI as any system or piece of software that is able to perceive its environment and make decisions to accomplish some goal or task.

The branch of AI on which this work focuses is machine learning, which creates systems that have the ability to learn to solve a task, without being explicitly programmed to do so.

The goal of this work is to explore the union of both fields, resulting in a study on quantum machine learning. Machine learning techniques and models will be applied using the quantum computing paradigm, and will be run on both simulators and real quantum computers.

Specifically, supervised learning problems will be solved, which are those in which we have a set of labelled data from which our computer will learn what is necessary to solve the problem.

It is expected that quantum computing will allow us to solve problems that are unfeasible with classical computing. Therefore, the aim is to solve machine learning problems, thus contributing to the understanding of these new areas of study.

1.4. Keywords

Quantum Computing, Machine Learning, QSVM, QNN, Artificial Intelligence, Supervised Learning

Capítulo 2

Aspectos Teóricos

2.1. Introducción

Primero de todo necesitamos saber qué es un ordenador cuántico y cómo funciona.

Un ordenador clásico utiliza una serie de bits, que pueden estar en estado 0 o estado 1. Por otro lado, en computación cuántica, la unidad mínima de información es el **qubit**, que sería el equivalente al bit en computación clásica y puede estar en una **combinación** de los estados 0 y 1. Esta es una de las diferencias fundamentales y una de las bases del poder de los ordenadores cuánticos con respecto a los clásicos.

En un ordenador clásico con n bits, la cantidad de información que contiene un estado concreto de la máquina tiene tamaño n , ya que es una cadena de unos y ceros. Sin embargo, en un ordenador cuántico con n **qubits**, un estado concreto de la máquina es una combinación de todas las posibles cadenas de n unos y ceros.

Hay 2^n posibles combinaciones, por lo que la cantidad de información que contiene un estado concreto de un ordenador cuántico tiene tamaño 2^n .

Como ya sabemos, en un ordenador clásico, para pasar de un estado a otro lo que se hace es usar una operación lógica sobre los bits que definen el estado actual del ordenador. Dichas operaciones se llaman puertas lógicas. La unión de varias puertas lógicas forma un algoritmo.

Un ordenador cuántico funciona de una forma parecida, ya que pasamos de un estado a otro usando **puertas lógicas cuánticas**, que se explicarán más en detalle en el siguiente apartado.

Aprovechando las propiedades de los sistemas cuánticos, tendremos la ventaja de tener sistemas con unas capacidades mucho más potentes que nos permitirán resolver problemas, que con un ordenador clásico, son muy costosos. Uno de

esos problemas, que es el principal objetivo de este trabajo, es la creación y entrenamiento de algoritmos de machine learning.

En la computación cuántica, el estado de un qubit está basado en probabilidades, por lo que no podemos conocer a priori cuál será el resultado de una ejecución con un 100 % de seguridad aunque conozcamos todos los cambios realizados, entradas, y salidas que han dado lugar a la obtención de dicho estado.

Esto sucede porque la mecánica cuántica se rige por probabilidades. En la teoría cuántica, según la interpretación de Copenhague¹, se dice que las propiedades de una partícula no están determinadas hasta el mismo momento de ser medidas.

Por otro lado, el aprendizaje automático es una rama de la computación que está en auge [1].

Un algoritmo de aprendizaje automático para problemas complejos como el reconocimiento de voz necesitará millones de ejemplos [2], lo que puede hacer muy costoso el procesamiento de todos estos datos.

Teniendo en cuenta la naturalidad probabilística de la computación cuántica, daré paso a explicar cómo podemos crear un algoritmo cuántico y la teoría detrás de todo este proceso.

¹Para saber más sobre las diferentes interpretaciones y otros temas similares recomiendo el canal de youtube *Date un Vlog*, en específico este vídeo: https://www.youtube.com/watch?v=GZJR_01QhGY

2.2. La computación cuántica

2.2.1. Introducción a las matemáticas detrás de la computación cuántica

En este apartado se explicará de forma breve las matemáticas necesarias para comprender la naturaleza de la computación cuántica, dejando de lado aquellos aspectos que sean más técnicos o complejos.

La mecánica cuántica está basada en álgebra lineal, por lo que para entender el funcionamiento de la computación cuántica, primero es necesario tener unas bases claras sobre este campo de las matemáticas.

2.2.1.1. Números complejos

Los números complejos son el núcleo de la mecánica cuántica y, por lo tanto, son absolutamente esenciales para una comprensión básica de la computación cuántica.

En mecánica cuántica, los estados de las partículas se representan mediante vectores de números complejos, y la evolución del sistema cuántico se puede representar mediante matrices de números complejos.

Se define la unidad imaginaria i como la raíz cuadrada de -1, es decir $i = \sqrt{-1}$

Un número complejo z es la suma de un número real a más un número real b multiplicado por la unidad imaginaria i , como en:

$$z = a + b \cdot i.$$

El número real a se llama parte real del complejo z y el número real b se llama parte imaginaria de z .

El conjunto de todos los números complejos se representa por \mathbb{C} .

Podemos realizar operaciones matemáticas con este conjunto, incluyendo la suma y la multiplicación entre otras. Lo importante es saber que

$$i^2 = \sqrt{-1}^2 = -1.$$

Para representar números complejos podemos hacerlo de dos formas: mediante la representación cartesiana o mediante la representación polar.

En la representación cartesiana, los números reales se representan en el plano en el eje horizontal, y los números imaginarios se representan en el eje vertical. Se puede ver representado en la Figura 2.1 [3].

Para número complejo dado en forma polar obtenemos el argumento de un número complejo z , que será el ángulo que el vector correspondiente forma con el semieje real positivo.

Se escribe $z = r_\alpha$, siendo α el llamado argumento principal. Así, el argumento principal es un ángulo comprendido entre 0° y 360° . Se puede ver representado en la Figura 2.2 [3].

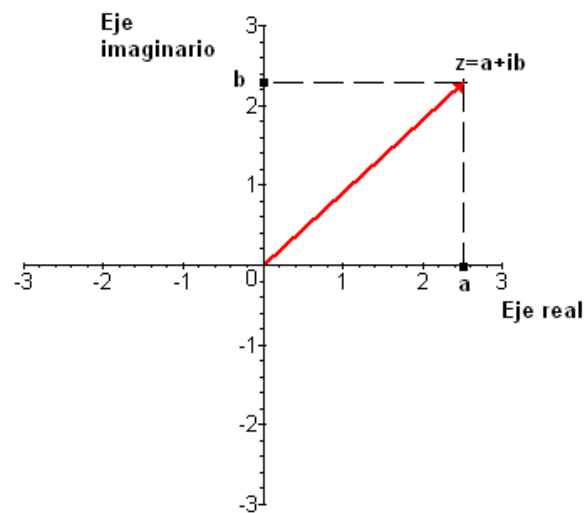


Figura 2.1: Representación cartesiana de números complejos

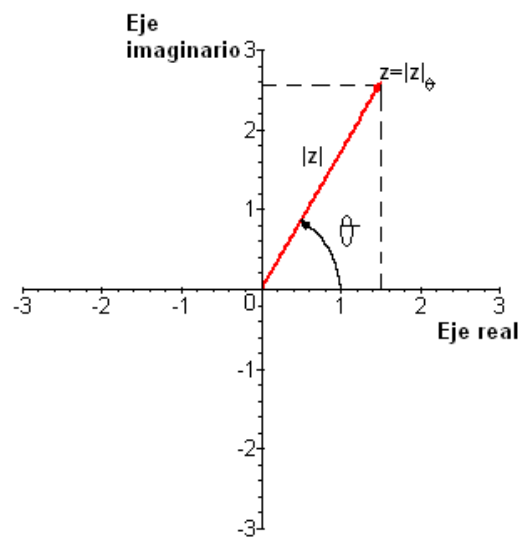


Figura 2.2: Representación de números complejos en forma polar

Notación

En mecánica cuántica y computación cuántica veremos unos símbolos que se repiten continuamente, pero que no veremos mucho fuera de este entorno. Este símbolo es la notación estándar para describir los estados cuánticos en la teoría de la mecánica cuántica:

$$|\psi\rangle$$

Esto se conoce como la **notación bra-ket** o la **notación de Dirac**. Es llamada así porque el producto interno de dos estados es denotado por $\langle\phi|\psi\rangle$, consistiendo en una parte izquierda, $\langle\phi|$, llamada *bra*, y una parte derecha, $|\psi\rangle$, llamada *ket*. En notación matricial, un ket es un vector columna y un bra es un vector fila.

2.2.1.2. Espacios vectoriales

Los espacios vectoriales complejos son muy importantes porque, en computación cuántica, los estados vienen representados por vectores complejos y los cambios de estados vienen representados mediante matrices unitarias.

Un espacio vectorial es una estructura algebraica que se compone de unos elementos llamados *vectores*. Para este trabajo nos interesa conocer el espacio vectorial complejo \mathbb{C}^n , que incluye todas las n -tuplas de números complejos (z_1, \dots, z_n) , siendo n un entero positivo que indica el número de elementos que tienen sus vectores [4]. Podemos representarlos usando la notación matricial

$$\begin{bmatrix} z_1 \\ \dots \\ z_n \end{bmatrix}$$

Para realizar una operación de suma con otro vector, se suma cada elemento de un vector más el elemento en la misma posición del otro vector de forma que

$$\begin{bmatrix} z_1 \\ \dots \\ z_n \end{bmatrix} + \begin{bmatrix} z'_1 \\ \dots \\ z'_n \end{bmatrix} \equiv \begin{bmatrix} z_1 + z'_1 \\ \dots \\ z_n + z'_n \end{bmatrix},$$

donde la operación de suma de la derecha es una simple suma de números complejos. De la misma forma, en un espacio vectorial también podemos realizar una multiplicación escalar de la forma

$$z \begin{bmatrix} z_1 \\ \dots \\ z_n \end{bmatrix} \equiv \begin{bmatrix} z \cdot z_1 \\ \dots \\ z \cdot z_n \end{bmatrix},$$

donde z es un escalar, que es un número complejo, y las multiplicaciones de la derecha son multiplicaciones normales de números complejos.

Bases

Se llama base de un espacio vectorial a un sistema generador de dicho espacio, que sea a la vez linealmente independiente. Que sea un sistema generador quiere decir que se pueden expresar todos los vectores del espacio vectorial como una combinación lineal de los vectores que forman la base.

Que los vectores de la base sean linealmente independientes quiere decir que ninguno de ellos puede ser escrito como una combinación lineal de los restantes.

Matriz unitaria

Una matriz unitaria es una matriz compleja que multiplicada por su matriz traspuesta conjugada es igual a la matriz identidad [4]. Es decir, se cumple la condición

$$U \cdot U^T = U^T \cdot U = I,$$

donde U es una matriz unitaria y U^T su traspuesta conjugada.

Son importantes en la computación cuántica, ya que las operaciones que se pueden realizar son transformaciones lineales que vienen dadas por matrices unitarias. Cada una de estas matrices es una posible puerta cuántica en un circuito cuántico.

Matrices de Pauli

Las matrices de Pauli, también llamadas matrices de espín, son muy usadas en mecánica cuántica para representar el espín de una partícula, como un electrón, un neutrón o un protón. En la computación cuántica, son las bases de las puertas cuánticas que usaremos para transformar nuestros qubits. Las cuatro matrices de Pauli son las siguientes:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Las puertas X , Y y Z corresponden a rotaciones de los ejes x , y y z de la esfera de Bloch (ver 2.2.3 La computación cuántica), respectivamente. La puerta I se corresponde con la matriz identidad y no cambia el estado del qubit [5].

Producto tensorial

Cuando queramos tratar a dos sistemas físicos como uno solo, deberemos combinar estos sistemas. Por ejemplo, puede que tengamos dos qubits y queramos combinarlos en un mismo sistema. Para lograr estas combinaciones, haremos uso del producto tensorial de vectores de estados.

El producto tensorial de dos vectores $|\phi\rangle$ y $|\psi\rangle$ se define como:

$$(a|\phi\rangle + b|\psi\rangle) \otimes (c|\phi\rangle + d|\psi\rangle) \equiv ac|\phi\rangle|\phi\rangle + ad|\phi\rangle|\psi\rangle + bc|\psi\rangle|\phi\rangle + bd|\psi\rangle|\psi\rangle$$

donde $|\phi\rangle|\psi\rangle$ es el producto tensorial, que a su vez es una abreviación de $|\phi\rangle \otimes |\psi\rangle$.

El vector base para un sistema de 2 qubits se crea realizando el producto tensorial de dos vectores de 1 qubit:

$$\begin{aligned} |00\rangle &= \begin{pmatrix} 1 & \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 & \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |01\rangle &= \begin{pmatrix} 1 & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ |10\rangle &= \begin{pmatrix} 0 & \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 & \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & |11\rangle &= \begin{pmatrix} 0 & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

El resultado obtenido sería un vector combinado a partir de dos estados independientes en espacios diferentes.

Además, si tenemos dos operadores A y B , los cuales actúan cada uno en un qubit, el producto tensorial de ambos $A \otimes B$ actuará en los dos qubits y se expresa como

$$A \otimes B = \begin{pmatrix} A_{00} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{01} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \\ A_{10} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{11} \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \end{pmatrix}$$

donde A_{jk} y B_{lm} son los elementos de las matrices A y B respectivamente.

2.2.2. Introducción a la física detrás de la computación cuántica

En este apartado, se explican los postulados básicos de la mecánica cuántica. Estos postulados actuarán como conector entre los conocimientos físicos del mundo de la cuántica y los conocimientos matemáticos adquiridos hasta ahora.

Según la hipótesis de De Broglie sobre las ondas de materia, *toda la materia presenta características tanto ondulatorias como corpusculares comportándose de uno u otro modo dependiendo del experimento específico* [6]. Este postulado se basó en el conocido *efecto fotoeléctrico*, publicado por Albert Einstein, que sugiere la naturaleza cuántica de la luz.

En el experimento de la doble rendija, realizado por Thomas Young, se demostró que la materia se comporta como una onda. Para ello, en una cámara oscura se introduce una fuente de luz y por un pequeño agujero se deja escapar un haz de luz que es dirigido hasta una pared con dos agujeros de tamaño muy pequeño. Como consecuencia, el haz de luz queda dividido en dos, cada uno por

un agujero. Por último, tras pasar por esos agujeros, al final se encuentra otra pared donde se proyectará el resultado del haz de luz dividido.

La conclusión es que, si la luz se comportara como corpúsculo, entonces veríamos dos grandes acumulaciones en la última pared, debido a la división. Pero el resultado obtenido no fue ese.

Se obtuvo un patrón de interferencia debido a que al comportarse como ondas, tras pasar por los agujeros se generaron dos frentes de onda diferentes, por lo que interfirieron entre sí, dando lugar a una serie de bandas de luz por toda la pared tal y como se muestra en la Figura 2.3 [7].

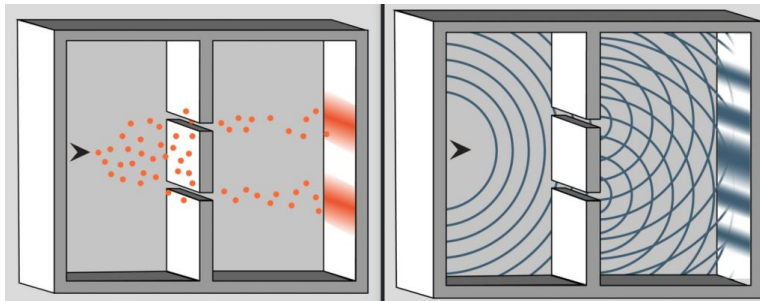


Figura 2.3: Experimento de la doble rendija

Este experimento marcó un antes y un después en el estudio de la mecánica cuántica, ya que se considera fundamental a la hora de demostrar la *dualidad onda corpúsculo*, que es un fenómeno cuántico comprobado en el que muchas partículas pueden comportarse como corpúsculos y ondas a la vez.

El hecho de que las partículas se comporten como ondas querrá decir que estas se definen mediante una función de onda. ¿Quiere decir esto que las partículas son como una especie de *fluctuación*, y que no son *algo* que podamos encontrar en un lugar, en un momento dado?

No exactamente. Debido a este comportamiento ondulatorio, nos da la sensación de que la partícula está en "varios sitios a la vez", pero al intentar medir esta función de onda, nos encontramos con uno de los mayores misterios de la física, que es el *problema de la medida* y el *colapso de la función de onda*.

Al realizar la medición de una partícula, la función de onda de esta colapsará, por lo que esta pasará de comportarse como una onda a una partícula tras la medición. Pero lo curioso de este colapso es que es de naturaleza **no local**, es decir, se produce un cambio repentino y global en la función de onda como sistema. Si se observa una región de la función de onda, esta colapsará por completo, no sólo la región observada, sin importar la distancia a la que estén otras regiones.

Actualmente no se ha encontrado una demostración completa de por qué sucede este colapso, pero es discutido en las diferentes interpretaciones de la

mecánica cuántica, siendo la más aceptada la *Interpretación de Copenhague*.

En la interpretación de Copenhague se dice que **el pasado está determinado pero el futuro es incierto**. En la física clásica se pueden calcular sucesos futuros debido a que nosotros, como individuos, estamos fuera del sistema a calcular, por lo que nuestra presencia y el hecho de que los estamos calculando no afecta al resultado que se va a obtener. Por ejemplo, para calcular a que velocidad irá una pelota cuando toque el suelo si se tira desde un quinto piso. En esta ecuación, lógicamente los observadores no afectamos en el resultado.

En cambio, en la mecánica cuántica, nosotros, como observadores, sí que pertenecemos al sistema que vamos a medir, por lo que por el simple hecho de existir y estar midiendo, afectaremos al resultado.

El famoso problema de la medida hace referencia a esto mismo que se comenta en la interpretación de Copenhague. Se refiere a que el proceso de medición de un sistema cuántico altera la evolución de este, por lo que sólo conoceremos las propiedades del sistema en el momento de la medición.

En 1935, Albert Einstein, Boris Podolsky y Nathan Rosen propusieron un experimento conocido como **la paradoja EPR** [8]. En este experimento, lo que intentaban demostrar era que la teoría de la mecánica cuántica era incompleta.

Introdujeron el término de **entrelazamiento cuántico**. El entrelazamiento es un fenómeno cuántico en el que dos o más partículas comparten el mismo estado cuántico aunque estas estén separadas a grandes distancias. Esto lleva a que, cuando midamos el sistema, obtendremos una correlación entre las propiedades de las partículas. Por ejemplo, si tenemos dos partículas entrelazadas, una en España y otra en Francia, medimos la partícula española y observamos que está girada hacia arriba, la partícula francesa se mostrará girada hacia abajo de forma instantánea. No es que una partícula mande una señal a la otra a la velocidad de la luz, si no que sucede en el mismo instante, como si se hubiera *teletransportado*.

El experimento planteado por EPR consistía en que tenemos dos partículas en entrelazamiento cuántico. Dos observadores reciben cada una de las partículas y uno de ellos mide la posición de una, sabiendo de forma instantánea la posición de la otra. Esta paradoja está en contradicción con la teoría de la relatividad, en la que se dice que nada puede viajar más rápido que la velocidad de la luz. Einstein afirmaba que esta teoría era incorrecta y que debían existir *variables ocultas* que explicaran estos sucesos.

El científico John S. Bell presentó el llamado teorema de Bell dónde se cuantificaban matemáticamente las implicaciones planteadas en la paradoja EPR. Este teorema demuestra que las partículas se comportan como predecía la mecánica cuántica y que se cumple la no localidad, por lo que el entrelazamiento de partículas es posible y existe. Por lo tanto, también se demuestra que la teoría de las variables ocultas locales era incorrecta.

2.2.2.1. Estado cuántico

El estado de un sistema cuántico, como ya se explicó anteriormente, es representado por un vector $|\psi\rangle$ en un espacio vectorial complejo.

El vector de estado $|\psi\rangle$ contiene toda la información sobre un sistema físico en un momento dado.

Este vector nos permite predecir valores posibles a la hora de medir el sistema y asignarles probabilidades.

Dependiendo de la base que escojamos, los estados representarán diferentes observaciones físicas. Por ejemplo, si observamos el estado de un qubit con respecto a los estados base $|0\rangle$ y $|1\rangle$, podemos representar el qubit como $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, donde α y β son las amplitudes de probabilidad correspondientes a los estados $|0\rangle$ y $|1\rangle$.

2.2.3. Teoría de la computación cuántica

Tras haber explicado las bases físicas y matemáticas de la computación cuántica, ahora se podrá entrar en detalle acerca de qué es la computación cuántica y cómo se consigue.

Como ya se ha explicado anteriormente (ver 2.3.1 Introducción), en la computación cuántica la información se almacena en los **qubits**. Los estados base de los qubits se expresan con la notación de Dirac: $|0\rangle$ y $|1\rangle$, que son los equivalentes a los estados 0 y 1 de un bit en computación clásica. En forma vectorial tendríamos:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Un qubit, antes de medirlo, se escribirá como una combinación lineal de los vectores base, es decir, tendrá la forma $a_0|0\rangle + a_1|1\rangle$. Si medimos el qubit, su estado automáticamente se colapsará a cualquiera de los dos posibles estados $|0\rangle$ o $|1\rangle$. La probabilidad de que al medir obtengamos $|0\rangle$ es $|a_0|^2$; la probabilidad de que al medir obtengamos $|1\rangle$ es $|a_1|^2$. Por lo tanto, el resultado final tras la medición es el mismo que en un sistema clásico: al final obtenemos un valor de 0 o 1.

Los valores a_0 y a_1 es lo que se denomina *amplitudes*. El cuadrado del módulo de la amplitud es la probabilidad que hay de que el qubit colapse al valor correspondiente del vector. Este valor, al ser una probabilidad, siempre será positivo. En cambio, la amplitud no tiene por qué ser positiva. Esta es la base de los fenómenos de interferencia.

Esfera de Bloch

Para entender la naturaleza del qubit de una forma más visual se utiliza la esfera de Bloch.

En computación cuántica, la esfera de Bloch es una representación geométrica de un qubit. Se trata de una esfera en la cual cada punto de la superficie corresponde a un posible estado del qubit. Cada par de puntos diametralmente

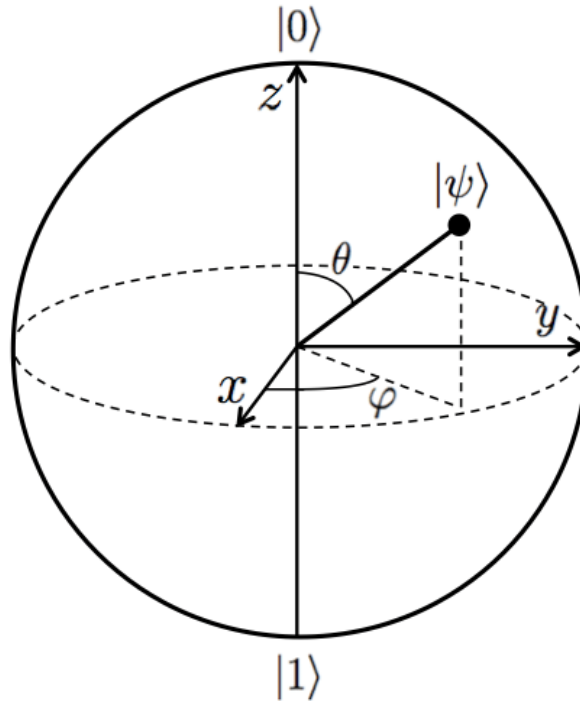


Figura 2.4: Esfera de Bloch

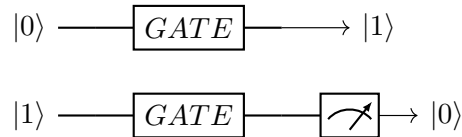
opuestos sobre la esfera corresponde a dos estados orto-normales en el espacio de Hilbert. Siendo orto-normales cuando el producto escalar entre ellos es igual a 0 y el módulo de ambos es igual a 1.

En esta esfera, el polo norte representa el estado $|0\rangle$ y el polo sur representa el estado $|1\rangle$. Una flecha que surge desde el centro de la esfera apunta a la superficie del qubit, representando así el estado de este. Cuanto más se acerque al polo norte, más probabilidades tendrá el qubit de colapsar al 0, mientras que más se acerque al polo sur, más probabilidades habrá de que colapse al 1. En la Figura 2.4 podemos ver la esfera de Bloch [9].

Circuitos cuánticos

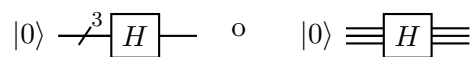
Un circuito cuántico, similar a los circuitos clásicos, es un modelo para la computación cuántica en el que aplicamos una secuencia de puertas cuánticas a un conjunto de qubits para crear un algoritmo que resuelva un problema planteado y finalmente medir el resultado obtenido.

Un circuito se puede representar de forma gráfica, de tal manera que el eje horizontal es la línea de tiempo y las líneas horizontales son los qubits del sistema, por lo que cada vez que una puerta se aplique a un qubit o se realice una medición de este, se dibujará en su correspondiente línea en el momento dado. Una representación de un circuito de 2 qubits se ve de la forma siguiente:

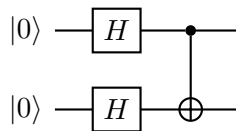


Como se puede ver en el circuito, a la izquierda aparecen los estados iniciales de los qubits de nuestro sistema, y en las líneas de cada uno el símbolo GATE representa la aplicación de una puerta lógica. Por último, el símbolo final del segundo qubit se corresponde con el símbolo de medición.

Una puerta lógica puede tener varias entradas y varias salidas, por lo que se representa como una barra cortando la línea horizontal indicando el número de qubits de entrada o salida, o también con múltiples líneas horizontales de la siguiente forma:



Otro tipo de representación es cuando tenemos una puerta que afecta a varios qubits. Un ejemplo claro es la puerta CNOT, donde se dibuja un punto negro en el qubit que será el de control, mientras que se dibuja el símbolo de la puerta en la línea del qubit objetivo. Se representa de la forma siguiente:



Puertas lógicas cuánticas

Tras haber visto los circuitos cuánticos, ahora veremos qué son las puertas cuánticas y cómo se representan en dichos circuitos.

Una puerta lógica cuántica es una operación que puede ser descrita por una matriz unitaria.

Al igual que en computación clásica, el objetivo es aplicar una colección de puertas simples para formar un circuito.

A diferencia de la mayoría de puertas clásicas, las puertas cuánticas son reversibles y no destruyen información, es decir, conociendo sus salidas podemos saber cuáles fueron las entradas. Empezaremos viendo las puertas más simples, que son aquellas que actúan en un único qubit.

Puerta I o Identidad

Al igual que en computación clásica, en la cuántica también existe una puerta identidad que no cambia el valor del qubit. Al aplicar esta puerta en cualquier parte de nuestro circuito no debería cambiar nada, pero a veces nos puede resultar útil para algún tipo de cálculos o por si queremos crear una operación que no haga nada. Se representa del siguiente modo:

Tabla 2.1: Representación Puerta I

Puerta	En circuito	En Matriz
I		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Puerta X

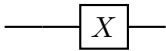
La puerta X es el equivalente cuántico a la puerta NOT. Si la aplicamos a un qubit, se intercambiarán las amplitudes de los componentes $|0\rangle$ y $|1\rangle$ [10]. Actúa de la siguiente forma:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$X: \alpha|0\rangle + \beta|1\rangle \rightarrow \beta|0\rangle + \alpha|1\rangle$$

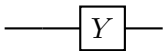
Tabla 2.2: Representación Puerta X

Puerta	En circuito	En Matriz
X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Puerta Y

Esta puerta se caracteriza por girar el qubit al que se aplica 180° sobre el eje y en la esfera de Bloch. Se representa como:

Tabla 2.3: Representación Puerta Y

Puerta	En circuito	En Matriz
Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

Puerta Z

Esta puerta cuando se aplica a un qubit de estado $|0\rangle$ no lo cambia, mientras que cuando se aplica a uno de estado $|1\rangle$ lo transforma en $-|1\rangle$. En general, girará al qubit 180° sobre el eje z en la esfera de Bloch. Se representa como:

Tabla 2.4: Representación Puerta Z

Puerta	En circuito	En Matriz
Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$


Puerta H o de Hadamard

La puerta de Hadamard es muy usada para poner en superposición de ambos estados a un qubit. Actúa en un qubit de estado $|0\rangle$ dejándolo en un estado de superposición de $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. A los qubits que estén en el estado $|1\rangle$ los transforma en $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.

Hay que tener en cuenta que, si se aplica dos veces seguidas, el estado del qubit permanecerá igual que al principio, ya que la matriz de Hadamard al cuadrado es igual a la matriz identidad.

Esta es una de las puertas más usadas y útiles en computación cuántica. En la Figura 2.4 podemos ver que en la esfera de Bloch esta operación representa una rotación de la esfera de 90° en el eje y y 180° en el eje x .

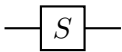
Tabla 2.5: Representación Puerta H

Puerta	En circuito	En Matriz
H		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Puerta S

Esta puerta también es conocida como la puerta de fase, ya que representa un giro de 90° en el eje z de la esfera de Bloch.

Tabla 2.6: Representación Puerta S

Puerta	En circuito	En Matriz
S		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

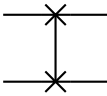
Puertas lógicas cuánticas de dos qubits

Hasta ahora hemos visto las puertas que se aplican a un único qubit, pero para operaciones más complejas necesitaremos puertas que alteren el estado de más de un qubit a la vez. Las más utilizadas y relevantes para este proyecto son la puerta SWAP, la puerta CNOT y la puerta de Toffoli. [10]

Puerta SWAP

Esta puerta sirve para intercambiar el estado de dos qubits. Se representa como:

Tabla 2.7: Representación Puerta SWAP

Puerta	En circuito	En Matriz
<i>SWAP</i>		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

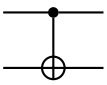
Puerta CNOT

También llamada controlled-NOT o controlled-X, actúa en un par de qubits, donde uno de ellos será el qubit de **control** y el otro será el qubit **objetivo**. Aplicará una operación NOT en el qubit objetivo siempre y cuando el qubit de control esté en el estado $|1\rangle$.

Esta puerta es muy usada, ya que si el qubit de control se encuentra en un estado de superposición y el objetivo está en estado $|0\rangle$ o $|1\rangle$, entonces esta puerta crea **entrelazamiento cuántico** entre ambos qubits.

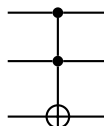
Se representa con un punto negro en la línea del qubit de control y un círculo con una suma en el qubit objetivo.

Tabla 2.8: Representación Puerta CNOT

Puerta	En circuito	En Matriz
<i>CNOT</i>		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Puerta de Toffoli

La puerta de Toffoli también es conocida como la doble controlled-NOT o CCX. Se parece a la puerta CNOT pero en este caso tenemos dos qubits de control y uno de objetivo. Al igual que en CNOT, se aplica una puerta NOT al objetivo únicamente si ambos qubits de control se encuentran en el estado $|1\rangle$. Se representa en el circuito como



Puertas de rotación

Las puertas de rotación $R_x(\theta)$, $R_y(\theta)$ y $R_z(\theta)$ representan las rotaciones en los tres ejes cartesianos de la esfera de Bloch. Se definen por las ecuaciones

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

donde θ es el ángulo de rotación expresado en radianes.

Un operador unitario en un solo qubit se puede escribir de muchas maneras como una combinación de rotaciones junto con cambios de fase globales en el qubit. Esto se expresa como $U = e^{i\alpha}R_n(\theta)$, donde $e^{i\alpha}$ es el cambio global de fase y $R_n(\theta)$ la puerta de rotación sobre el eje n (puede ser el eje X, Y o Z) de un ángulo θ .

Suponiendo que U es una operación unitaria en un qubit, existen números reales α, β, γ y δ tal que $U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$.

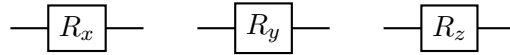
Por lo tanto, con un conjunto de puertas de rotación adecuadas se puede generar cualquier otra puerta de un qubit.

Por ejemplo, podemos obtener las puertas X, Y y Z de la siguiente forma:

$$R_x(\pi) = -iX, R_y(\pi) = -iY, R_z(\pi) = -iZ$$

También, podemos crear la puerta de Hadamard con $R_y(\frac{\pi}{2})Z = H$.

Representación en circuito:



2.2.4. Hardware de los ordenadores cuánticos

Actualmente nos encontramos en lo que se denomina la era NISQ (Noisy Intermediate-Scale Quantum) [11].

Los procesadores cuánticos en esta era contienen un número limitado de qubits (entre 50 y 100), por ello se le da el término de ‘intermediate-scale’ ya que no es un número muy elevado.

Se utiliza también el término ‘noisy’ debido a que estos procesadores son muy sensibles al entorno y no tienen tolerancia a fallos. Las condiciones necesarias para que el entorno no afecte al procesador son bastante específicas, por ejemplo, la temperatura a la que debe estar el hardware debe ser extremadamente baja para evitar que la energía térmica interfiera con los qubits y modifique su estado.

Esta modificación del estado cuántico debido al entorno se conoce como decoherencia cuántica. Resulta en la pérdida de las propiedades cuánticas como sería el entrelazamiento.

Para la creación y ejecución de estos circuitos cuánticos existen diversas opciones que son proporcionadas por las grandes empresas tecnológicas del momento.

Google dispone de su propia librería de Python llamada Cirq que te permite programar para sus ordenadores cuánticos.

IBM, considerado uno de los pioneros en la creación de estos ordenadores, también ofrece su propia librería de Python que se llama Qiskit.

Haciendo uso de estas librerías podremos crear nuestro circuito. Para ejecutarlo podemos hacerlo de varias formas:

En un simulador: Tanto Google como IBM ofrecen un servicio de simuladores online en los cuales podrás ejecutar tus programas. Estos simuladores no se ejecutan sobre ordenadores cuánticos reales.

En un ordenador cuántico real: Estas empresas también disponen de ordenadores cuánticos reales de hasta 127 qubits (aún en fase de investigación) que dejan a nuestra disposición para ejecutar nuestros programas. Existen varias restricciones, por ejemplo, en IBM tan sólo están disponibles para el público aquellos ordenadores de menos de 15 qubits, reservando los más potentes para aquellos que dispongan de una suscripción o sean personal de investigación. Además, cada ordenador tendrá una cola de trabajo, por lo que tendremos que esperar a nuestro turno para poder usarlo.

2.3. Aprendizaje Automático

2.3.1. Introducción

El aprendizaje automático es la ciencia que le ofrece a los sistemas software la habilidad de aprender a partir de un conjunto de datos. Esta ciencia nos ayuda a resolver problemas que son muy complejos desde el punto de vista tradicional o para los que no se ha encontrado un algoritmo capaz de resolver.

Un ejemplo de aprendizaje automático es el famoso filtro de spam utilizado en los servidores de correo. Este filtro aprende recibiendo varios ejemplos de lo que sería un correo spam y gracias a esta información, cuando llegue un nuevo correo, lo clasificará de acuerdo con las características que aprendió de los anteriores correos, marcándolo o no como spam.

Dentro de la inteligencia artificial podemos distinguir diversas ramas como la computación evolutiva, el aprendizaje profundo, el procesamiento del lenguaje natural, entre muchas otras, pero la que nos interesa para este trabajo es el aprendizaje automático o machine learning en inglés.

Dentro del machine learning disponemos de diversas técnicas, que podemos clasificar según si están o no entrenadas bajo la supervisión humana.

- Aprendizaje supervisado: partimos de un conjunto de datos de entrenamiento etiquetado previamente, es decir, incluye la solución deseada.
- Aprendizaje no supervisado: partimos de datos no etiquetados previamente.
- Aprendizaje semisupervisado: partimos normalmente de una pequeña cantidad de datos etiquetados junto a una gran cantidad de datos no etiquetados.
- Aprendizaje por refuerzo: los datos de los que partimos no están etiquetados. Se recompensa al sistema cada vez que toma decisiones acertadas y de esa forma aprende.

Este trabajo se centra en el aprendizaje supervisado, donde disponemos de unos datos de entrenamiento etiquetados.

2.3.2. Modelos de Aprendizaje Automático

Se podrían definir los modelos de aprendizaje automático supervisado como funciones, algoritmos o reglas que definen una relación entre los datos de entrada y las predicciones [10]. Después del entrenamiento, al proporcionar un modelo con una entrada, se recibirá una salida.

Un uso muy común para algoritmos de aprendizaje supervisado es la **clasificación**. Un ejemplo de clasificación sería el sistema de spam que se describió anteriormente (ver 2.3.1 Introducción). El sistema debería aprender a clasificar nuevos emails.

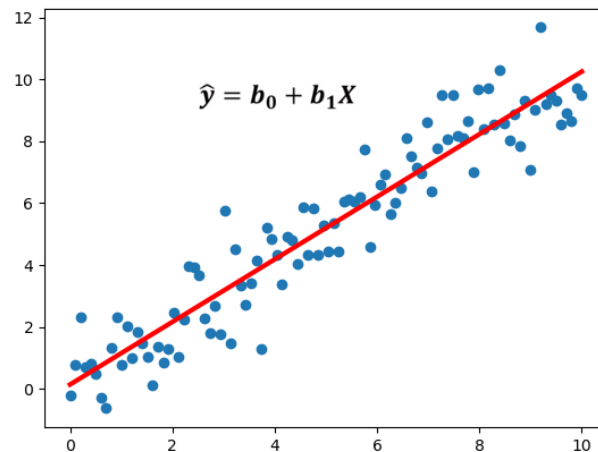


Figura 2.5: Regresión lineal

Otro uso común sería para problemas de **regresión**. En estos problemas debemos predecir un valor numérico, como por ejemplo el precio de un coche dadas una serie de características (marca, año de fabricación, kilometraje, etc.). Para esta clase de problemas, necesitaríamos entrenar el sistema con ejemplos de coches, incluyendo sus características y su atributo objetivo (que en este caso sería el precio). Un ejemplo de regresión lineal se ve representado en la Figura 2.5 [12].

Algunos de los algoritmos de aprendizaje supervisado más importantes son:

- K vecinos más próximos
- Regresión lineal
- Regresión logística
- Árboles de decisión y bosques aleatorios
- Máquinas de vectores de soporte
- Redes neuronales

Estos dos últimos son los que se implementarán en este trabajo.

2.3.3. Cómo construir un sistema inteligente

Para poder construir un modelo de aprendizaje automático cuántico, primero deberemos saber cómo hacerlo en el paradigma clásico.

Siguiendo la estructura planteada en el libro *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow* [2], hay una serie de pasos que se podrían seguir para cualquier problema de machine learning que son **Obtener el conjunto de datos inicial**, **Preparar los datos**, **Seleccionar y entrenar un modelo**, y **Lanzar, monitorizar y mantener el sistema**

2.3.3.1. Obtener el conjunto de datos inicial

Lo primero y más importante sería obtener los datos sobre los que vamos a trabajar. En el caso de que sea un problema de aprendizaje supervisado, sabemos que estos datos deberán estar etiquetados previamente.

Normalmente nuestro conjunto de datos se encontrará en una base de datos, por lo que necesitaremos acceder a esta usando credenciales. Otra opción, si el conjunto es pequeño, sería tener los datos en uno o varios archivos CSV (valores separados por comas).

También, como se verá más adelante, existen diversos módulos de Python (por ejemplo *Scikit-Learn*) para la creación de estos modelos de aprendizaje, que ya contienen conjuntos de datos sencillos. Estos conjuntos, denominados *Conjuntos de datos de juguete*² son útiles para ilustrar el comportamiento de diversos algoritmos de aprendizaje, pero debido a su sencillez, no son representativos de los problemas en un entorno real de machine learning.

No sólo debemos de disponer de los datos, si no que debemos conocer también su estructura, quizás también los atributos más importantes, valores que más se repiten, etc. Para tener una mejor visión general sobre lo que estamos trabajando y por lo tanto orientarnos en la obtención de buenos resultados.

2.3.3.2. Preparar los datos

La mayoría de algoritmos de machine learning necesitan que los datos estén en unas determinadas condiciones para poder ejecutarse sin errores. Por ejemplo, la mayoría no admite valores nulos, por lo que es imprescindible asegurarnos de que cubrimos todos estos valores.

Además, si tuviéramos datos en formato texto en vez de numérico, esto podía generarnos problemas ya que la mayoría de algoritmos prefieren trabajar sólo con números, por lo que la mejor opción sería convertir cada categoría a un número.

Por último, es conveniente transformar nuestros datos para que tengan una escala similar, ya que los algoritmos de machine learning no actúan de forma tan precisa con escalas muy diferentes. Por ejemplo, en el caso de un concesionario, tenemos dos posibles características numéricas que son el nº de puertas y el precio del vehículo. En este caso el rango de puertas sería en torno a 1-7 mientras que el precio tiene un rango mucho mayor que sería sobre 1.000-100.000. Esta enorme diferencia nos afectaría bastante de forma negativa, por lo que lo óptimo sería normalizar los valores, es decir, a cada valor le restamos el valor mínimo y luego lo dividimos entre la resta del valor máximo menos el mínimo. El resultado sería un valor entre el 0 y el 1.

²En el módulo Scikit-learn disponemos de varios ‘Toy datasets’. Se explican en detalle en su página web: https://scikit-learn.org/stable/datasets/toy_dataset.html

2.3.3.3. Seleccionar y entrenar un modelo

Para obtener los mejores resultados en la predicción, elegir un buen modelo con unos determinados parámetros que encaje en nuestros datos es un paso crucial.

Si tenemos un conjunto de datos donde el objetivo es predecir un valor numérico dentro de un rango, entonces se trataría de un problema de regresión en el que escogeríamos modelos como regresión lineal, máquinas de vectores de soporte o árboles de regresión. En cambio, si queremos predecir de qué clase es un dato, teniendo dos posibles opciones, entonces se trataría de un problema de clasificación donde podemos usar modelo como máquinas de vectores de soporte o árboles de decisión.

A la hora de decidir cuál debemos usar, lo mejor sería entrenar varios modelos y comparar los resultados.

El proceso de entrenamiento consiste en proporcionarle al modelo un conjunto de datos de entrenamiento, de los que deberá ‘aprender’.

En problemas de clasificación estos datos de entrenamiento deben contener la respuesta correcta, es decir, la clase a la que pertenece cada dato.

El algoritmo utilizará los datos de entrenamiento para buscar patrones y así poder predecir la clase de futuros datos.

Después de entrenar el modelo con el conjunto de datos de entrenamiento, necesitamos un conjunto de datos de prueba. Estos datos de prueba se introducen en el modelo sin la respuesta correcta, de forma que el modelo deberá predecir sus clases, pero se sabe de antemano las respuestas correctas, por lo que comparando las predicciones con dichas respuestas podemos calcular la exactitud del modelo.

En este proyecto, para el entrenamiento de los modelos, se utiliza un único conjunto de datos que será dividido en dos subconjuntos, uno de entrenamiento y otro de prueba. Por lo general siempre se debe tener más datos de entrenamiento que datos de prueba, por lo que el conjunto total será dividido en 80 % datos de entrenamiento y 20 % datos de prueba.

Tras entrenar los modelos podemos obtener una puntuación sobre el acierto de la predicción, a la cuál llamaremos **accuracy** o **exactitud**. Si obtenemos una baja puntuación, es bastante probable que los datos no proporcionen suficiente información para hacer buenas predicciones, o que el modelo no sea lo suficientemente potente. A esto se le llama **subajuste** o **underfitting**, y quiere decir que el modelo es demasiado simple como para aprender la estructura que siguen nuestros datos. Las posibles soluciones serían:

- Seleccionar un modelo más potente, con más parámetros.
- Mejorar el conjunto de datos.

- Reducir las restricciones del modelo.

Si obtuviésemos una puntuación perfecta en el entrenamiento pero no en el test, lo más probable es que se trate del caso contrario, **sobreajuste** u **overfitting**. Esto quiere decir que el modelo funciona bien en los datos de entrenamiento, pero no generaliza bien. Ocurre cuando el modelo es demasiado complejo en relación con la cantidad de datos. Las posibles soluciones serían:

- Seleccionar un modelo más simple, con menos parámetros o más restringido.
- Entrenar el modelo con más datos.
- Reducir el ruido del conjunto de datos de entrenamiento (ej: arreglando errores en los datos)

2.3.3.4. Lanzar, monitorizar y mantener el sistema

El último paso sería tener nuestro sistema preparado para la producción.

Es recomendable comprobar que el rendimiento es el deseado cada cierto tiempo y alertar en caso de que no sea así, preferiblemente de forma automática con un código de mantenimiento.

Lógicamente, esta comprobación requerirá de la intervención humana, por lo tanto debe estar hecha por expertos en la materia a poder ser posible.

También debemos procurar que los conjuntos de datos nuevos sean de buena calidad antes de lanzar el modelo para evitar malos resultados. Esto es especialmente útil, ya que es muy probable que queramos entrenar nuestro modelo con datos nuevos de forma frecuente, ya que los datos tienden a fluctuar bastante en el tiempo.

Finalmente, debemos automatizar todos los procesos que podamos, para poder renovar o actualizar nuestro modelo cada cierto tiempo sin falta de hacer cambios significativos en el código.

2.4. Aprendizaje Automático Cuántico

2.4.1. Introducción

Después de conocer los fundamentos de la computación cuántica y la inteligencia artificial por separado, es el momento de pensar cómo se podrían fundir estas dos ramas para dar lugar al aprendizaje automático cuántico o quantum machine learning (QML) y todo lo que ello abarca.

Aunque cada vez es mayor la potencia de nuestros ordenadores, el incremento de nuevos datos e información en la red ha sido muchísimo mayor que el rendimiento de los ordenadores necesario para procesar todos estos datos.

La computación cuántica ha demostrado que se pueden crear algoritmos cuánticos capaces de resolver problemas clásicos en un rango de mejora exponencial.

En los últimos años ha habido numerosos avances y algunos de los famosos algoritmos de machine learning ya tienen su versión cuántica, como son la **máquina de vectores de soporte cuántica** y las **redes neuronales cuánticas**, entre otros.

En este trabajo se estudiarán a fondo estos dos algoritmos, dando una explicación teórica de sus fundamentos y finalmente implementándolos junto con una aplicación web dónde podremos visualizar los resultados obtenidos. Ambos serán puestos a prueba en algunos casos sencillos.

2.4.2. QSVM

Las máquinas de vectores de soporte (SVM) son uno de los algoritmos más famosos dentro de la inteligencia artificial, ganando su popularidad en la década de los 90.

Es un modelo muy potente y versátil que es capaz de realizar clasificaciones lineales o no lineales y regresiones.

En este proyecto se implementará una máquina cuántica de vectores de soporte para problemas de clasificación (QSVC) binarios. La variante de QSVM que resuelve problemas de regresión es similar en cuanto a forma de usar, pero tiene una formulación cuántica muy diferente [13].

La idea se basa en que dados un conjunto de datos con dos posibles clases, el algoritmo encuentre un hiperplano que separe las clases con el mayor margen posible. Como se ve en la Figura 2.6 [14], los datos de una clase y otra son separados por el hiperplano.

El conjunto de datos que utilizará el algoritmo se compone de unos datos de entrenamiento $\{(x_1, y_1), \dots, (x_n, y_n)\}$ donde x_n son vectores de números reales y $y_n \in \{-1, 1\}$ son clases binarias a las que pertenece cada dato.

El hiperplano se formula como $w \cdot x + b = 0$, donde w es el vector normal del hiperplano y b es la constante que determina la desviación respecto al origen.

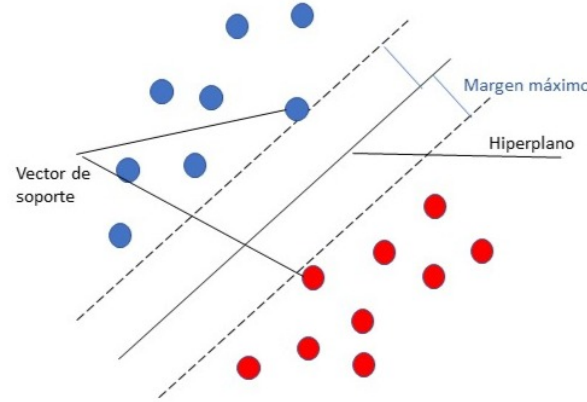


Figura 2.6: SVM

Para encontrar un hiperplano, asumimos que los datos de entrenamiento son linealmente separables. En el caso más sencillo, tendremos un hiperplano que separa los datos de una clase y otra, en el que el margen será $\frac{2}{|w|}$.

Ya que el objetivo es maximizar el margen, entonces debemos optimizar w y b de forma que se minimice

$$\min_w \frac{1}{2} \|w\|^2 \quad \text{sujeto a} \quad y_n(w \cdot x_n + b) \geq 1$$

Por lo tanto, si $w \cdot x_n + b \geq 0$ entonces $y_n = 1$ y el dato se clasificará como clase 1. En caso contrario, si $w \cdot x_n + b < 0$ entonces $y_n = -1$ lo que significa que el dato se clasificará como clase -1.

Como se ve en la Figura 2.6, llamaremos vectores de soporte a los vectores entre los 2 datos, de las 2 clases, más cercanos.

No todos los problemas de clasificación son linealmente separables. Es posible que algún dato de una clase se encuentre mezclado entre los elementos de la otra clase. En estos casos, la forma de calcular los márgenes como se ha visto hasta ahora no permite al algoritmo entrenar con los datos de forma eficaz.

Para tratar con este tipo de casos se usan **márgenes blandos**. Se puede ver una representación en la Figura 2.7 [15].

En los márgenes blandos se introduce un hiperparámetro $C \geq 0$, que sirve para medir cuanto se desvía un dato del margen. También vemos que en la fórmula aparece ξ , que describe ese valor de margen extra que permitirá el modelo.

La fórmula de optimización busca minimizar $\|w\|$ y la desviación del margen descrita por C , de forma que el problema es:

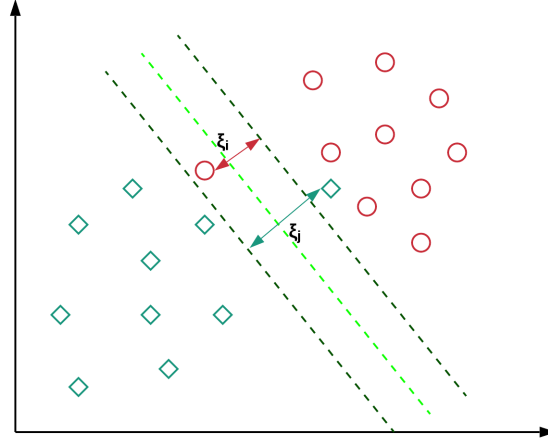


Figura 2.7: SVM. Margen blando.

$$\min \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{sujeto a} \quad y_n(w \cdot x_n + b) \geq 1 - \xi_i \text{ y } \xi_i \geq 0$$

Realizar la optimización de los parámetros puede ser computacionalmente caro, por lo que se suele simplificar transformando el problema de optimización en una **formulación dual**:

$$\begin{aligned} &\text{Maximizar } \sum_i \alpha_i - \frac{1}{2} \sum_{ij} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \\ &\quad \text{sujeto a} \\ &0 \leq \alpha_i \leq C \quad \sum_i \alpha_i y_i = 0 \end{aligned}$$

siendo x_i y x_j ejemplos del conjunto de datos de entrenamiento.

Para clasificar un nuevo dato x en el SVM podemos calcular

$$w \cdot x + b = \sum_i \alpha_i y_i (x_i \cdot x) + b$$

y clasificarlo según el valor sea positivo o negativo.

Ya que vamos a utilizar este modelo en problemas de clasificación, deberemos saber que antes de poder ejecutar nuestro modelo, debemos codificar los datos al espacio de estados cuántico. Es decir, pasarlos a formato cuántico.

En QML, este proceso se llama *encoding* o *codificación* (ver Figura 2.8 [10]).

Existen diversas técnicas para codificar los datos, como por ejemplo mediante amplitudes, pero para hacer más sencillo este proceso recurriremos a una técnica llamada **Quantum Feature Map** representada por $V(\Phi(x))$ [16].

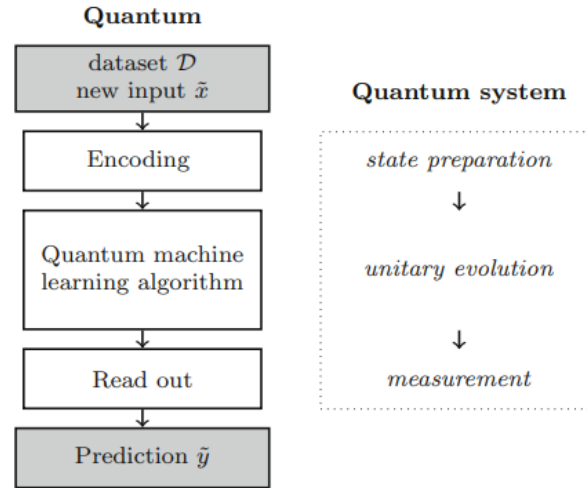


Figura 2.8: Proceso seguido en problemas de QML

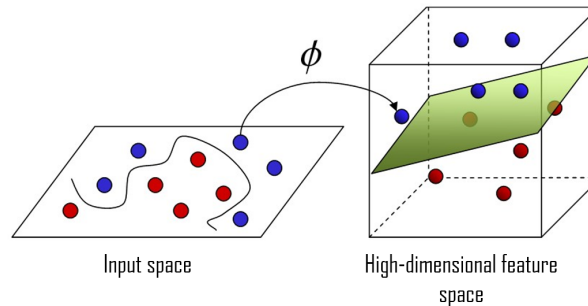


Figura 2.9: SVM. Feature map.

Φ es una función que aplicaremos a los datos, mientras que V es el circuito cuántico de rotación que se encargará de la transformación de los datos.

En el feature map, se trasladan los datos x_i a un espacio de dimensión superior. Por ejemplo si tuviéramos los datos representados en un plano de 2 dimensiones X e Y , los podríamos trasladar a un espacio de 3 dimensiones X , Y y Z . Se puede ver representado en la Figura 2.9 [17].

Este feature map asignará los datos a estados cuánticos y existen diversas formas de codificar los datos en un feature map. Un método bastante conocido, y que nos interesa en este proyecto, es la codificación mediante ángulos de rotación.

En la codificación mediante ángulos de rotación, se utilizan puertas cuánticas de rotación (ver 2.2.3 Puertas lógicas cuánticas). Codifican el dato mediante una rotación en el eje X , Y o Z con un ángulo x . En el prototipo creado se usará la rotación en el eje Y , por lo que para un qubit en un estado inicial $|0\rangle$, tras la

rotación el estado cuántico será $R_y(x)|0\rangle$ [18].

Usando la idea de tener una feature map $\phi(x)$ que mapea los datos a una dimensión mayor, podemos redefinir el problema de optimización para resolver problemas de clasificación **no lineales**. Lo único que hacemos es reemplazar todas las instancia del vector x por $\phi(x)$. De forma que el problema es:

$$\begin{aligned} &\text{Maximizar } \sum_i \alpha_i - \frac{1}{2} \sum_{ij} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j)) \\ &\quad \text{sujeto a} \\ &0 \leq \alpha_i \leq C \quad \sum_i \alpha_i y_i = 0 \end{aligned}$$

Por lo tanto, para clasificar un punto x necesitaremos calcular:

$$w \cdot x + b = \sum_i \alpha_i y_i (\phi(x_i) \cdot \phi(x_j)) + b$$

El producto escalar de los feature maps que se calcula es lo que llamamos **función kernel**.

Los kernels definen el producto escalar de los vectores de entrada, por lo que en casos donde la solución de un problema no sea posible de forma lineal, el kernel aporta más dimensiones al hiperplano para poder resolver el problema.

El cálculo de la función kernel es realizado por el ordenador cuántico. Una de las ventajas del QSVM frente al SVM clásico es precisamente el cálculo de funciones kernel, ya que se puede dar que en ordenadores clásicos dichas funciones sean complejas computacionalmente.

Los datos son trasladados a un espacio vectorial de mayor dimensión mediante un circuito variacional cuántico $U_\phi(x_i)$, donde x_i será el dato que queremos trasladar a $|\phi(x_i)\rangle$, que es una representación de un estado cuántico en el espacio vectorial.

Por lo tanto $U_\phi(x_i)|0\rangle = |\phi(x_i)\rangle$

Finalmente, el entrenamiento se realiza en el ordenador clásico. Para este entrenamiento se utilizan los cálculos del kernel.

2.4.3. Redes neuronales cuánticas

Las redes neuronales artificiales clásicas fueron inspiradas por la arquitectura de los cerebros humanos. Científicos se inspiraron en el funcionamiento de los cerebros para crear una máquina que fuera capaz de pensar, de ser **inteligente**.

Las redes neuronales artificiales (Artificial Neural Networks en inglés) son muy potentes, versátiles y escalables, siendo así el modelo ideal para problemas complejos o extremadamente grandes, como clasificar millones de imágenes, entablar conversaciones mediante reconocimiento del habla o recomendar posibles intereses según tus gustos entre millones de opciones [2].

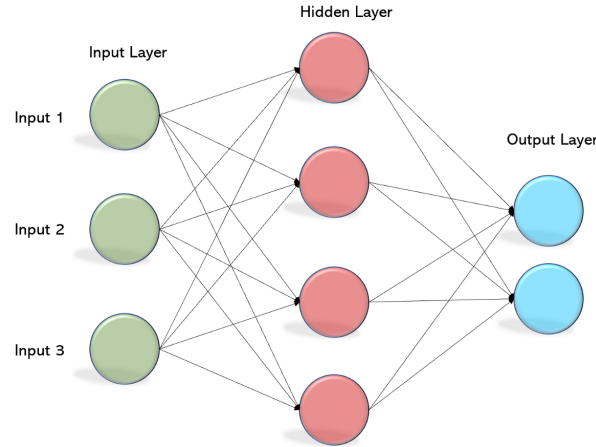


Figura 2.10: Perceptrón Multi-Capa

El **perceptrón** es la red neuronal más básica que existe. Consiste en una única neurona, la cual puede ser usada en problemas simples de clasificación lineal binaria. Calcula una combinación lineal de las entradas de la neurona y, si el resultado excede un límite concreto, entonces la predicción será la clase positiva; en caso contrario, será la clase negativa [2].

La función de este modelo viene definida por

$$f(x; w) = \phi(w^T x)$$

donde la x será el dato de entrada y w el peso. Por lo tanto, para una clasificación binaria, se clasifica el dato en la clase 1 o 0 según el valor de la función, de forma que

$$f(x; w) = \begin{cases} 1 & \text{si } w^T x + b \geq 0 \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Debido a la simplicidad del perceptrón, este no nos permitirá resolver problemas complejos. Pero, en cambio, si juntásemos una gran cantidad de perceptrones divididos en diferentes capas, el resultado sería una red neuronal conocida como **perceptrón multicapa**.

Un perceptrón multicapa se compone de una primera capa de elementos de entrada, una o más capas de **capas ocultas**, y una capa final de salidas. Ver **Figura 2.10** [19].

En una red neuronal cuántica (o a veces también llamada circuito variacional cuántico), a diferencia del QSVM, disponemos de más de un bloque variacional. En este caso, tenemos una primera parte del circuito que se corresponderá con el **feature map** encargado de trasladar los datos del conjunto a estados cuánticos.

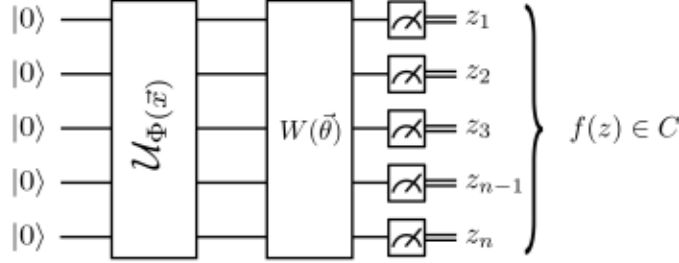


Figura 2.11: Arquitectura de una QNN

Estos estados tendrán la forma $|\phi(x)\rangle$, donde x son los datos de entrada. Se muestra en la Figura 2.11, en la que el primer elemento del circuito sería el feature map.

Una vez estén mapeados los datos, estos pasarán a otro circuito variacional ($W(\theta)$) en el cual se realizará la optimización de los parámetros (θ). Se puede ver representado en la Figura 2.11.

Una **función de pérdida** o loss function, es una función que evalúa la desviación entre las predicciones del modelo y los valores reales de los datos.

Se calcula la función de pérdida comparando la clasificación obtenida con la etiqueta real de los datos. El objetivo de las redes neuronales, para que estas sean más eficaces, es minimizar el resultado de la función de pérdida. Para ello se realizará la optimización los parámetros que se comentaba antes.

Existen diversos algoritmos de optimización, pero para las redes neuronales el más usado es el **descenso del gradiente**. Para realizar el descenso del gradiente, se requiere calcular el gradiente respecto a los parámetros.

El gradiente en una capa en concreto estará determinado por la función de pérdida con respecto a los pesos y ejecuciones en las capas anteriores. Si se asume que el función de pérdida es C y que los parámetros del circuito cuántico son el vector θ , en muchos casos se puede calcular el gradiente como

$$\nabla_{\theta}^C(\theta) = \frac{[C(\theta+s) - C(\theta-s)]}{2s}.$$

De esta forma, se evalúa el coste del circuito cuántico con dos valores de los parámetros diferentes ($(\theta + s)$ y $(\theta - s)$) y luego se normaliza la diferencia para obtener el gradiente.

Este cálculo de la menor función de pérdida con respecto a los pesos se lleva a cabo mediante el ajuste de los distintos pesos de la red neuronal. Este ajuste se hace mediante pasos cortos, usando un hiperparámetro denominado tasa de aprendizaje (learning rate).

A la hora de entrenar el modelo mediante el descenso del gradiente, en la primera ejecución se inicializan los ángulos de rotación con un valor aleatorio.

Al introducir los datos de entrenamiento en el modelo, el algoritmo se encargará de encontrar patrones entre los atributos de los datos y las clases a las que pertenecen, de esta forma pueden aplicar dichos patrones a futuros datos para predecir sus clases.

Estos datos que usaremos para predecir sus clases serán los llamados datos de test.

Una vez realizada la predicción, el algoritmo podrá calcular la función de pérdida comparando los valores obtenidos en la predicción con los valores reales de los datos.

Ya que el gradiente está determinado por la función de pérdida, una vez que se calcule esta se podrá obtener el gradiente y actualizar los parámetros θ mediante el descenso del gradiente.

Todo este proceso haciendo uso de todos los datos de entrenamiento constituye un ciclo o **época de entrenamiento**.

En la práctica existen muchos métodos y optimizadores diferentes para entrenar las redes neuronales. Se ha explicado el descenso del gradiente por ser el más común, pero también existe otros muchos como COBYLA, SPSA o SLSQP³.

Este último optimizador es usado en las librerías de Qiskit y en concreto se usa por defecto con las redes neuronales cuánticas específicas para problemas de clasificación [20].

³Para conocer los detalles de este optimizador visitar: <https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SLSQP.html>

Capítulo 3

PLANIFICACIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE PLANIFICACIÓN

PSI

3.1. INICIO DEL PLAN DE SISTEMAS DE INFORMACIÓN

3.1.1. Análisis de la Necesidad del PSI

El tutor responsable del proyecto recomendó la creación de un sistema que implementara dos tipos de algoritmos de aprendizaje automático de tipo supervisado siguiendo el paradigma de la computación cuántica. Esta recomendación se debe a que la alumna había pedido algún tema en relación con la computación cuántica a nivel de inteligencia artificial debido a su interés y curiosidad por estos estudios. Este sistema tiene la finalidad de aportar a la alumna mayores conocimientos acerca de la computación cuántica, además de conocer de primera mano el proceso a seguir para un proyecto de investigación en el ámbito del software.

El software resultante del proyecto incluye, además de la parte cuántica y de aprendizaje automático mencionadas, una aplicación web donde la alumna podrá demostrar también que ha adquirido los conocimientos de arquitectura, diseño, desarrollo e implementación del software propios del Grado de Ingeniería Informática.

3.1.2. Identificación del Alcance del PSI

En este proyecto se implementarán dos algoritmos de machine learning en su versión cuántica. Estos dos algoritmos son la máquina de vectores de soporte y una red neuronal.

Los algoritmos a implementar serán utilizados para resolver una serie de problemas únicamente de tipo supervisado, es decir, donde el sistema aprende mediante unos datos donde ya se especifica el resultado que se desea obtener. Por lo tanto, no se implementará ningún otro tipo de algoritmo, aunque algunos han sido mencionado anteriormente en la sección 2. Aspectos Técnicos.

Para visualizar los resultados de las ejecuciones se creará una sencilla e intuitiva aplicación web. Dicha aplicación sólo tendrá las funcionalidades básicas para la ejecución del programa, entre ellas aparecen al menos:

- Menú principal donde escoger el algoritmo a utilizar
- Menú de ejecución donde se especificará si se desea ejecutar en un entorno local o en un dispositivo de IBM.
- Menú de configuración donde se concretan ciertos aspectos extras de la ejecución.
- Pantalla de salida donde se muestra el resultado de la ejecución.

La aplicación web está pensada para ser ejecutada únicamente en local, por lo que no se desplegará en un servidor.

El sistema deberá poder ejecutarse en un simulador cuántico y ordenadores cuánticos reales.

Por tanto, los objetivos estratégicos a lograr para que el proyecto sea un éxito son:

- Implementar un algoritmo QSVM sobre un dataset escogido por el usuario.
- Implementar un algoritmo QNN sobre un dataset escogido por el usuario.
- Creación de una aplicación web para que el usuario interaccione y ejecute los algoritmos implementados.
- Integración de los algoritmos con la aplicación web.
- Implementar un entorno de ejecución que conecte con IBM Quantum Experience y con simuladores cuánticos de la librería Qiskit.

3.1.3. Determinación de Responsables

- La dirección de la escuela de ingeniería informática será la encargada de la supervisión y aprobación de este proyecto.
- La alumna se encargará de la creación del proyecto al completo, incluyendo desarrollo del software, documentación e investigación.
- El tutor del proyecto será el responsable de validar el trabajo realizado a medida que el proyecto se vaya desarrollando.

3.2. DEFINICIÓN Y ORGANIZACIÓN DEL PSI

3.2.1. Especificación del Ámbito y Alcance

En función de los objetivos estratégicos vistos, el proyecto se divide en las siguientes fases/objetivos generales, con los siguientes objetivos por cada fase:

3.2.1.1. Fase 1: Implementación del QSVM

Se implementará un algoritmo de aprendizaje supervisado basándose en las máquinas de soporte de vectores. Este clase ya viene implementada en la librería Qiskit, por lo que se usará para los problemas planteados.

En este caso, lo queremos usar para problemas de clasificación binaria, donde los datos deban separarse por dos tipos de clase diferente. A este tipo se les llama quantum support vector cassifier o clasificador cuántico de vectores de soporte.

El algoritmo deberá ser capaz de recibir una serie de datos, para posteriormente procesarlos. El algoritmo será entrenado con el conjunto de datos de prueba y finalmente se ejecutará para predecir la clasificación de los datos seleccionados. Además, mostrará unas gráficas para poder visualizar el resultado final, separando los datos en las diferentes clases.

Objetivos de la fase:

- Hacer uso de Qiskit para la implementación.
- Que el algoritmo clasifique de forme correcta los datos proporcionados haciendo uso de una función kernel.
- Que se visualice de forma sencilla e intuitiva los resultados obtenidos.

3.2.1.2. Fase 2: Implementación del QNN

Se implementará un algoritmo de aprendizaje supervisado basándose en redes neuronales. Esta clase ya viene implementada en la librería Qiskit, por lo que se usará para los problemas planteados.

En este caso, lo queremos usar, al igual que en QSVM, para problemas de clasificación binaria.

El algoritmo deberá ser capaz de recibir una serie de datos, para posteriormente procesarlos. El algoritmo será entrenado con el conjunto de datos de prueba y finalmente se ejecutará parapredecir la clasificación de los datos seleccionados. Además, mostrará unas gráficas para poder visualizar el resultado final.

Objetivos de la fase:

- Hacer uso de Qiskit para la implementación.
- Que el algoritmo clasifique de forma correcta los datos proporcionados.
- Que se visualice de forma sencilla e intuitiva los resultados obtenidos.

3.2.1.3. Fase 3: Creación de la aplicación web

Se desarrollará una página web en la que el usuario pueda escoger entre dos opciones: QSVM y QNN. Tras escoger uno de ellos, se le redirigirá a otra ventana donde podrá configurar los parámetros de entrada de este, como el conjunto de datos a usar o el tipo de ejecución. Además, en el caso de que se escoja ejecutar en hardware cuántico, se deberá indicar el token de la cuenta de IBM Q y el dispositivo que se desea usar.

Tras la ejecución, se mostrará en otra pantalla el resultado obtenido de los algoritmos, junto con una serie de gráficos.

La aplicación web deberá tener una interfaz sencilla e intuitiva que sea fácil de usar para cualquier usuario. Además, esta no dispondrá de ningún sistema de gestión de usuarios por lo tanto, no se dispone de ningún formulario de identificación ni de registro de usuarios.

Objetivos de la fase:

- Que se pueda escoger entre usar QSVM o QNN.
- Que se pueda escoger entre una ejecución local en un simulador o en un ordenador cuántico de IBM.
- En el caso de ejecución en un ordenador cuántico, se mostrará un formulario donde el usuario ingresará la configuración deseada del ordenador.
- Mostrar en una pantalla los resultados obtenidos junto con los gráficos.
- Tener una interfaz intuitiva y sencilla.

3.2.1.4. Fase 4: Integración de los algoritmos con la aplicación web

Esta fase se trata de un periodo intermedio donde se integrarán las partes desarrolladas en las dos fases previas.

El resultado a obtener deberá ser una aplicación web responsiva, que sea capaz de ejecutar los algoritmos cuánticos desarrollados, de forma que el front-end esté conectado con el back-end.

Con esta integración, el sistema se podrá ejecutar de forma local en un simulador cuántico.

Objetivos de la fase:

- Que se pueda ejecutar el algoritmo QSVM en local.
- Que se pueda ejecutar el algoritmo QNN en local.
- Que la interfaz de usuario funcione correctamente y sin ningún cambio visual respecto a la fase anterior.
- Que se muestren los resultados obtenidos tras la ejecución.

3.2.1.5. Fase 5: Ejecución en simuladores y ordenadores cuánticos reales

Una vez que tenemos el sistema funcionando en local, el punto final sería ponerlo a prueba en hardware cuántico. Para ello se hará uso de la API IBM Q, a la cual se accederá haciendo uso de un token.

Para realizar la ejecución, primero se comprobará que el token es correcto, y después se escogerá el ordenador deseado (y que esté disponible) para su posterior uso.

Tras la ejecución, se mostrará una pantalla con los resultados obtenidos acompañados de gráficas.

Objetivos de la fase:

- Conectarse a la API de IBM.
- Permitir la ejecución del algoritmo QSVM en un ordenador cuántico de IBM.
- Permitir la ejecución del algoritmo QNN en un ordenador cuántico de IBM.
- Comprobar que el token introducido es correcto.
- Mostrar al usuario una lista de los simuladores disponibles y permitirle escoger uno.
- Que se muestren los resultados obtenidos junto con las correspondientes gráficas.

Capítulo 4

DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA

FASE DE PLANIFICACIÓN

PSI

4.1. Identificación de las Necesidades de Infraestructura Tecnológica

Las necesidades de la infraestructura tecnológica para este sistema son las listadas a continuación (más información que completa esta sección está detallada en 7.1 DEFINICIÓN DEL SISTEMA):

- Los algoritmos escogidos serán implementados usando el lenguaje de programación Python.
- Los algoritmos cuánticos se implementarán haciendo uso de un framework o librería dedicada a la programación cuántica. Este framework deberá estar disponible en el lenguaje escogido, que es Python.
- El sistema podrá ejecutarse tanto en un simulador como en un ordenador cuántico real.
- El sistema necesitará disponer de hardware cuántico para las ejecuciones.
- El backend de la aplicación web también será desarrollado en Python. Es una restricción debido a que las librerías escogidas para el desarrollo de los algoritmos son específicas de Python.
- La batería de problemas a resolver se seleccionará a partir de los datasets proporcionados por las librerías de Qiskit.

Para la parte del desarrollo de los algoritmos, hay diversas librerías disponibles de forma gratuita que son candidatas. Entre ellas se encuentran Qiskit, Cirq y PennyLane.

En cuanto a la parte de desarrollo web, se pueden considerar diferentes frameworks para el front-end, como sería React, Angular o, para simplificar, HTML y CSS. Para el back-end se consideran otros como Django o Flask.

En la Figura 4.1 se pueden ver los componentes principales del sistema y su tecnología principal.



Figura 4.1: Esquema de la infraestructura tecnológica

4.2. Selección de la Arquitectura Tecnológica

Una vez descritas las opciones disponibles para la arquitectura tecnológica, en este apartado se muestran aquellas que han sido seleccionadas. Las motivaciones que llevaron a cabo esta toma de decisiones se explican en el capítulo 5. ESTUDIO DE VIABILIDAD DEL SISTEMA.

- **Qiskit:** como framework principal para el desarrollo cuántico.
- **Flask:** para el desarrollo del back-end y la estructura de la aplicación web.
- **IBM Quantum Experience:** es el proveedor seleccionado que aportará el hardware cuántico a utilizar.
- **HTML, CSS y Javascript:** para la creación de la interfaz de usuario de la aplicación web.

Capítulo 5

ESTUDIO DE VIABILIDAD DEL SISTEMA

FASE DE DESARROLLO

EVS

5.1. ESTUDIO Y VALORACIÓN DE ALTERNATIVAS DE SOLUCIÓN. SELECCIÓN DE ALTERNATIVA FINAL

En esta sección son expuestas las diferentes alternativas propuestas para el desarrollo del prototipo. La solución elegida se encuentra detallada en el capítulo 4. Definición de la arquitectura del sistema.

5.1.1. PennyLane vs Qiskit

Para el desarrollo de la parte cuántica del sistema se estudiaron dos posibles alternativas que eran PennyLane y Qiskit. Ambas son kits de desarrollo de software que nos permiten crear circuitos y algoritmos capaces de ejecutarse en simuladores y ordenadores cuánticos reales.

Se decidió escoger Qiskit debido a:

- La **familiaridad**: Antes de haber realizado la oferta del trabajo de fin de grado, realicé un curso de introducción a la computación cuántica en Qiskit. Gracias a este curso adquirí cierta familiaridad con los componentes de Qiskit y la metodología de trabajo.
- Quería poner en práctica los conocimientos obtenidos en el curso realizado. Por el hecho de que en dicho curso no se vieron directamente algoritmos de machine learning, quería comprobar si sería capaz de crear un sistema de machine learning con lo que ya había aprendido.
- Más documentación: Tras la investigación y la obtención de diversos artículos sobre aprendizaje automático cuántico, pude ver que la mayoría de investigaciones se centraban en el desarrollo de sistemas con Qiskit, en cambio muy pocos de ellos usaban PennyLane.

PennyLane es también buena opción, sobre todo en el campo del aprendizaje automático debido a:

- Buena documentación: La documentación disponible en su página web [21] es bastante clara, completa y fácil de usar.
- Gran cantidad de información acerca de quantum machine learning: Desde su página web se pueden acceder a diferentes tutoriales acerca de cómo desarrollar algoritmos de aprendizaje automático cuántico. También tienen apartados donde explican los conceptos clave para entender esta metodología. Además cuentan con demos y vídeos de ejemplo.
- Es independiente del sistema de ejecución a utilizar: Gracias a la instalación de diferentes plugins, un mismo circuito cuántico puede ser ejecutado en distintos dispositivos, incluso de empresas diferentes, como Amazon Braket, IBM Q, Google Cirq, Rigetti Forest, Microsoft QDK, y ProjectQ [21].

Tanto PennyLane como Qiskit están en constante desarrollo, por lo que es común encontrarse con errores a la hora de usar dichos kits. Hay que tener especial cuidado y revisar en el historial de versiones de su documentación qué aspectos han cambiado de una versión anterior a una nueva, ya que entre versiones hacen cambios importantes y es posible que afecten considerablemente al proyecto.

5.1.2. Google vs IBM

Cirq, al igual que Qiskit, es una librería que sirve para escribir, manipular y optimizar circuitos cuánticos y ejecutarlos tanto en simuladores como en ordenadores cuánticos.

Esta librería pertenece a Google, por lo que no sólo puedes crear circuitos cuánticos, sino que también se tiene acceso a los simuladores y ordenadores de cuánticos de diferentes proveedores, como Microsoft Azure Quantum, IonQ, Pasqal o Rigetti.

Al ser una de las mayores potencias tecnológicas a nivel mundial, su alcance, documentación y publicaciones al respecto son muy abundantes y de gran calidad.

Debido a que el framework de desarrollo escogido fue **Qiskit**, entonces también se decidió usar los dispositivos de IBM Quantum Experience, ya que Qiskit pertenece a IBM.

5.1.3. Java vs C# vs Python

Para el lenguaje de programación a usar, se distinguieron 3 opciones principales: Python, Java y C#.

Cada lenguaje cuenta con diferentes librerías, siendo Python el más usado en computación cuántica. Le sigue C# con librerías como la de Q#, desarrollada por Microsoft. Y por último, Java, con librerías como QJava, que no es muy conocida.

Se decidió usar el lenguaje de programación Python debido a:

- Mayor calidad de las librerías: El resto de lenguajes tienen librerías menos conocidas y de peor calidad que las disponibles en Python, como Qiskit y PennyLane.
- Más documentación: debido a que la computación cuántica está más extendida en este lenguaje, la documentación disponible también es mayor y de mejor calidad.
- Experiencia en Python: en la carrera prácticamente no use Python, pero a nivel laboral sí que tengo experiencia con este lenguaje, además, justamente en el ámbito de la inteligencia artificial, lo que me dio bastante ventaja a la hora de desarrollar el prototipo.

Por otro lado, Java es el lenguaje con el que más familiarizada estoy, ya que es el que más se usa en la carrera con diferencia. Pero prefiero priorizar la cantidad de documentación y investigaciones realizadas antes que la familiaridad del lenguaje.

Capítulo 6

PLANIFICACIÓN Y GESTIÓN DEL TFG

6.1. PLANIFICACIÓN DEL PROYECTO

6.1.1. Identificación de Interesados

En este apartado se presentan los interesados identificados en este proyecto.

- Escuela de Ingeniería Informática de Oviedo:
 - Director de la Escuela
 - Subdirector de la Escuela
 - Tutor del proyecto
 - Estudiantes de la escuela
- Universidad de Oviedo
- Grupos de investigación:
 - Investigación en computación cuántica
 - Investigación en machine learning
- Usuarios finales del sistema
- Equipo de desarrollo:
 - Jefa de proyecto, Desarrolladora del software, Tester, Arquitecta del Software: Alba Aparicio Pérez
- IBM Quantum Qiskit API

6.1.2. OBS y PBS

6.1.2.1. OBS

La Estructura de Desglose Organizacional (**OBS**) de este proyecto se muestra en la Figura 6.1: OBS. Este organigrama representa los roles de los que se encarga la estudiante en este proyecto.

6.1.2.2. PBS

La estructura de descomposición del producto (**PBS**) serán todos aquellos entregables resultado de la realización de este proyecto. Como se verá en la sección 6.1.3, los productos finales se obtienen a partir de las actividades referenciadas en la planificación.

Debido a que el esquema de todos los productos es demasiado grande, se ha decidido dividir por Investigación (ver Figura 6.3), Desarrollo (ver Figura 6.4), Documentación (ver Figura 6.5), y un esquema general donde se resumen los elementos más importantes (ver Figura 6.2).

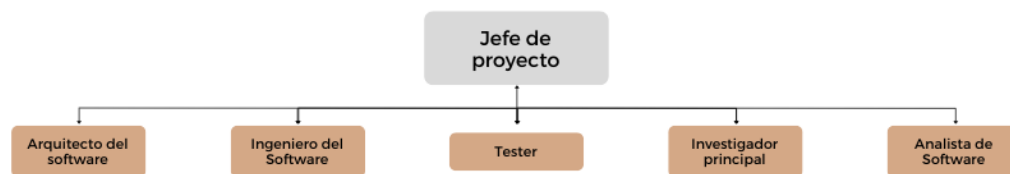


Figura 6.1: OBS

6.1.3. Planificación Inicial. WBS

La estructura de descomposición del trabajo (**WBS**) serán todas aquellas actividades a realizar a lo largo del proyecto.

Debido a que el esquema de todas las actividades es demasiado grande, se ha decidido dividir por Investigación (ver Figura ??), Desarrollo (ver Figura 6.8), Documentación (ver Figura ??), y un esquema general donde se resumen los elementos más importantes (ver Figura 6.6).

También, al inicio del proyecto se establecieron unos hitos divididos de la misma forma que las actividades del WBS. Se pueden ver en la Figuras 6.10, 6.11 y 6.12.

6.1.4. Riesgos

Esta sección contiene todos los riesgos identificados en el proyecto, así como el Plan de Gestión de Riesgos creado para realizar de forma correcta y efectiva la gestión de estos.

6.1.4.1. Plan de Gestión de Riesgos

El Plan de Gestión de Riesgos se encuentra listado en el Anexo I: PLAN DE GESTIÓN DE RIESGOS.

6.1.4.2. Identificación de Riesgos

En esta sección se identifican los riesgos potenciales del proyecto ordenados de mayor a menor impacto. Cada riesgo incluye una breve descripción, así como la estrategia y la respuesta determinada para afrontarlo.

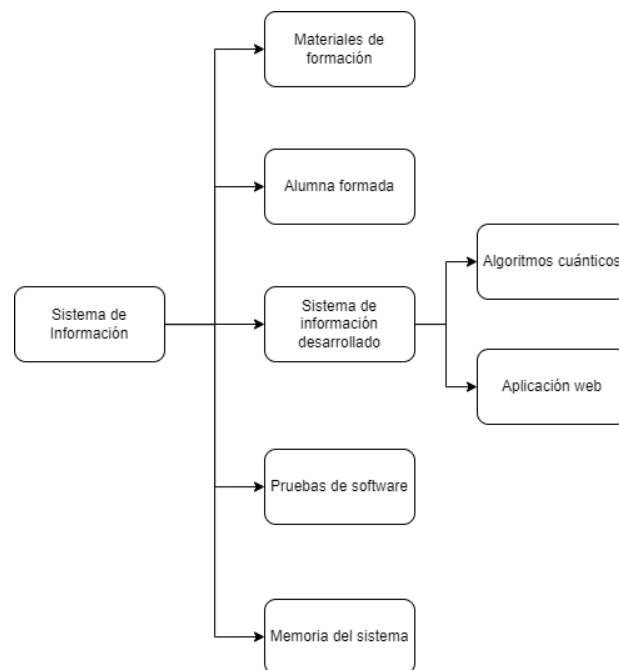


Figura 6.2: PBS General

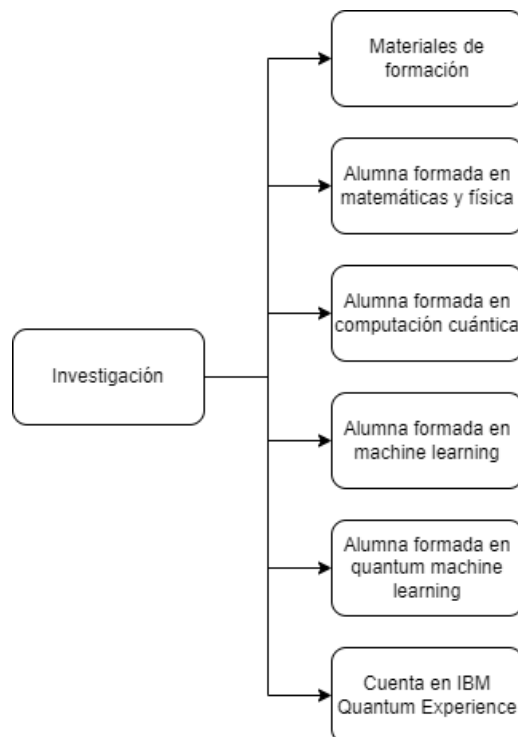


Figura 6.3: PBS Investigación

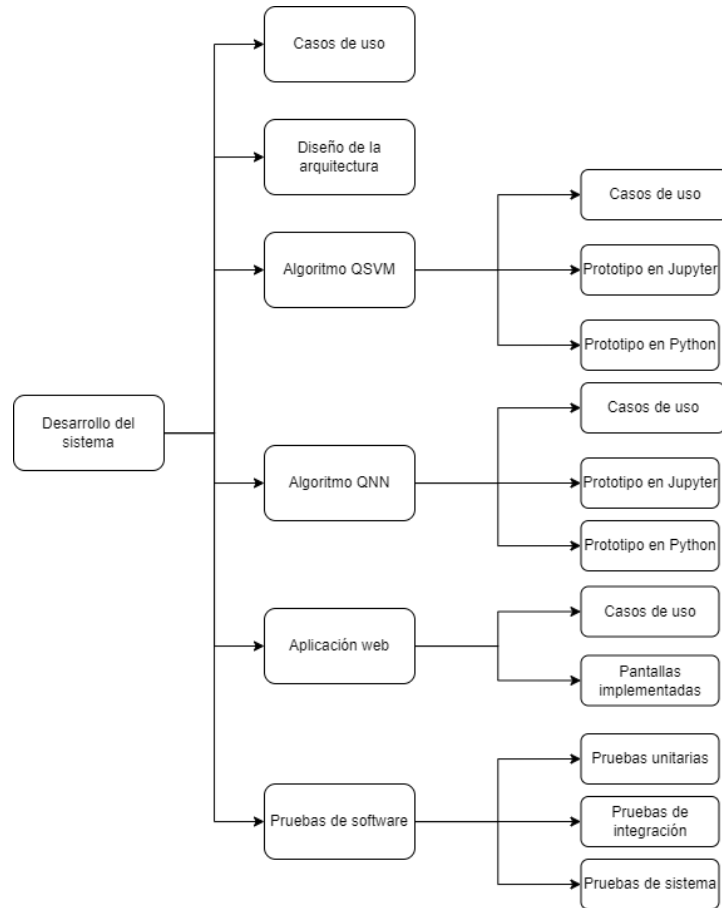


Figura 6.4: PBS Desarrollo

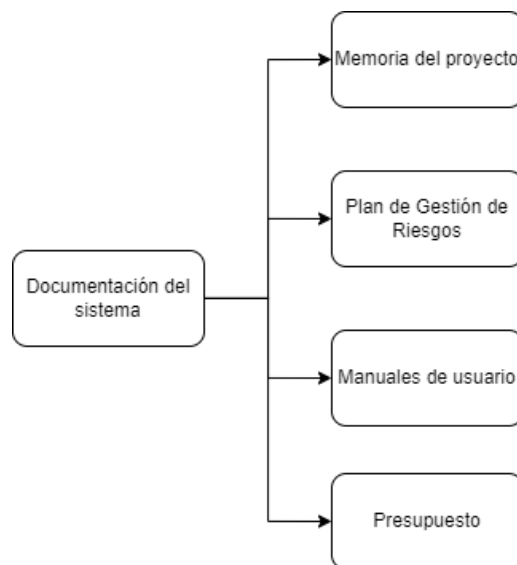


Figura 6.5: PBS Documentacion

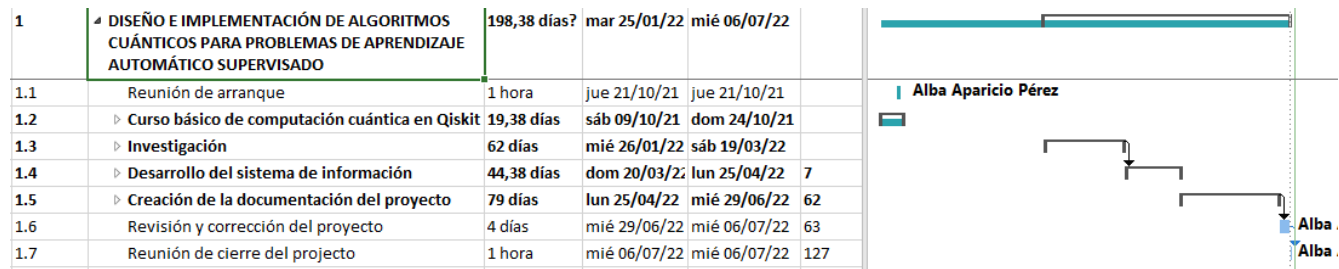


Figura 6.6: WBS. General

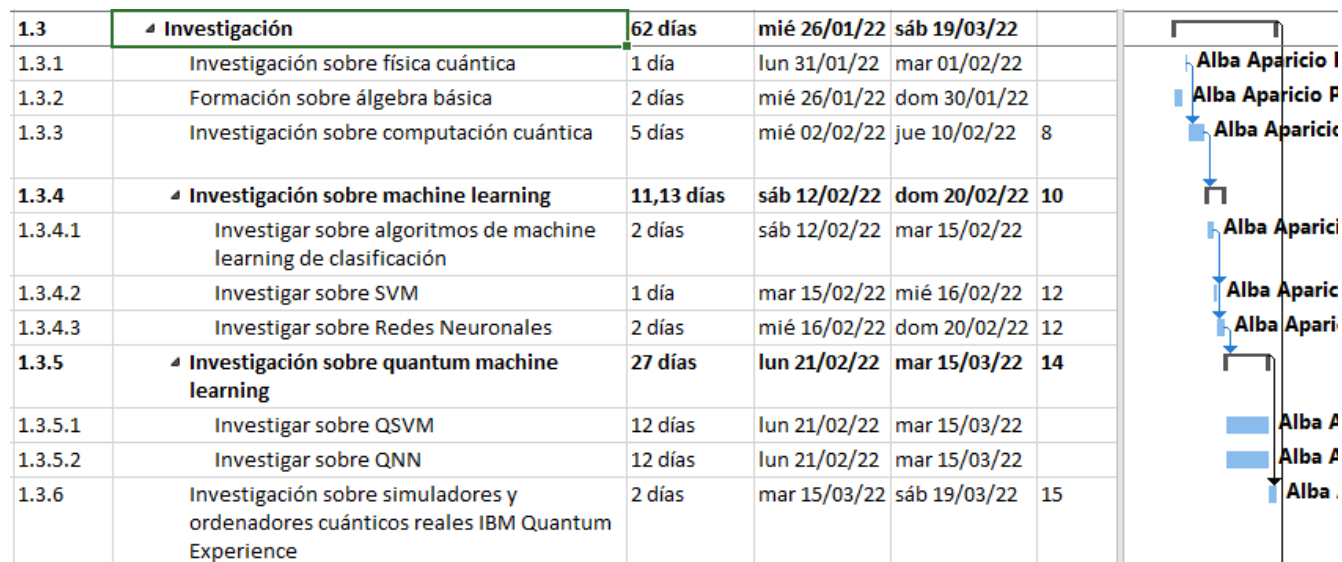


Figura 6.7: WBS. Investigación

1.4	▄ Desarrollo del sistema de información	44,38 días	dom 20/03/22	lun 25/04/22	7	
1.4.1	Análisis e identificación de casos de uso	1 día	dom 20/03/22	lun 21/03/22		
1.4.2	Diseño de la arquitectura	2 días	lun 21/03/22	jue 24/03/22	20	
1.4.3	▄ Implementación de QSVM	8,63 días	jue 24/03/22	jue 31/03/22	21	
1.4.3.1	Requisitos y casos de uso	2 horas	jue 24/03/22	sáb 26/03/22		
1.4.3.2	▸ Implementar prototipo en Jupyter	7,88 días	sáb 26/03/22	jue 31/03/22	23	
1.4.3.3	▸ Implementar prototipo en Python	0,25 días	jue 31/03/22	jue 31/03/22	24	
1.4.4	▸ Implementación de QNN	4,13 días	sáb 02/04/22	lun 04/04/22	22	
1.4.5	▸ Implementación de las Entidades	16 días	mar 05/04/22	lun 18/04/22	31	
1.4.6	▄ Aplicación web	3,13 días	mar 19/04/22	jue 21/04/22	40	
1.4.6.1	Requisitos y casos de uso	2 horas	mar 19/04/22	mar 19/04/22		
1.4.6.2	Formación sobre Flask	2 horas	mar 19/04/22	mar 19/04/22	49	
1.4.6.3	Diseñar la interfaz gráfica	3 horas	mar 19/04/22	mié 20/04/22	50	
1.4.6.4	▸ Implementación de pantallas	1,5 días	mié 20/04/22	jue 21/04/22	51	
1.4.7	Integrar back-end y front-end	5 horas	jue 21/04/22	sáb 23/04/22	48	
1.4.8	Pruebas de integración	5 horas	sáb 23/04/22	dom 24/04/22	60	
1.4.9	Pruebas unitarias	5 horas	dom 24/04/22	lun 25/04/22	61	

Figura 6.8: WBS. Desarrollo

1.5	▄ Creación de la documentación del proyecto	79 días	lun 25/04/22	mié 29/06/22	62	
1.5.1	▸ Introducción	0,25 días	lun 25/04/22	lun 25/04/22		
1.5.2	▸ Aspectos teóricos	4,88 días	lun 25/04/22	sáb 30/04/22		
1.5.3	▸ Planificación del sistema de información	2,25 días	sáb 11/06/22	dom 12/06/22	97	
1.5.4	▸ Definición de la arquitectura tecnológica	0,38 días	mar 26/04/22	mar 26/04/22		
1.5.5	Estudio de viabilidad del sistema	8 horas	jue 28/04/22	sáb 30/04/22	84	
1.5.6	▸ Planificación y gestión del tfg	55,75 días	lun 25/04/22	sáb 11/06/22		
1.5.7	▸ Análisis del sistema de información	10,13 días	mar 26/04/22	mié 04/05/22	21	
1.5.8	▸ Diseño del sistema de información	30,13 días	mié 04/05/22	dom 29/05/22	102	
1.5.9	▸ Construcción del sistema de información	34,13 días	lun 02/05/22	lun 30/05/22	19	
1.5.10	▸ Conclusiones y ampliaciones	0,25 días	mar 28/06/22	mar 28/06/22	115	
1.5.11	Anexos	0,5 días	mar 28/06/22	mié 29/06/22	125	

Figura 6.9: WBS. Documentación

EDT	Nombre de tarea	Duración	Comienzo	Fin
1	▲ Hitos	0 días	vie 01/10/21	vie 01/10/21
1.1	▲ Investigación	0 días	vie 01/10/21	vie 01/10/21
1.1.1	Conocimiento sobre física cuántica	0 días	vie 01/10/21	vie 01/10/21
1.1.2	Conocimiento sobre álgebra básica	0 días	vie 01/10/21	vie 01/10/21
1.1.3	Conocimiento sobre computación cuántica	0 días	vie 01/10/21	vie 01/10/21
1.1.4	Conocimiento sobre machine learning	0 días	vie 01/10/21	vie 01/10/21
1.1.5	Conocimiento sobre quantum machine learning	0 días	vie 01/10/21	vie 01/10/21
1.1.6	Conocimiento sobre simuladores y ordenadores cuánticos de IBM	0 días	vie 01/10/21	vie 01/10/21
1.2	▲ Desarrollo del sistema	0 días	vie 01/10/21	vie 01/10/21
1.2.1	Lista de casos de uso del sistema de información	0 días	vie 01/10/21	vie 01/10/21
1.2.2	Diseño de la arquitectura del sistema	0 días	vie 01/10/21	vie 01/10/21
1.2.3	▲ Algoritmo QSVM	0 días	vie 01/10/21	vie 01/10/21
1.2.3.1	Lista de casos de uso del algoritmo QSVM	0 días	vie 01/10/21	vie 01/10/21
1.2.3.2	Componente desarrollado en Jupyter	0 días	vie 01/10/21	vie 01/10/21
1.2.3.3	Componente desarrollado en Python	0 días	vie 01/10/21	vie 01/10/21
1.2.4	▲ Algoritmo QNN	0 días	vie 01/10/21	vie 01/10/21

Figura 6.10: Hitos del proyecto (1)

1.2.4	▲ Algoritmo QNN	0 días	vie 01/10/21	vie 01/10/21
1.2.4.1	Lista de casos de uso del algoritmo QNN	0 días	vie 01/10/21	vie 01/10/21
1.2.4.2	Componente desarrollado en Jupyter	0 días	vie 01/10/21	vie 01/10/21
1.2.4.3	Componente desarrollado en Python	0 días	vie 01/10/21	vie 01/10/21
1.2.5	▲ Entidades	0 días	vie 01/10/21	vie 01/10/21
1.2.5.1	Lista de casos de uso de las entidades	0 días	vie 01/10/21	vie 01/10/21
1.2.5.2	Componente Dataset	0 días	vie 01/10/21	vie 01/10/21
1.2.5.3	▲ Componente Executor	0 días	vie 01/10/21	vie 01/10/21
1.2.5.3.1	Componente IBMExecutor	0 días	vie 01/10/21	vie 01/10/21
1.2.5.3.2	Componente LocalExecutor	0 días	vie 01/10/21	vie 01/10/21
1.2.5.4	Componente Validator	0 días	vie 01/10/21	vie 01/10/21
1.2.5.5	Componente QMLAlgorithm	0 días	vie 01/10/21	vie 01/10/21
1.2.6	▲ Aplicación web	0 días	vie 01/10/21	vie 01/10/21
1.2.6.1	Lista de casos de uso de la aplicación web	0 días	vie 01/10/21	vie 01/10/21
1.2.6.2	Conocimientos sobre Flask	0 días	vie 01/10/21	vie 01/10/21
1.2.6.3	Diseño de la interfaz gráfica	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4	▲ Pantallas del sistema	0 días	vie 01/10/21	vie 01/10/21

Figura 6.11: Hitos del proyecto (2)

1.2.6.4	⚡ Pantallas del sistema	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.1	Pantalla principal	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.2	Pantalla de documentación	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.3	Pantalla de contacto	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.4	Pantallas de algoritmos	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.5	Pantalla de ejecución	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.6	Pantalla de salida	0 días	vie 01/10/21	vie 01/10/21
1.2.6.4.7	Pantalla de error	0 días	vie 01/10/21	vie 01/10/21
1.2.7	Sistema integrado	0 días	vie 01/10/21	vie 01/10/21
1.2.8	Desarrollo de las pruebas de integración	0 días	vie 01/10/21	vie 01/10/21
1.2.9	Desarrollo de las pruebas unitarias	0 días	vie 01/10/21	vie 01/10/21
1.2.10	Creación de la documentación	0 días	vie 01/10/21	vie 01/10/21
1.2.11	Introducción	0 días	vie 01/10/21	vie 01/10/21
1.2.12	Aspectos teóricos	0 días	vie 01/10/21	vie 01/10/21
1.2.13	Planificación del sistema de información	0 días	vie 01/10/21	vie 01/10/21
1.2.14	Definición de la arquitectura tecnológica	0 días	vie 01/10/21	vie 01/10/21
1.2.15	Estudio de viabilidad del sistema	0 días	vie 01/10/21	vie 01/10/21
1.2.16	Planificación y gestión del TFG	0 días	vie 01/10/21	vie 01/10/21
1.2.17	Análisis del sistema de información	0 días	vie 01/10/21	vie 01/10/21
1.2.18	Diseño del sistema de información	0 días	vie 01/10/21	vie 01/10/21
1.2.19	Construcción del sistema de información	0 días	vie 01/10/21	vie 01/10/21
1.2.20	Conclusiones y ampliaciones	0 días	vie 01/10/21	vie 01/10/21
1.2.21	Anexos	0 días	vie 01/10/21	vie 01/10/21

Figura 6.12: Hitos del proyecto (3)

Disponibilidad de ordenadores cuánticos con un nº elevado de qubits

El objetivo de este proyecto es ejecutar una serie de algoritmos de aprendizaje supervisado tanto en simuladores cuánticos como en ordenadores cuánticos reales. Es un riesgo muy importante el hecho de que a día de hoy no estén disponibles de forma gratuita al público aquellos ordenadores cuánticos con un número elevado de qubits (más de 50), ya que estos ordenadores son los necesarios para garantizar unos resultados satisfactorios en este proyecto.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Muy Alta	Bajo	Bajo	Crítico	Crítico	0,81

- **Estrategia:** Asumir el riesgo
- **Respuesta:** La disponibilidad de estos ordenadores depende una empresa externa, en este caso IBM, por lo que la alumna no puede realizar ninguna acción aparte de asumir el riesgo.

Escasez de tiempo

El alcance de este proyecto se puede considerar bastante generoso, por lo que hay que tener en cuenta a la hora de realizar la planificación las fechas límites de entrega del mismo. Debido a la extensión y complejidad de este es bastante probable que el desarrollo del proyecto conlleve más tiempo del que se dispone.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Gestión de Proyecto	Alta	Alto	Crítico	Bajo	Bajo	0,63

- **Estrategia:** Mitigar el riesgo
- **Respuesta:** Realizando una buena planificación del proyecto y resolviendo la misma batería de problemas en ambos algoritmos de aprendizaje supervisado se espera reducir el tiempo de desarrollo.

Disponibilidad del hardware cuántico

IBM deja a disposición del público una serie de ordenadores cuánticos que tienen una cola de ejecución. Es posible que haya usuarios ejecutando sus circuitos en todos los ordenadores de IBM, por lo no habría ninguno 'libre' para nosotros poder ejecutar los nuestros.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Externo	Media	Inapreciable	Inapreciable	Alto	Crítico	0,45

- **Estrategia:** Asumir el riesgo
- **Respuesta:** La única opción en este caso es esperar a que algún ordenador quede disponible o que llegue nuestro turno en la cola de ejecución.

Errores en las librerías a usar

Debido a que las librerías que se han decidido usar para la implementación de la parte cuántica son nuevas y experimentales, es bastante probable que contengan bugs que alteren el desarrollo esperado del sistema.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Media	Medio	Alto	Bajo	Bajo	0,28

- **Estrategia:** Mitigar el riesgo
- **Respuesta:** Se hará una investigación acerca de las versiones más estables de las librerías y se usará aquella que contenga el número mínimo de bugs detectados para el sistema durante todo el ciclo de vida del proyecto.

Librerías en desarrollo

Las librerías a utilizar están en continuo desarrollo ya que la computación cuántica es un paradigma en pleno auge. Por ello, es posible que se produzcan cambios en esta durante el desarrollo del proyecto.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Muy Alta	Medio	Bajo	Medio	Bajo	0,27

- **Estrategia:** Eliminar el riesgo
- **Respuesta:** Se hará una investigación acerca de las versiones de las librerías y se usará la misma versión durante todo el ciclo de vida del proyecto.

Tecnología con demasiada curva de aprendizaje

Al ser una tecnología tan novedosa existe poca información acerca de esta, lo que complica el aprendizaje de la misma. Es difícil encontrar manuales de funcionamiento, además de que una gran mayoría del existente está desactualizado debido a la rapidez con la que avanza la tecnología.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Alta	Bajo	Medio	Medio	Bajo	0,21

- **Estrategia:** Mitigar el riesgo
- **Respuesta:** Se obtendrá la información de fuentes fiables proporcionadas por el tutor del proyecto y se hará un estudio intensivo del paradigma cuántico.

Largos periodos de espera en la cola de ejecución

A la hora de ejecutar el programa en un ordenador cuántico es posible que este tenga diversos programas en su pila de ejecución, por lo tanto se debe esperar

a que esta se vacíe o llegue nuestro turno. Si hubiera muchos programas o muy complejos, este tiempo de espera se podría demorar.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Externo	Muy Baja	Inapreciable	Inapreciable	Medio	Alto	0,06

- **Estrategia:** Asumir el riesgo
- **Respuesta:** La única opción en este caso es esperar a que algún ordenador quede disponible o que llegue nuestro turno en la cola de ejecución.

6.1.5. Presupuesto Inicial

En esta sección se mostrará el presupuesto de costes y el presupuesto del cliente resumidos.

El proceso seguido y los costes en detalle aparecen en el Anexo II: PRESUPUESTO.

6.1.5.1. Presupuesto de Costes

El presupuesto de coste totales son **49.372,27 €**. Se muestra en la Figura 6.13. Para calcular el presupuesto del cliente se aplica un porcentaje de ponderación mostrada en la Tabla 6.1 con el objetivo de obtener un 25 % de beneficio. Esta ponderación será aplicada a las entradas del presupuesto de costes sin incluir ‘Otros costes’.

Presupuesto de costes		
Cod.	Partida	Total
01	Investigación	12.273,69 €
02	Desarrollo del prototipo	4.276,29 €
03	Documentación	30.832,29 €
04	Otros costes	1.990,00 €
TOTAL COSTES		49.372,27 €

Figura 6.13: Presupuesto de Costes

Beneficio (25 %)	Ponderación
12.343,07 €	0,2605

Tabla 6.1: Ponderación de Costes

6.1.5.2. Presupuesto de Cliente

El presupuesto del cliente total son **59.957,75 €** y se muestra en la Figura 6.14. También, se muestra detallado en la Figura 6.14.

Desglose Presupuesto de cliente				
Partida	Item	Partida	Importe	Total
1		Investigación		15.703,40 €
	01	Investigación sobre física cuántica	396,69 €	
	02	Investigación sobre álgebra básica	793,38 €	
	03	Investigación sobre computación cuántica	1.983,46 €	
	04	Investigación sobre machine learning	2.215,87 €	
	05	Investigación sobre quantum machine learning	9.520,60 €	
	06	Investigación sobre simuladores y ordenadores cuánticos reales IBM Quantum Experience	793,38 €	
2		Desarrollo del prototipo		5.390,26 €
	01	Análisis e identificación de casos de uso	300,05 €	
	02	Diseño de la arquitectura	600,10 €	
	03	Implementación de QSVM	1.145,64 €	
	04	Implementación de QNN	753,14 €	
	05	Implementación de Entidades	1.636,63 €	
	06	Aplicación web	545,54 €	
	07	Integrar back-end y front-end	136,39 €	
	08	Pruebas de integración	136,39 €	
	09	Pruebas unitarias	136,39 €	
3		Documentación		38.864,09 €
	01	¿Qué es este trabajo?	354,60 €	
	02	Aspectos teóricos	20.479,25 €	
	04	Definición de la arquitectura tecnológica	490,99 €	
	05	Estudio de viabilidad del sistema	136,39 €	
	06	Planificación y gestión del tfg	109,11 €	
	07	Análisis del sistema de información	8.810,54 €	
	08	Diseño del sistema de información	2.373,12 €	
	09	Construcción del sistema de información	5.019,01 €	
	10	Conclusiones y ampliaciones	709,21 €	
	11	Anexos	381,88 €	
TOTAL CLIENTE				59.957,75 €

Figura 6.14: Presupuesto de Cliente detallado

6.2. EJECUCIÓN DEL PROYECTO

6.2.1. Plan Seguimiento de Planificación

Para el desarrollo de este proyecto se han creado 3 líneas base principales:

- **Línea base inicial:** Esta línea mostrará el comienzo del desarrollo del proyecto, tras haber realizado las investigaciones pertinentes contempladas en la planificación. Comienza el **17 de Marzo de 2022**
- **Línea base intermedia:** Esta línea marca el punto medio del proyecto, la cual se establece tras el desarrollo de gran parte de la documentación y del sistema. Comienza el **20 de Mayo de 2022**.
- **Línea base final:** Marca el final del desarrollo del proyecto que coincide con la fecha de entrega de este. Comienza el **6 de Julio de 2022**.

6.2.2. Bitácora de Incidencias del Proyecto

A continuación, se muestra un listado con los errores que se encontraron durante el desarrollo del proyecto y que influyeron de cierta forma en la planificación global:

Presupuesto del cliente		
Cod.	Partida	Total
01	Investigación	15.703,40 €
02	Desarrollo del prototipo	5.390,26 €
03	Documentación	38.864,09 €
TOTAL CLIENTE		59.957,75 €

Figura 6.15: Presupuesto de Cliente resumido

- **27/03/2022: Versiones incompatibles de las librerías utilizadas.** Se dio por echo que la mejor versión de la librería qiskit-machine-learning sería la más reciente, pero resultó que dicha versión tenía algunos conjuntos de datos que son usados en el prototipo deprecados, por lo que se usó una versión anterior. Este error se encontró después de haber implementado la nueva versión.
- **23/04/2022: Versiones antiguas de los simuladores.** Se utilizaban unos simuladores Aer antiguos hasta casi al final del desarrollo. Luego se cambiaron por una versión más moderna.
- **23/04/2022: Simuladores incompatibles.** Se creía que todos los simuladores disponibles serían capaces de ejecutar el circuito cuántico creado, pero varios no soportaban las puertas usadas. Se descartaron estos simuladores.
- **30/05/2022: Demasiado larga la ejecución de los tests.** No se esperaba que la ejecución de las pruebas de integración y sistema se fueran a demorar tanto. Debido a que hay varias pruebas que requieren conectarse a un ordenador de IBM, y dicha conexión requiere al menos un par de horas, este proceso demoró significativamente la planificación esperada.

6.2.3. Riesgos

Los riesgos principales que han afectado al proyecto son detallados en esta sección. También están documentados en la sección 6.2.2 Bitácora de Incidencias del Proyecto.

- **Disponibilidad de ordenadores cuánticos con un n^o elevado de qubits:** El ordenador cuántico disponible gratuitamente con mayor capacidad tiene 7 qubits, lo cual no es suficiente para demostrar una mejora en cuanto a eficiencia frente a ordenadores convencionales.
- **Errores en la librería a usar:** Dentro de la librería Qiskit, el módulo qiskit.aqua se ha dejado de actualizar y actualmente se usa el nuevo módulo qiskit.terra, aunque algunas funciones necesarias que existen en aqua no existen en terra, por lo que eso dio lugar a problemas.
- **Librerías en desarrollo:** El módulo qiskit.terra está en desarrollo actualmente y es bastante reciente, por lo que la documentación es escasa y no se

encuentra prácticamente ningún tutorial o información en internet acerca de cómo usar esta nueva versión.

- **Largos periodos de espera en la cola de ejecución:** Como se explicó en la bitácora de incidencias, los periodos de espera a la hora de ejecutar en ordenadores cuánticos fueron extremadamente largos, incluso periodos de más de 7 horas. Este riesgo se tuvo en cuenta, aunque con una probabilidad muy baja. Debido a la fuerte repercusión que tuvo en el proyecto, debería tener uno de los mayores impactos en la lista de riesgos.

6.3. CIERRE DEL PROYECTO

6.3.1. Planificación Final

La planificación, debido a los riesgos reproducidos durante el desarrollo del proyecto, ha sido modificada. La parte modificada aparece en la Figura 6.16.

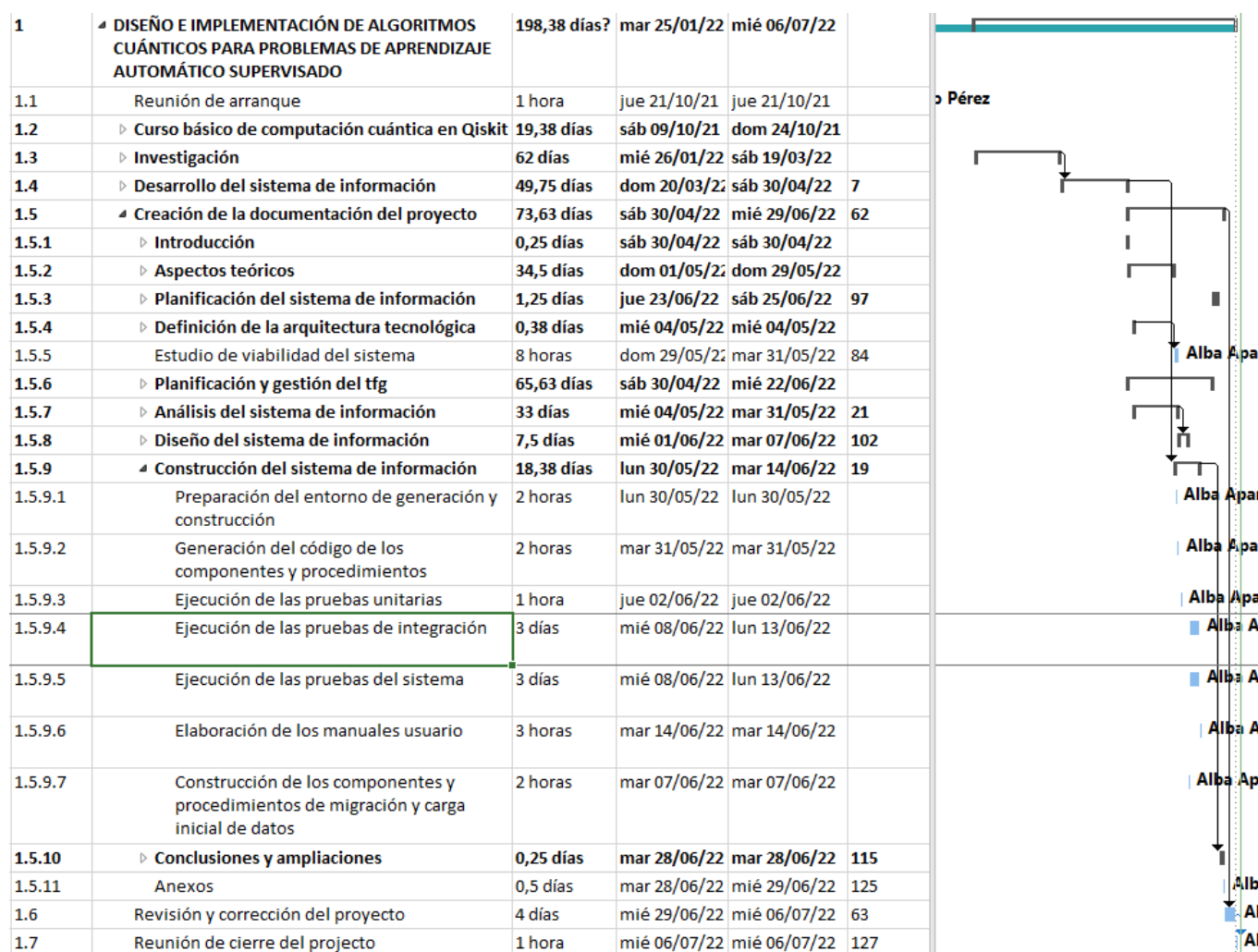


Figura 6.16: Planificación Final

6.3.2. Informe Final de Riesgos

Los riesgos más importantes que han amenazado este proyecto se ven reflejados en la tabla mostrada a continuación. Esta sección se complementa con la sección 6.2.3 Riesgos.

Fecha	Riesgo	Detalles
06/06/2022	Disponibilidad de ordenadores cuánticos con un n ^o elevado de qu-bits	Sólo hay disponibles gratuitamente ordenadores cuánticos con un máximo de 7 qubits.
27/03/2022	Errores en la librería a usar	El módulo qiskit.aqua se ha dejado de actualizar y actualmente se usa el nuevo módulo qiskit.terra, lo que produjo errores en ciertas funciones.
23/04/2022	Librerías en desarrollo	El módulo qiskit.terra está en desarrollo actualmente y es bastante reciente, por lo que hay poca documentación.
30/05/2022	Largos periodos de espera en la cola de ejecución	Las colas de espera se demoraron varias horas cuando lo previsto inicialmente era una espera de menos de media hora.

6.3.3. Presupuesto Final de Costes

El presupuesto de costes final del proyecto es de **58.937,15 €**. Se muestra en la Figura 6.17. Para calcular el presupuesto final del cliente se aplica un porcentaje de ponderación mostrada en la Tabla 6.2

El proceso seguido y los costes en detalle aparecen en el Anexo III: PRESUPUESTO FINAL.

Presupuesto de costes		
Cod.	Partida	Total
01	Investigación	12.273,69 €
02	Desarrollo del prototipo	4.276,29 €
03	Documentación	40.397,17 €
04	Otros costes	1.990,00 €
TOTAL COSTES		58.937,15 €

Figura 6.17: Presupuesto final de Costes

Beneficio (25 %)	Ponderación
14.734,29 €	0,2587

Tabla 6.2: Ponderación final de Costes

6.3.3.1. Presupuesto final de Cliente

El presupuesto final del cliente total son **71.916,63 €** y se muestra en la Figura 6.19. También, se muestra detallado en la Figura 6.18.

Desglose Presupuesto de cliente				
Partida	Item	Partida	Importe	Total
1		Investigación		15.684,53 €
	01	Investigación sobre física cuántica	396,14 €	
	02	Investigación sobre álgebra básica	792,27 €	
	03	Investigación sobre computación cuántica	1.980,68 €	
	04	Investigación sobre machine learning	2.215,87 €	
	05	Investigación sobre quantum machine learning	9.507,28 €	
	06	Investigación sobre simuladores y ordenadores cuánticos reales IBM Quantum Experience	792,27 €	
2		Desarrollo del prototipo		5.382,72 €
	01	Análisis e identificación de casos de uso	299,63 €	
	02	Diseño de la arquitectura	599,26 €	
	03	Implementación de QSVM	1.144,04 €	
	04	Implementación de QNN	752,08 €	
	05	Implementación de Entidades	1.634,34 €	
	06	Aplicación web	544,78 €	
	07	Integrar back-end y front-end	136,20 €	
	08	Pruebas de integración	136,20 €	
	09	Pruebas unitarias	136,20 €	
3		Documentación		50.849,38 €
	01	¿Qué es este trabajo?	354,11 €	
	02	Aspectos teóricos	20.450,60 €	
	04	Definición de la arquitectura tecnológica	490,30 €	
	05	Estudio de viabilidad del sistema	136,20 €	
	06	Planificación y gestión del tfg	108,96 €	
	07	Análisis del sistema de información	8.798,21 €	
	08	Diseño del sistema de información	2.369,80 €	
	09	Construcción del sistema de información	5.011,99 €	
	10	Conclusiones y ampliaciones	12.747,88 €	
	11	Anexos	381,35 €	
TOTAL CLIENTE				71.916,63 €

Figura 6.18: Presupuesto final de cliente detallado

6.3.4. Informe de Lecciones Aprendidas

Las lecciones aprendidas han sido:

- Una buena planificación es extremadamente importante a la hora de realizar un proyecto. Pero más importante aún es seguir la planificación. Puede haber días que el trabajo se haga muy pesado o no se tengan ánimos de hacer nada y eso al final repercute bastante.
- Para evitar una baja productividad hay que planificar el proyecto teniendo en cuenta también periodos de descanso y cargas de trabajo no muy elevadas. Esto no se tuvo en cuenta en este proyecto y al final pasó factura.

Presupuesto del cliente		
Cod.	Partida	Total
01	Investigación	15.684,53 €
02	Desarrollo del prototipo	5.382,72 €
03	Documentación	50.849,38 €
TOTAL CLIENTE		71.916,63 €

Figura 6.19: Presupuesto final de cliente resumido

- Realizar una investigación previa sobre las versiones de las librerías a usar, bugs y otros elementos, sobretodo en proyectos de investigación como es en este caso, es extremadamente importante para ahorrar cambios a mitad de proyecto o errores que puedan surgir.
- Un buen manejo del tiempo es esencial, ya que si lo descuidamos, no sólo perderemos tiempo y dinero, sino también limitaremos la calidad.

Capítulo 7

ANÁLISIS DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

ASI

7.1. DEFINICIÓN DEL SISTEMA

7.1.1. Determinación del Alcance del Sistema

El alcance del sistema ya ha sido explicado anteriormente en el apartado 3.1.2 Identificación del Alcance del PSI.

7.2. ESTABLECIMIENTO DE REQUISITOS

7.2.1. Obtención de los Requisitos del Sistema

7.2.1.1. Requisitos funcionales

Requisitos del QSVM

ReqQSVM1. El sistema permite utilizar el algoritmo QSVM para resolver problemas de clasificación binaria.

ReqQSVM2. El algoritmo QSVM utilizará una función kernel para una óptima clasificación.

ReqQSVM3. El sistema entrenará al algoritmo QSVM utilizando un conjunto de datos de entre los siguientes:

ReqQSVM3.1. El conjunto de datos proporcionado por la librería *qiskit* llamado **ad hoc data**.

ReqQSVM3.2. El conjunto de datos proporcionado por la librería *qiskit* llamado **breast cancer**.

ReqQSVM3.3. El conjunto de datos proporcionado por la librería *qiskit* llamado **gaussian**.

ReqQSVM4. El sistema permitirá al usuario ejecutar el algoritmo SVM en su versión cuántica.

ReqQSVM4.1. El sistema pedirá al usuario la siguiente información:

ReqQSVM4.1.1. Seleccionar un conjunto de datos sobre el que ejecutar el algoritmo. Deberá ser uno de los mencionados en **ReqQSVM3.**

ReqQSVM4.1.2. Tipo de ejecución

ReqQSVM4.1.2.1. En un simulador cuántico (en local)

ReqQSVM4.1.2.1.1. El sistema listará los simuladores cuánticos ofrecidos por *Qiskit*.

ReqQSVM4.1.2.1.2. El sistema permitirá al usuario seleccionar un simulador de los mencionados en **ReqQSVM4.1.2.1.1.**

ReqQSVM4.1.2.2. En hardware cuántico real (en IBM)

ReqQSVM4.1.2.2.1. El sistema pedirá al usuario el TOKEN de autenticación de la API IBM Quantum Experience.

ReqQSVM4.1.3. El nº de ejecuciones del algoritmo.

Requisitos del QNN

ReqQNN1. El sistema permite utilizar el algoritmo QNN para resolver problemas de clasificación binaria.

ReqQNN2. El sistema entrenará al algoritmo QNN utilizando un conjunto de datos de entre los siguientes:

ReqQNN2.1. El conjunto de datos proporcionado por la librería *qiskit* llamado **ad hoc data**.

ReqQNN2.2. El conjunto de datos proporcionado por la librería *qiskit* llamado **breast cancer**.

ReqQNN2.3. El conjunto de datos proporcionado por la librería *qiskit* llamado **gaussian**.

ReqQNN3. El sistema permitirá al usuario ejecutar el algoritmo de red neuronal en su versión cuántica.

ReqQNN3.1. El sistema pedirá al usuario la siguiente información:

ReqQNN3.1.1. Seleccionar un conjunto de datos sobre el que ejecutar el algoritmo. Deberá ser uno de los mencionados en **ReqQNN2..**

ReqQNN3.1.2. Tipo de ejecución

ReqQNN3.1.2.1. En un simulador cuántico (en local)

ReqQNN3.1.2.1.1. El sistema listará los simuladores cuánticos ofrecidos por *Qiskit*.

ReqQNN3.1.2.1.2. El sistema permitirá al usuario seleccionar un simulador de los mencionados en **ReqQNN3.1.2.1.1..**

ReqQNN3.1.2.2. En hardware cuántico real (en IBM)

ReqQNN3.1.2.2.1. El sistema pedirá al usuario el TOKEN de autenticación de la API IBM Quantum Experience.

ReqQNN3.1.3. El n^o de ejecuciones del algoritmo.

Requisitos del sistema cuántico

ReqCuántica1. El sistema será implementado sobre un circuito cuántico que se ejecutará en un simulador o en un ordenador cuántico real.

ReqCuántica2. El sistema permitirá ejecutar los algoritmos mencionados en **ReqQSVM1.** y **ReqQNN1.** proporcionados por la librería *qiskit-machine-learning*.

ReqCuántica3. El sistema codificará las diferentes clases de datos en qubits.

ReqCuántica3.1. El sistema se compondrá de 2 qubits.

ReqCuántica4. El sistema predecirá la clase de un subconjunto de datos de prueba.

ReqCuántica5. El sistema ejecutará el circuito preparado.

ReqCuántica6. El sistema calculará el porcentaje de exactitud alcanzado en la predicción mencionada en **ReqCuántica4..**

Requisitos de la salida esperada

- Salida1.** El sistema mostrará un esquema del circuito ejecutado resumido.
- Salida2.** El sistema mostrará las clases de los elementos del subconjunto de prueba.
- Salida3.** El sistema mostrará la predicción, resultado del algoritmo ejecutado, de las clases de los elementos del subconjunto de prueba.
- Salida4.** El sistema mostrará el porcentaje de exactitud alcanzado sobre el subconjunto de prueba.
- Salida5.** El sistema mostrará los gráficos generados durante la ejecución del algoritmo de aprendizaje ejecutado.

Requisitos de la parametrización del algoritmo

- Parametrización1.** El sistema validará que el conjunto de datos escogido por el usuario sea alguno de los listados en **ReqQSVM3.**, en el caso de ejecutar el algoritmo QSVM o **ReqQNN2.** en el caso de ejecutar QNN.
- Parametrización2.** El sistema validará que el TOKEN mencionado en **ReqQSVM4.1.2.2.1.** exista y sea correcto en el caso de que el usuario seleccione ejecutar el sistema en un ordenador cuántico real.
- Parametrización3.** El sistema validará que el dispositivo seleccionado exista y sea correcto en el caso de que el usuario seleccione ejecutar el sistema en un simulador cuántico.
- Parametrización4.** El sistema validará que el n^o de ejecuciones sea un número natural.
- Parametrización5.** El n^o de ejecuciones por defecto de los algoritmos serán 1024.

Requisitos de la aplicación web

- ReqAppWeb-1.** El usuario accederá al sistema mediante una aplicación web.
 - ReqAppWeb-1.1.** La aplicación web permitirá al usuario introducir la configuración deseada para la ejecución del sistema.
 - ReqAppWeb-1.2.** La aplicación web le comunicará al sistema cuántico los parámetros escogidos por el usuario (**ReqAppWeb-1.1.**).
 - ReqAppWeb-1.3.** La aplicación web avisará al usuario cuando alguno de los parámetros introducidos sea inválido.
 - ReqAppWeb-1.4.** La aplicación web le mostrará al usuario el resultado de la ejecución del sistema cuántico.

7.2.1.2. Requisitos no funcionales

Requisitos de usabilidad

Usabilidad1. La aplicación web tendrá una interfaz intuitiva y sencilla.

Usabilidad2. La aplicación web será responsiva.

Usabilidad3. El sistema no recopilará ningún tipo de información personal de los usuarios.

Requisitos de la API IBMQ

IBMQ1. El sistema hará uso de la API IBMQ para:

IBMQ1.1. Validar que el TOKEN de autenticación del usuario existe y es válido en el sistema de IBM Quantum.

IBMQ1.2. Guardar la sesión en disco.

IBMQ1.3. Cargar la sesión guardada en el disco (si existe).

IBMQ1.4. Obtener acceso a un ordenador cuántico que:

IBMQ1.4.1. Tenga un n^o de qubits mayor o igual que el necesario para ejecutar el sistema.

IBMQ1.4.2. Esté disponible en el momento de realizar la ejecución.

IBMQ1.4.3. Tenga la menor lista de espera posible.

Requisitos de usuario

RNF-1. El usuario tendrá conocimientos básicos sobre algoritmos de aprendizaje automático.

RNF-2. El usuario tendrá conocimientos básicos sobre el resultado esperado tras ejecutar un algoritmo de aprendizaje automático.

RNF-3. El usuario dispondrá de una cuenta personal en IBM Quantum.

7.2.2. Identificación de Actores del Sistema

El sistema contará con 3 actores principales. El usuario, que será el responsable de ejecutar el sistema. El simulador cuántico, que será proporcionado por la librería usada en este proyecto (**Qiskit**). Esta librería cuenta con un proveedor de simuladores llamado **Aer**.

El último actor sería la API de IBM Quantum Experience.

Usuario

El usuario que ejecutará el sistema será el actor principal y más importante. A la hora de ejecutar el sistema, este simplemente accederá mediante la aplicación web sin necesidad de registrarse previamente. Pero sí será necesario que disponga de una cuenta en IBM Quantum, ya que ahí se le proporcionará el token necesario para ejecutar el sistema en un ordenador cuántico real.

Al no disponer de formulario de registro ni identificación, todos los usuarios del sistema tendrán el mismo nivel de privilegios (básico), por lo tanto no existe un sistema de gestión de usuarios.

Simulador Aer

El Simulador Aer es proporcionado por la librería Qiskit. El objetivo de este será ejecutar el algoritmo simulando un ordenador cuántico sin ruido. Los simuladores proporcionados, y usados por este sistemas serán de 32 qubits [22].

El sistema se comunicará con este actor para ejecutar en un simulador cuántico un circuito cuántico.

IBM Quantum Experience

Este actor hace referencia a la API IBM Quantum Experience. El sistema se comunicará con este actor para los casos de uso relacionados con la ejecución de los algoritmos en ordenadores cuánticos reales. Como ya se comentó en el apartado de **Usuario**, es necesario disponer de una cuenta en IBM Quantum para poder comunicarse con este actor.

7.2.3. Especificación de Casos de Uso

En este apartado especificaré los casos de uso principales para este sistema. La Figura 7.1 contiene todos los casos de uso contemplados.

Tabla 7.1: Especificación Caso de Uso - Obtener un ordenador cuántico de IBM

Nombre del caso de uso
Obtener un ordenador cuántico de IBM
Descripción
Un usuario podrá obtener acceso a un ordenador cuántico real, disponible y válido para el circuito que desea ejecutar, a través de la API IBM Quantum.

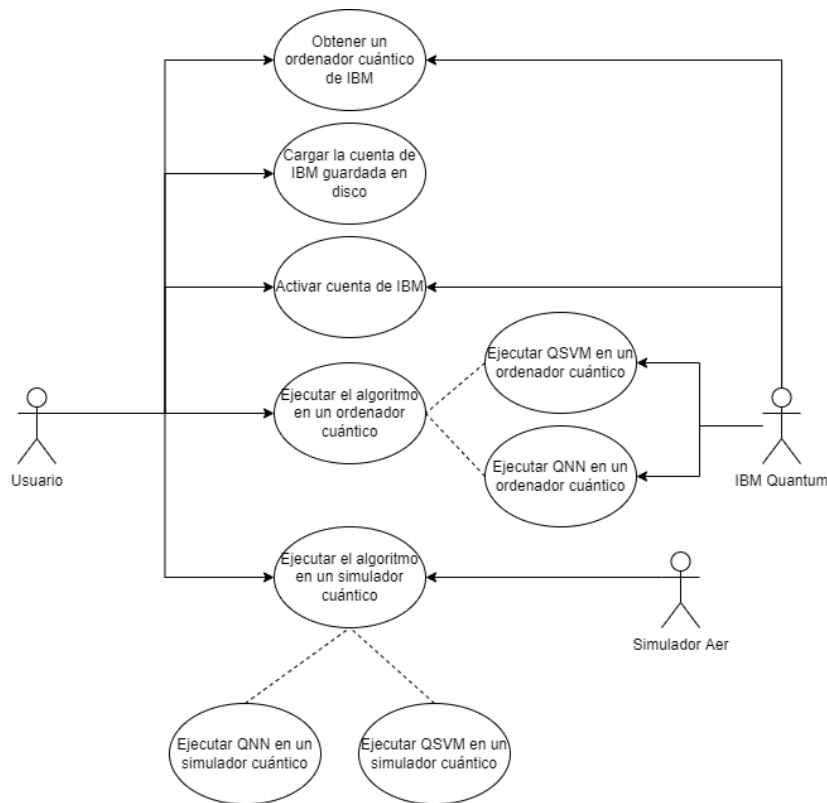


Figura 7.1: Casos de uso

Tabla 7.2: Especificación Caso de Uso - Cargar la cuenta de IBM guardada en disco

Nombre del caso de uso
Cargar la cuenta de IBM guardada en disco
Descripción
Un usuario podrá almacenar y cargar en disco su cuenta de IBM para usos posteriores. Esto le permitirá no tener que introducir más de una vez el token de su cuenta IBM en el sistema.

Tabla 7.3: Especificación Caso de Uso - Activar cuenta de IBM

Nombre del caso de uso
Activar cuenta de IBM
Descripción
Un usuario introducirá su token de IBM Quantum Experience para que la API lo valide y le de acceso a los ordenadores cuánticos.

Tabla 7.4: Especificación Caso de Uso - Ejecutar QSVM en un simulador cuántico

Nombre del caso de uso
Ejecutar QSVM en un simulador cuántico
Descripción
Un usuario podrá enviar a un simulador Aer un circuito cuántico de QSVM para ejecutarlo simulando un ordenador cuántico real.

Tabla 7.5: Especificación Caso de Uso - Ejecutar QSVM en un ordenador cuántico

Nombre del caso de uso
Ejecutar QSVM en un ordenador cuántico
Descripción
Un usuario podrá enviar a la API de IBM un circuito cuántico de QSVM para ejecutarlo en un ordenador cuántico real.

Tabla 7.6: Especificación Caso de Uso - Ejecutar QNN en un simulador cuántico

Nombre del caso de uso
Ejecutar QNN en un simulador cuántico
Descripción
Un usuario podrá enviar a un simulador Aer un circuito cuántico de QNN para ejecutarlo simulando un ordenador cuántico real.

Tabla 7.7: Especificación Caso de Uso - Ejecutar QNN en un ordenador cuántico

Nombre del caso de uso
Ejecutar QNN en un ordenador cuántico
Descripción
Un usuario podrá enviar a la API de IBM un circuito cuántico de QNN para ejecutarlo en un ordenador cuántico real.

7.3. ANÁLISIS DE LOS CASOS DE USO

7.3.1. Caso de Uso - Obtener un ordenador cuántico de IBM

El diagrama de robustez de la Figura 7.2 se recoge el correspondiente al caso de uso Obtener un ordenador cuántico de IBM. La tabla 7.8 recoge el análisis de este caso de uso.

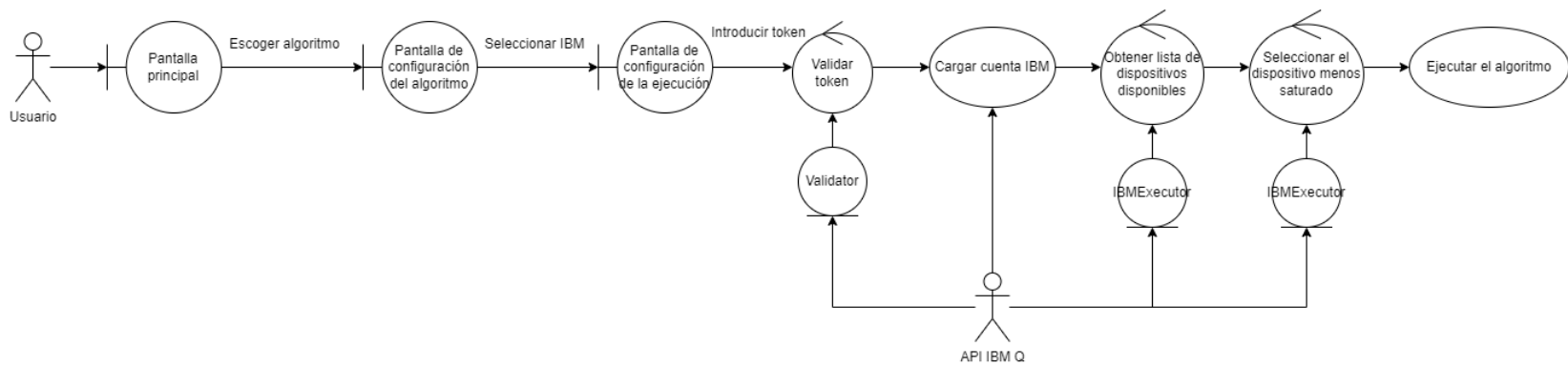


Figura 7.2: Diagrama de robustez - Obtener un ordenador cuántico de IBM

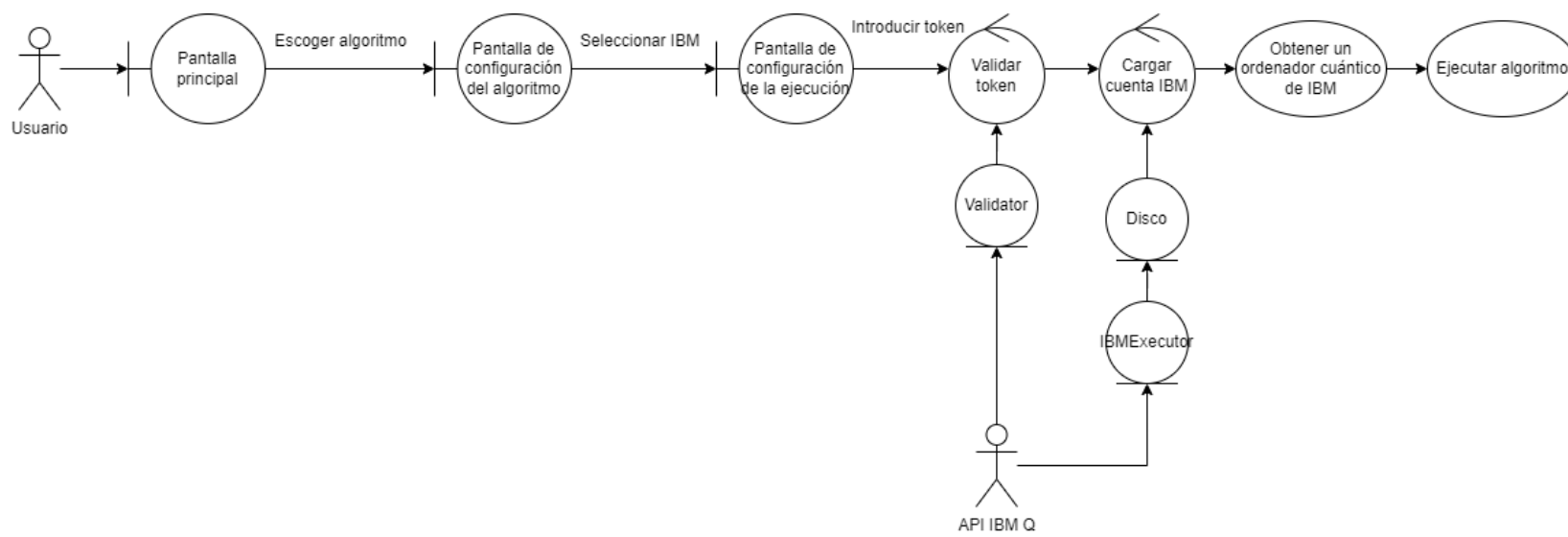


Figura 7.3: Diagrama de robustez - Cargar la cuenta de IBM guardada en disco

Tabla 7.8: Análisis del Caso de Uso - Obtener un ordenador cuántico de IBM

Obtener un ordenador cuántico de IBM	
Precondiciones	El usuario debe tener una cuenta en IBM Quantum Experience. El usuario debe seleccionar ejecutar el sistema en IBM.
Postcondiciones	-
Actores	Usuario y API IBM Quantum
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará uno de los dos algoritmos disponibles (QSVM o QNN). Seleccionará que la ejecución se hará en IBM, y en la siguiente pantalla introducirá el token de su cuenta de IBM Quantum Experience. El sistema buscará y obtendrá un ordenador cuántico disponible y óptimo para el algoritmo a ejecutar.
Escenarios Secundarios	-
Excepciones	Ningún ordenador cuántico, con las características requeridas, está disponible en el momento solicitado. Se notifica el error y se le pide al usuario que lo intente de nuevo más tarde.
Notas	-

7.3.2. Caso de Uso - Cargar la cuenta de IBM guardada en disco

El diagrama de robustez de la Figura 7.3 se recoge el correspondiente al caso de uso Cargar la cuenta de IBM guardada en disco. La tabla 7.9 recoge el análisis de este caso de uso.

Tabla 7.9: Análisis del Caso de Uso - Cargar la cuenta de IBM guardada en disco

Cargar la cuenta de IBM guardada en disco	
Precondiciones	El usuario debe tener una cuenta en IBM Quantum Experience. El usuario debe seleccionar ejecutar el sistema en IBM.
Postcondiciones	-
Actores	Usuario
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará uno de los dos algoritmos disponibles (QSVM o QNN). Seleccionará que la ejecución se hará en IBM, y en la siguiente pantalla introducirá el token de su cuenta de IBM Quantum Experience. El sistema verificará que ya se accedió anteriormente debido a que existe una cuenta guardada en disco, por lo que devuelve esta cuenta y permite que el usuario acceda a los ordenadores cuánticos de IBM.
Escenarios Secundarios	El usuario no introdujo anteriormente el token de IBM: el sistema detecta que no hay ninguna sesión en disco, por lo que creará una y la guardará en el caso de que el token sea válido.
Excepciones	El token introducido es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.3.3. Caso de Uso - Activar cuenta de IBM

El diagrama de robustez de la Figura 7.4 se recoge el correspondiente al caso de uso Activar cuenta de IBM. La tabla 7.10 recoge el análisis de este caso de uso.

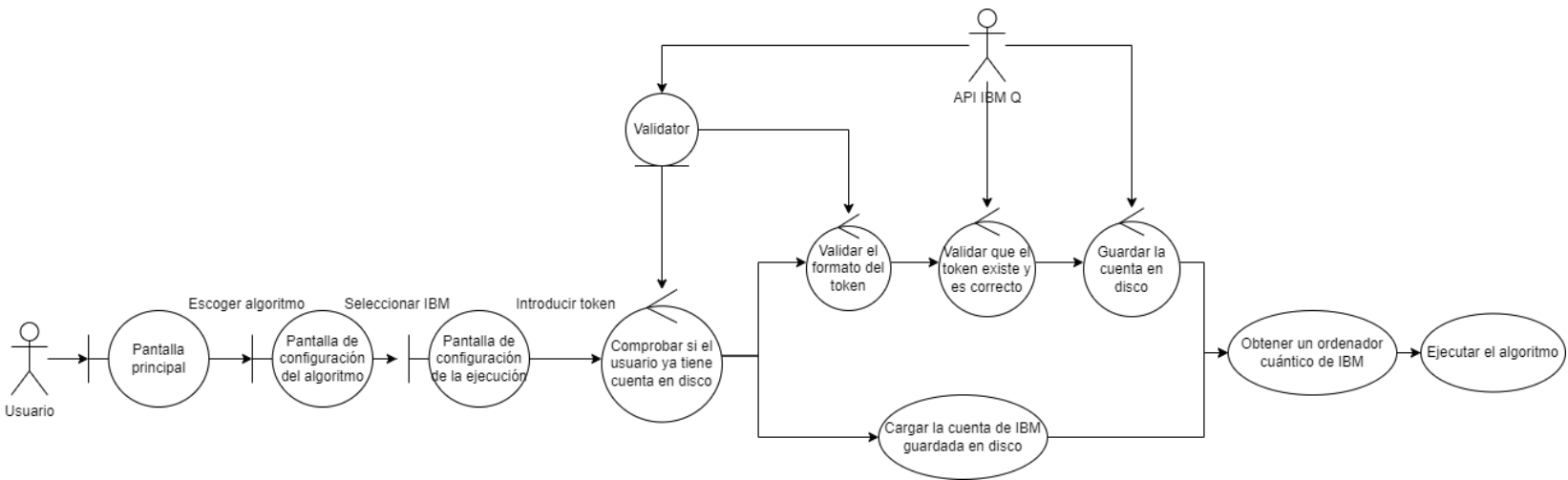


Figura 7.4: Diagrama de robustez - Activar cuenta de IBM

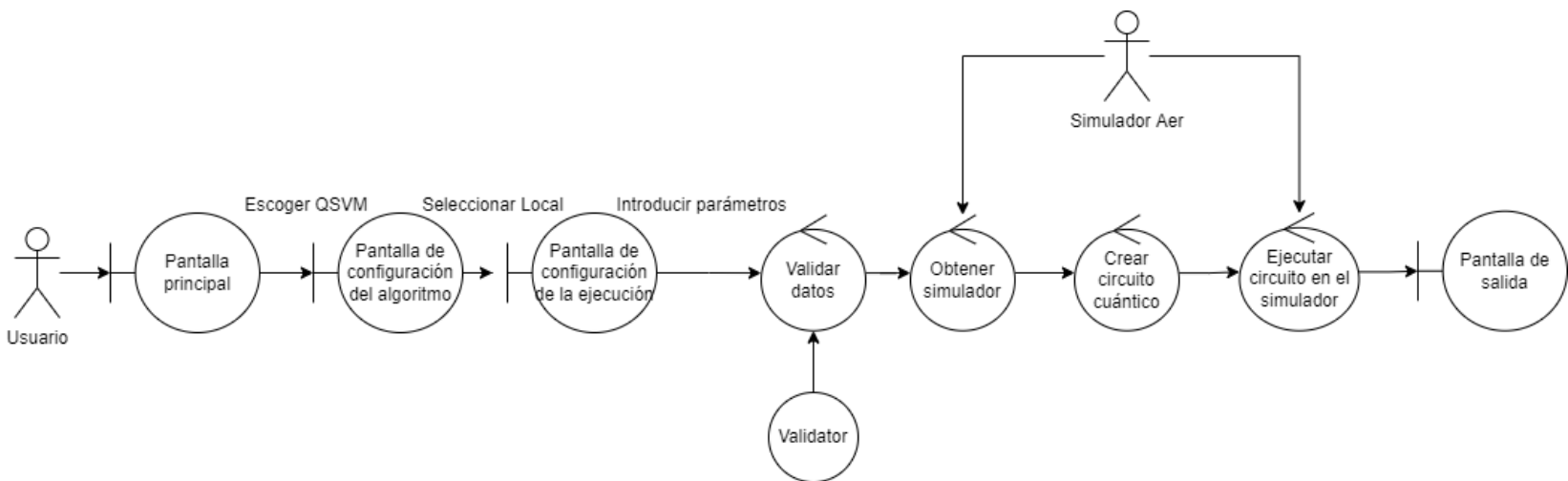


Figura 7.5: Diagrama de robustez - Ejecutar QSVM en un simulador cuántico

Tabla 7.10: Análisis del Caso de Uso - Activar cuenta de IBM

Activar cuenta de IBM	
Precondiciones	El usuario debe tener una cuenta en IBM Quantum Experience. El usuario debe seleccionar ejecutar el sistema en IBM.
Postcondiciones	-
Actores	Usuario y API IBM Quantum
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará uno de los dos algoritmos disponibles (QSVM o QNN). Seleccionará que la ejecución se hará en IBM, y en la siguiente pantalla introducirá el token de su cuenta de IBM Quantum Experience. El sistema verificará que ya se accedió anteriormente debido a que existe una cuenta guardada en disco, por lo que devuelve esta cuenta y permite que el usuario acceda a los ordenadores cuánticos de IBM.
Escenarios Secundarios	El usuario no introdujo anteriormente el token de IBM: el sistema detecta que no hay ninguna sesión en disco, por lo que creará una y la guardará en el caso de que el token sea válido.
Excepciones	El token introducido es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.3.4. Caso de Uso - Ejecutar QSVM en un simulador cuántico

El diagrama de robustez de la Figura 7.5 se recoge el correspondiente al caso de uso Ejecutar QSVM en un simulador cuántico. La tabla 7.11 recoge el análisis de este caso de uso.

Tabla 7.11: Análisis del Caso de Uso - Ejecutar QSVM en un simulador cuántico

Ejecutar QSVM en un simulador cuántico	
Precondiciones	El usuario debe seleccionar ejecutar QSVM en local.
Postcondiciones	-
Actores	Usuario y Simulador Aer
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará QSVM. Seleccionará que la ejecución se hará en local y sobre uno de los conjuntos de datos listados. En la siguiente pantalla seleccionará el simulador y el nº de ejecuciones deseado. El sistema ejecutará QSVM en el simulador seleccionado.
Escenarios Secundarios	-
Excepciones	Alguno de los parámetros introducidos es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.3.5. Caso de Uso - Ejecutar QSVM en un ordenador cuántico

El diagrama de robustez de la Figura 7.6 se recoge el correspondiente al caso de uso Ejecutar QSVM en un ordenador cuántico. La tabla 7.12 recoge el análisis de este caso de uso.

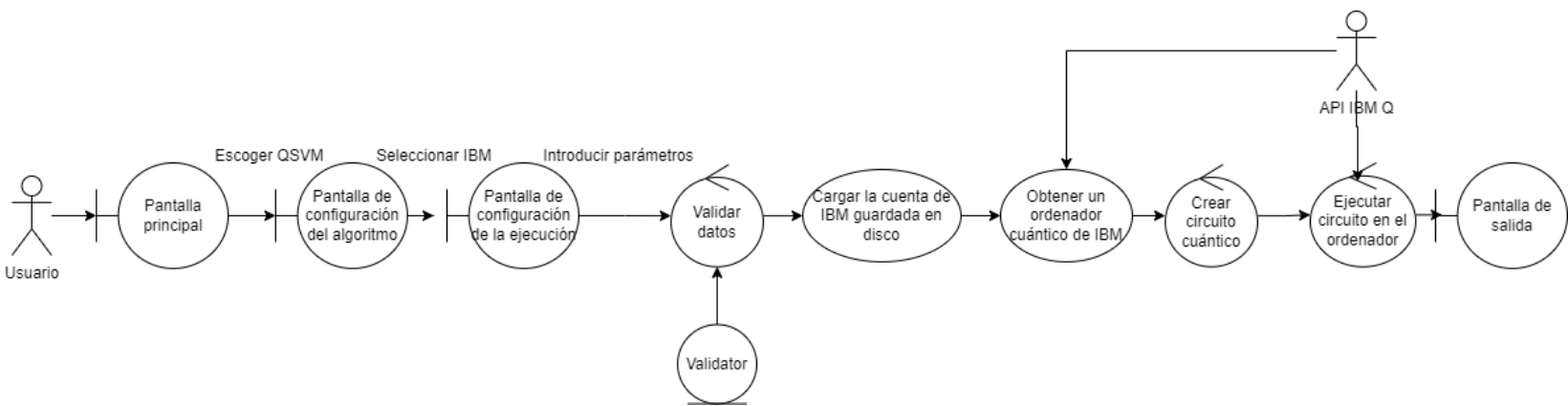


Figura 7.6: Diagrama de robustez - Ejecutar QSVM en un ordenador cuántico

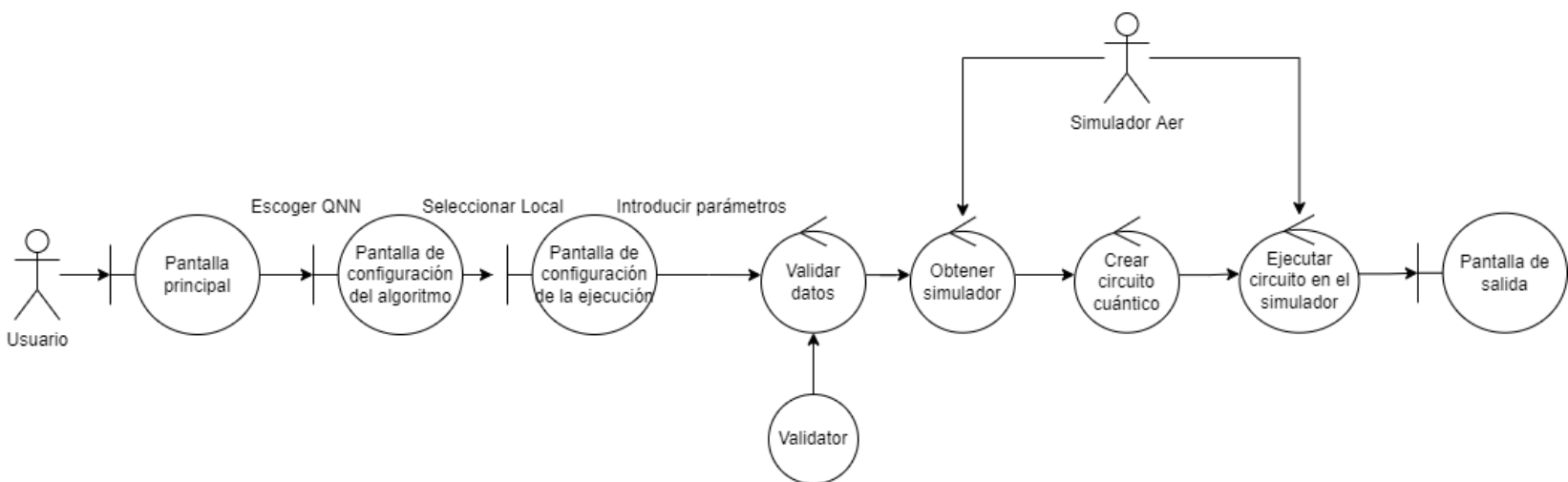


Figura 7.7: Diagrama de robustez - Ejecutar QNN en un simulador cuántico

Tabla 7.12: Análisis del Caso de Uso - Ejecutar QSVM en un ordenador cuántico

Ejecutar QSVM en un ordenador cuántico	
Precondiciones	El usuario debe seleccionar ejecutar QSVM en IBM.
Postcondiciones	-
Actores	Usuario y API IBM Quantum
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará QSVM. Seleccionará que la ejecución se hará en IBM y sobre uno de los conjuntos de datos listados. En la siguiente pantalla se introducirá el TOKEN de autenticación de IBM Quantum y el nº de ejecuciones deseado. El sistema ejecutará QSVM en el ordenador de IBM menos ocupado que sea válido para el problema planteado.
Escenarios Secundarios	-
Excepciones	Alguno de los parámetros introducidos es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.3.6. Caso de Uso - Ejecutar QNN en un simulador cuántico

El diagrama de robustez de la Figura 7.7 se recoge el correspondiente al caso de uso Ejecutar QNN en un simulador cuántico. La tabla 7.13 recoge el análisis de este caso de uso.

Tabla 7.13: Análisis del Caso de Uso - Ejecutar QNN en un simulador cuántico

Ejecutar QNN en un simulador cuántico	
Precondiciones	El usuario debe seleccionar ejecutar QNN en local.
Postcondiciones	-
Actores	Usuario y Simulador Aer
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará QNN. Seleccionará que la ejecución se hará en local y sobre uno de los conjuntos de datos listados. En la siguiente pantalla seleccionará el simulador y el nº de ejecuciones deseado. El sistema ejecutará QNN en el simulador seleccionado.
Escenarios Secundarios	-
Excepciones	Alguno de los parámetros introducidos es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.3.7. Caso de Uso - Ejecutar QNN en un ordenador cuántico

El diagrama de robustez de la Figura 7.8 se recoge el correspondiente al caso de uso Ejecutar QNN en un ordenador cuántico. La tabla 7.14 recoge el análisis de este caso de uso.

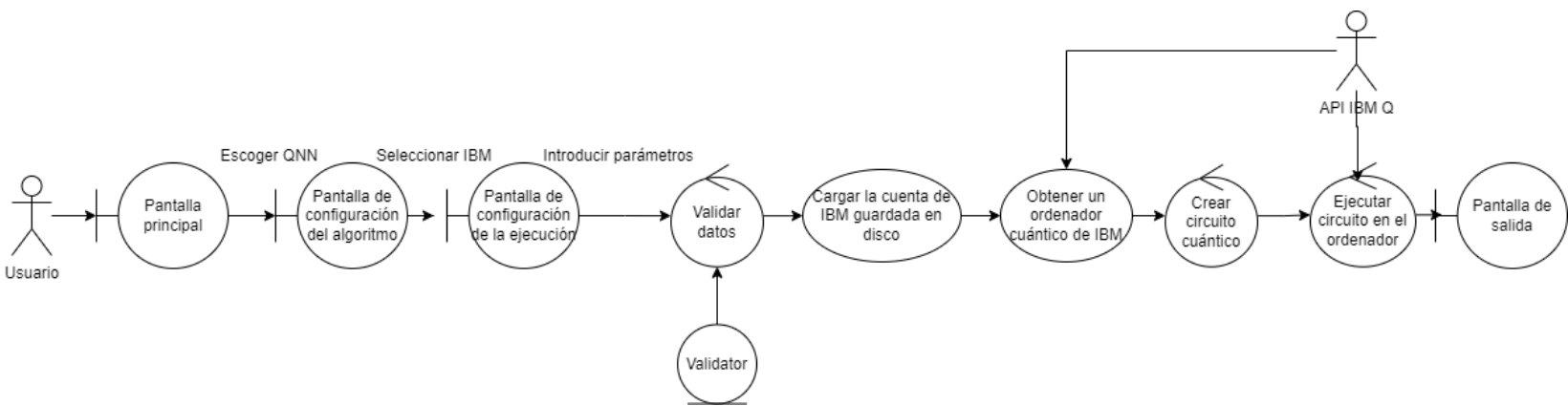


Figura 7.8: Diagrama de robustez - Ejecutar QNN en un ordenador cuántico

Tabla 7.14: Análisis del Caso de Uso - Ejecutar QNN en un ordenador cuántico

Ejecutar QNN en un ordenador cuántico	
Precondiciones	El usuario debe seleccionar ejecutar QNN en IBM.
Postcondiciones	-
Actores	Usuario y API IBM Quantum
Descripción	El usuario accederá a la pantalla principal de la aplicación y seleccionará QNN. Seleccionará que la ejecución se hará en IBM y sobre uno de los conjuntos de datos listados. En la siguiente pantalla se introducirá el TOKEN de autenticación de IBM Quantum y el n ^o de ejecuciones deseado. El sistema ejecutará QNN en el ordenador de IBM menos ocupado que sea válido para el problema planteado.
Escenarios Secundarios	-
Excepciones	Alguno de los parámetros introducidos es inválido. Notificar un error asociado al problema encontrado.
Notas	-

7.4. ANÁLISIS DE CLASES

7.4.1. Diagrama de Clases

En la Figura 7.9 se puede ver el diagrama de clases del sistema implementado.

7.4.2. Descripción de las Clases

En esta sección se describirá de forma breve la composición de cada clase representada en el apartado 7.4.1 Diagrama de Clases.

Tabla 7.15: Descripción de diseño de QMLAlgorithm

QMLAlgorithm
Descripción
Es la encargada de las acciones y la renderización de la pantalla de inicio de sesión.
Atributos propuestos
dataset: Es el conjunto de datos escogido por el usuario.
execution_type: Es el tipo de ejecución del algoritmo. Podrá tener dos posibles valores: local o ibm. En cualquier otro caso se producirá un error.
ml_model: Es modelo de aprendizaje supervisado escogido por el usuario. Podrá tener dos posibles valores: qsvm o qnn. En cualquier otro caso se producirá un error.
n_executions: Es el número de veces que va a ejecutarse el algoritmo. Debido a la naturaleza probabilística de la computación cuántica, se recomienda establecer un número alto de ejecuciones. IBM recomienda 1024.
device: Es el simulador cuántico escogido por el usuario. En caso de que el usuario haya escogido realizar la ejecución en un ordenador cuántico, este atributo será nulo.
Métodos propuestos
run: Prepara el algoritmo al completo y lo ejecuta.
create_quantum_instance: Crea el circuito cuántico y el entorno de ejecución del algoritmo (en local o en ibm).
create_quantum_model: Crea el modelo de aprendizaje con el conjunto de datos seleccionado por el usuario.

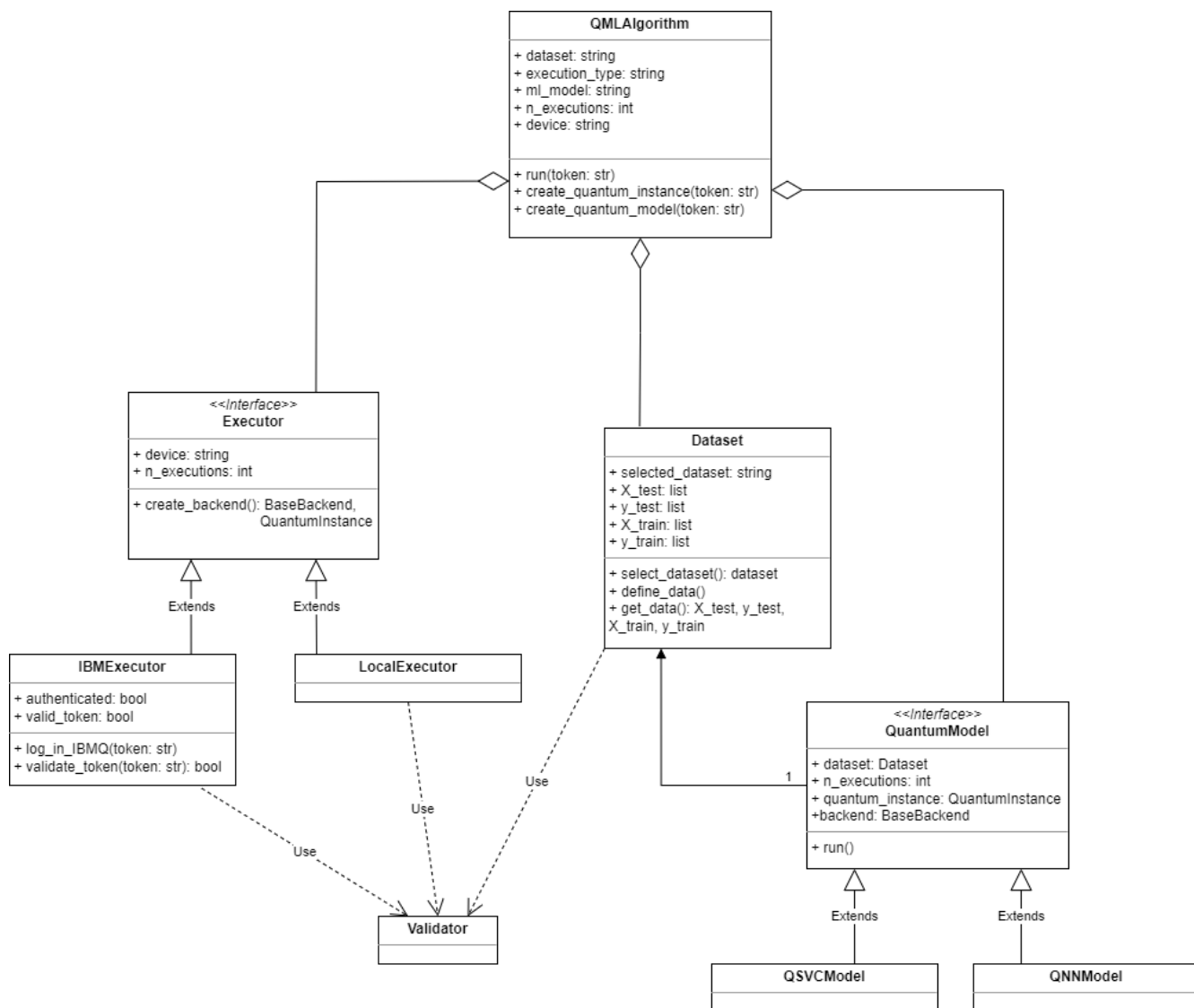


Figura 7.9: Diagrama de Clases

Tabla 7.16: Descripción de diseño de Executor

Executor
Descripción
Es una interfaz que da estructura al sistema que ejecutará el algoritmo. Este sistema podrá ser local o el propio de IBM para los ordenadores cuánticos. Ambos comparten esta misma interfaz.
Atributos propuestos
n_executions: Es el número de veces que va a ejecutarse el algoritmo.
device: Es el simulador cuántico escogido por el usuario. En base a esto, se creará un Executor de tipo IBM o de tipo local.
Métodos propuestos
create_backend: Crea el circuito cuántico y el entorno de ejecución del algoritmo (en local o en ibm).

Tabla 7.17: Descripción de diseño de IBMExecutor

IBMExecutor
Descripción
Es la encargada de crear el sistema backend de IBM dónde se ejecutará nuestro algoritmo. Se conectará con la API IBM Quantum Experience para obtener acceso a ordenadores cuánticos reales.
Atributos propuestos
authenticated: Este parámetro servirá para saber si existe una cuenta de IBM almacenada en disco.
valid_token: Este parámetro servirá para saber si el token que ha introducido el usuario es válido o no y actuar en consecuencia.
Métodos propuestos

Tabla 7.18: Descripción de diseño de LocalExecutor

LocalExecutor
Descripción
Es la encargada de crear el sistema backend del simulador local dónde se ejecutará nuestro algoritmo. Se obtendrá un simulador del proveedor Aer seleccionado por el usuario.
Atributos propuestos
Métodos propuestos

Tabla 7.19: Descripción de diseño de Validator

Validator
Descripción
Es la encargada de comprobar que los parámetros introducidos por el usuario son correctos. En caso contrario, se producirá una excepción.
Atributos propuestos
Métodos propuestos

Tabla 7.20: Descripción de diseño de Dataset

Dataset
Descripción
Es la encargada de crear el conjunto de datos seleccionado por el usuario.
Atributos propuestos
selected_dataset: Este parámetro indicará el nombre del dataset seleccionado.
X_test: Este parámetro es el subconjunto de valores para la predicción de valores.
y_test: Este parámetro es el subconjunto de etiquetas para la predicción de valores.
X_train: Este parámetro es el subconjunto de valores para el entrenamiento del modelo.
y_train: Este parámetro es el subconjunto de etiquetas para el entrenamiento del modelo.
Métodos propuestos
select_dataset: Este método es el encargado de instanciar el conjunto de datos seleccionado por el usuario.
define_data: Este método es el encargado de dividir el conjunto de datos seleccionado en subconjuntos de entrenamiento y pruebas.
select_dataset: Este método devuelve todos los subconjuntos de datos creados.

Tabla 7.21: Descripción de diseño de QuantumModel

QuantumModel
Descripción
Esta interfaz es la encargada de crear una estructura común para el modelo QSVM y QNN. Tendrá todos los elementos necesarios para ejecutar el algoritmo de aprendizaje y devolver el resultado al usuario.
Atributos propuestos
dataset: Es el conjunto de datos procesado que ha sido seleccionado por el usuario.
n_executions: Es el número de veces que va a ejecutarse el algoritmo.
quantum_instance: Es el backend cuántico que incluye la configuración necesaria para crear y ejecutar el circuito cuántico.
backend: Es el sistema cuántico seleccionado para ejecutar el algoritmo. Puede ser un simulador o un ordenador cuántico.
Métodos propuestos
run: Este método se encarga de crear el circuito con el modelo seleccionado, entrenarlo y devolver el resultado obtenido.

Tabla 7.22: Descripción de diseño de QSVModel

QSVModel
Descripción
Es el modelo que contiene el algoritmo de aprendizaje para problemas de clasificación QSVM.
Atributos propuestos
Métodos propuestos

Tabla 7.23: Descripción de diseño de QNNModel

QNNModel
Descripción
Es el modelo que contiene el algoritmo de aprendizaje para problemas de clasificación QNN.
Atributos propuestos
Métodos propuestos

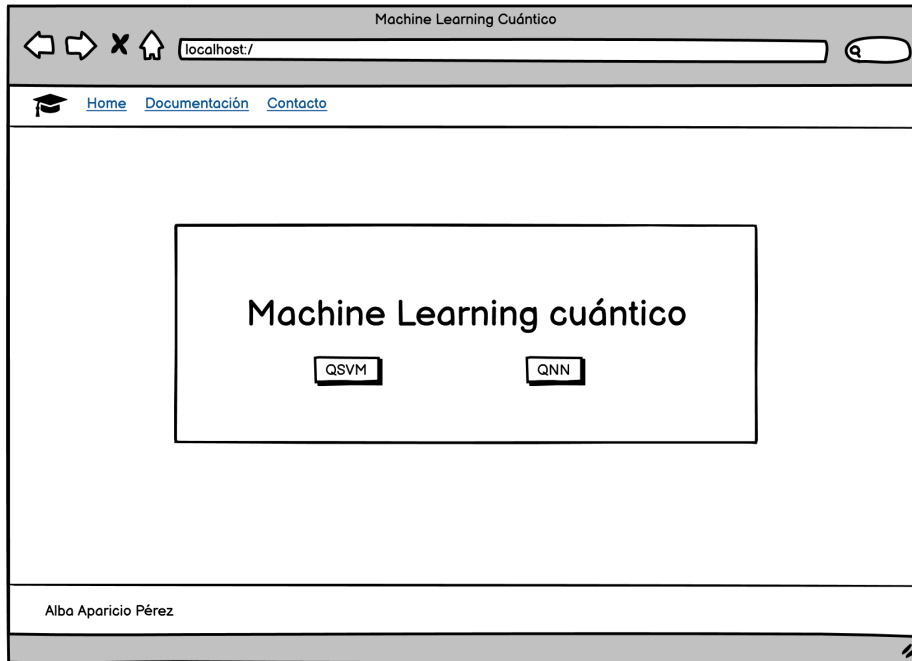


Figura 7.10: Boceto - Pantalla principal

7.5. DEFINICIÓN DE INTERFACES DE USUARIO

7.5.1. Descripción de la Interfaz

En esta sección se presentará, haciendo uso de bocetos, el diseño que tendrán las distintas pantallas de la aplicación web, además de una descripción de las funcionalidades para mostrar cómo el usuario podrá hacer uso de ellas.

A continuación, se mostrarán los prototipos de todas las pantallas que se ha decidido incorporar a la aplicación web.

Cabe destacar que, para esta aplicación web, no es necesario que los usuarios se identifiquen, por lo que no existe ningún formulario de registro ni de inicio de sesión, ya que no está pensada para desplegarse en un servidor, tan solo en local.

7.5.1.1. Pantalla principal

En la Figura 7.10, podemos ver el boceto de la pantalla principal. Esta pantalla es el menú inicial de la aplicación web, a la cual se puede acceder pulsando el botón **Home** desde la barra de navegación.

En esta pantalla tendremos dos botones, QSVM y QNN, que nos redirigirán a las pantallas de configuración de cada algoritmo.

7.5.1.2. Pantalla del algoritmo

En la Figura 7.11 podemos ver el boceto de la pantalla de configuración del algoritmo QSVM. Aquí, el usuario podrá especificar el conjunto de datos deseados

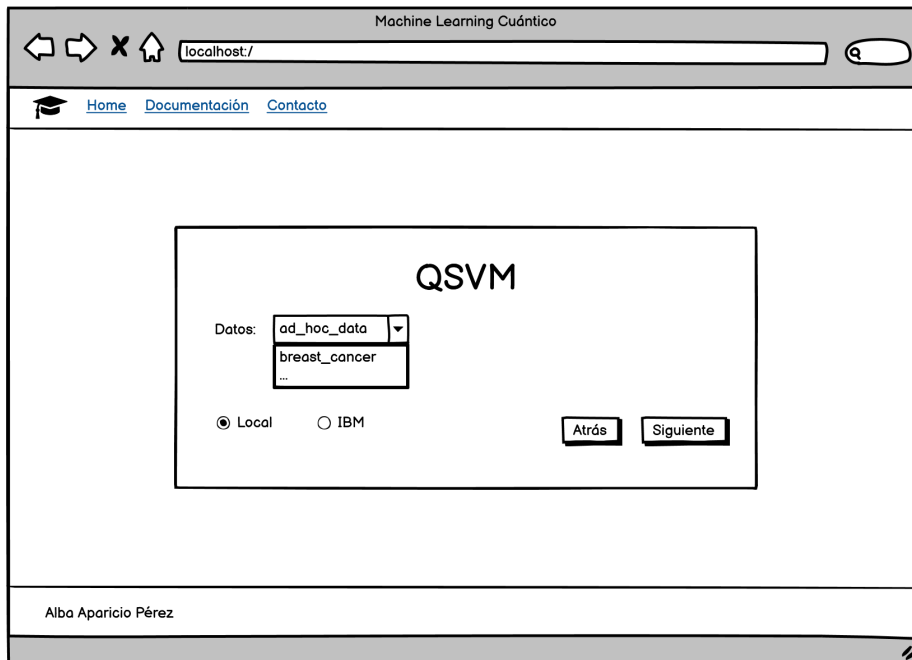


Figura 7.11: Boceto - Pantalla del algoritmo QSVM

entre los listados y el tipo de ejecución, que será en local o en un ordenador de IBM.

En la Figura 7.12 podemos ver el boceto de la pantalla de configuración del algoritmo QNN. Aquí, el usuario podrá especificar el conjunto de datos deseados entre los listados y el tipo de ejecución, que será en local o en un ordenador de IBM.

En ambas pantallas se encuentran dos botones, **Atrás** para volver al menú principal, y **Siguiente** para pasar a la Pantalla de configuración de la ejecución.

7.5.1.3. Pantalla de configuración de la ejecución

En la Figura 7.13, podemos ver el boceto de la pantalla de configuración de la ejecución en un simulador cuántico. En esta pantalla el usuario podrá especificar el simulador en el que quiere ejecutar el algoritmo y el número de ejecuciones que se realizarán. El nº de ejecuciones será 1024 por defecto, ya que se garantiza una mayor exactitud en las predicciones.

En la Figura 7.14, podemos ver el boceto de la pantalla de configuración de la ejecución en un ordenador de IBM. En esta pantalla el usuario podrá especificar el TOKEN de autenticación de su cuenta de IBM Quantum Experience (requisito necesario para realizar la ejecución en un ordenador cuántico). También se podrá especificar el número de ejecuciones que se realizarán. El nº de ejecuciones será

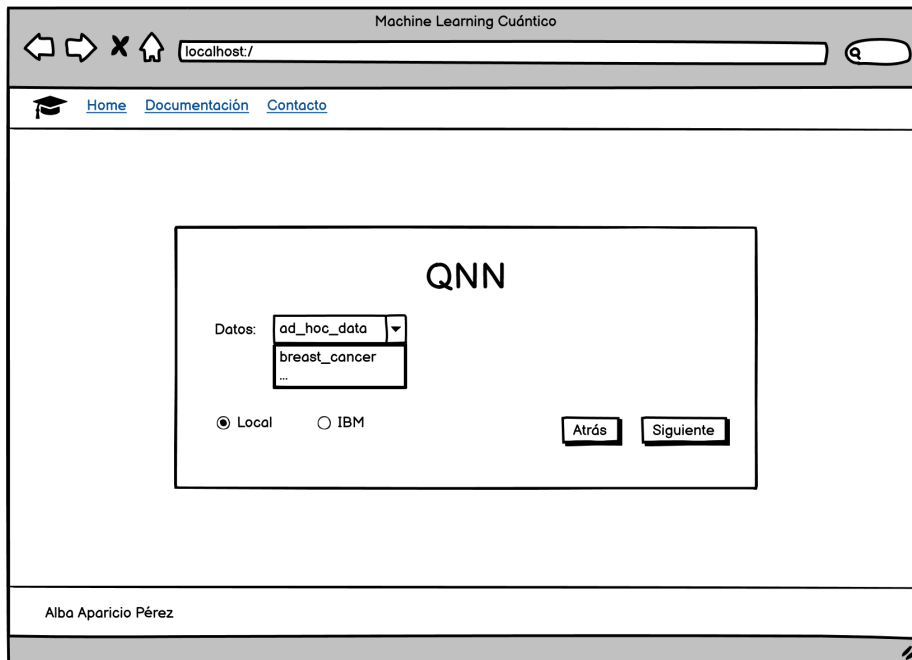


Figura 7.12: Boceto - Pantalla del algoritmo QNN

1024 por defecto.

Ambas pantallas tendrán un botón **Cancelar** para volver al menú principal, otro botón **Atrás** para volver a la pantalla de configuración del algoritmo y por último el botón de **Siguiente**, el cuál activará la ejecución del algoritmo si todos los parámetros son válidos y redirigirá al usuario a la pantalla de salida.

7.5.1.4. Pantalla de salida

En la Figura 7.15, podemos ver el boceto de la pantalla de salida. En esta pantalla se mostrarán los resultados obtenidos de la ejecución realizada. Estos resultados se componen de los valores obtenidos y el porcentaje de exactitud alcanzado, además de una serie de gráficos en el caso de que el algoritmo los haya generado.

7.5.1.5. Pantalla de documentación

En la Figura 7.16, podemos ver el boceto de la pantalla de documentación. En esta pantalla se mostrará en formato pdf esta misma documentación del proyecto. De esta forma, los usuarios podrán acceder a la documentación directamente e informarse del uso, objetivos y funciones del sistema.

Los usuarios podrán acceder a esta pantalla a través del botón **Documentación** que se encuentra en la barra de navegación.

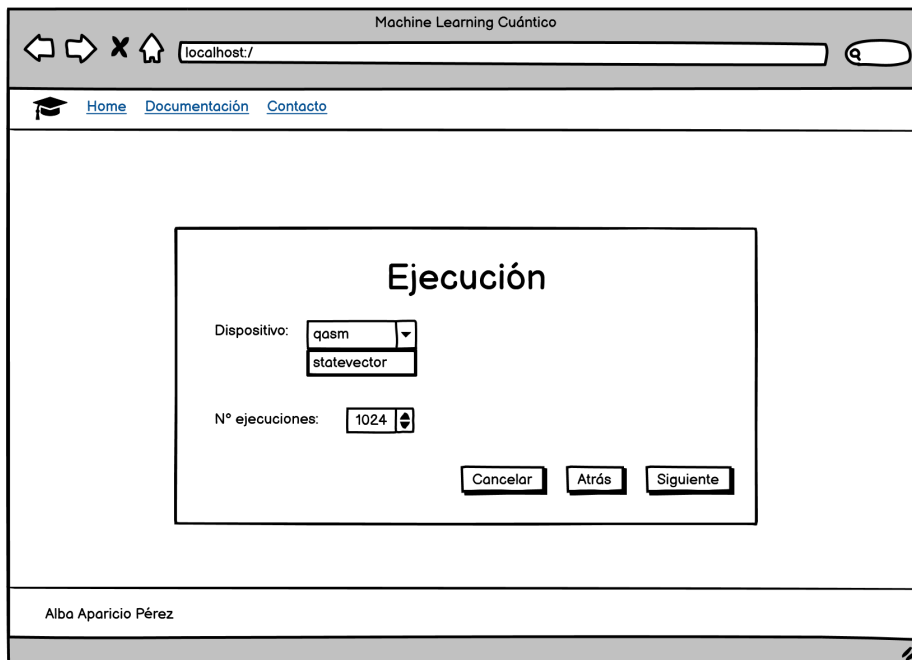


Figura 7.13: Boceto - Pantalla de configuración de la ejecución en un simulador

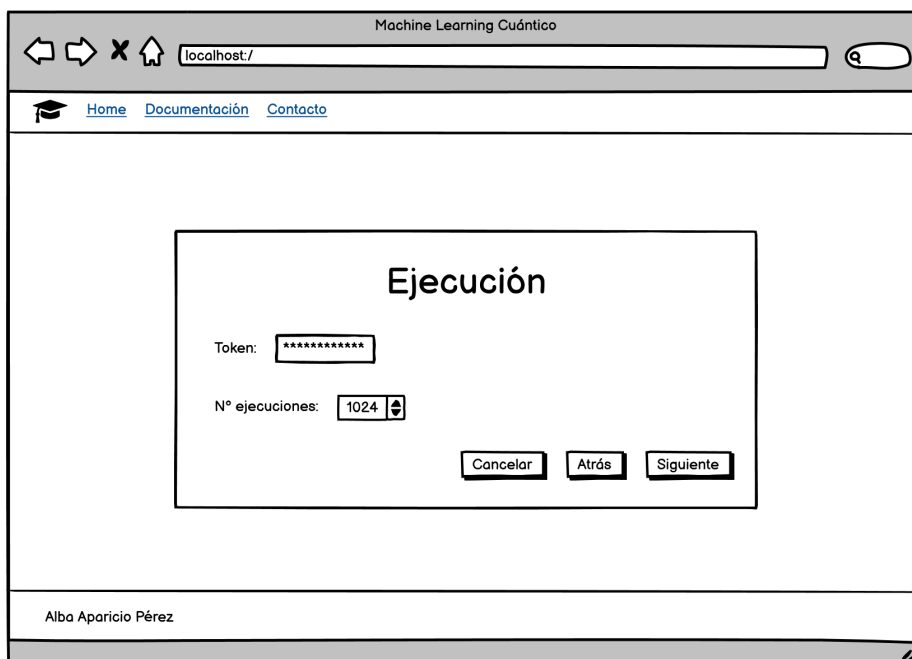


Figura 7.14: Boceto - Pantalla de configuración de la ejecución en un ordenador de IBM

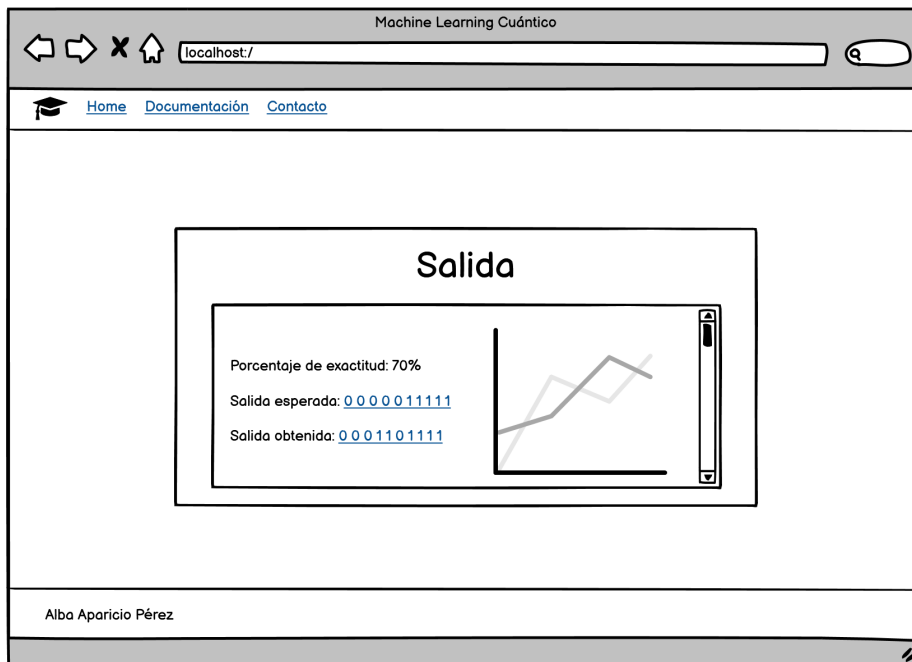


Figura 7.15: Boceto - Pantalla de salida

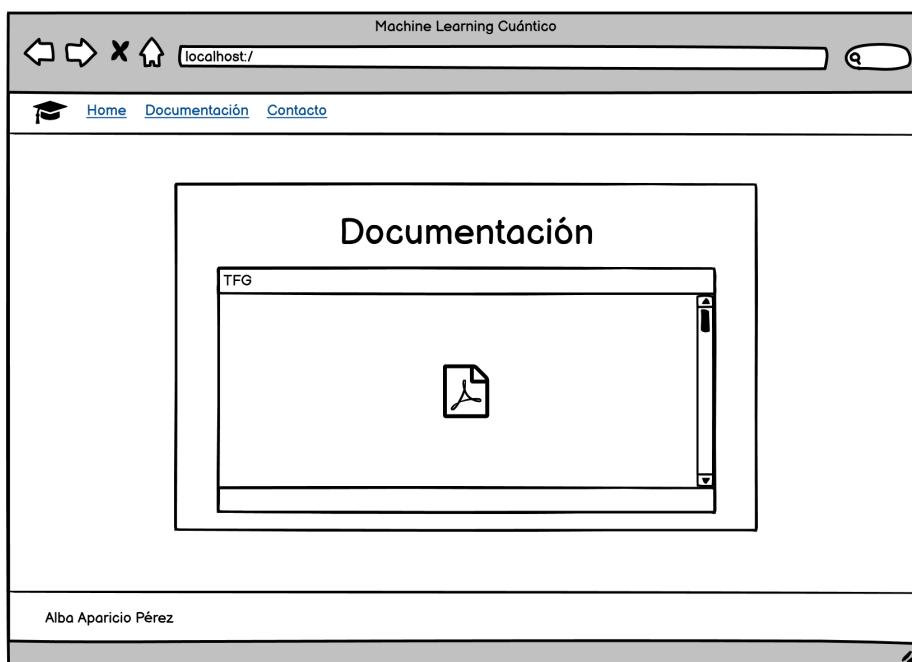


Figura 7.16: Boceto - Pantalla de documentación

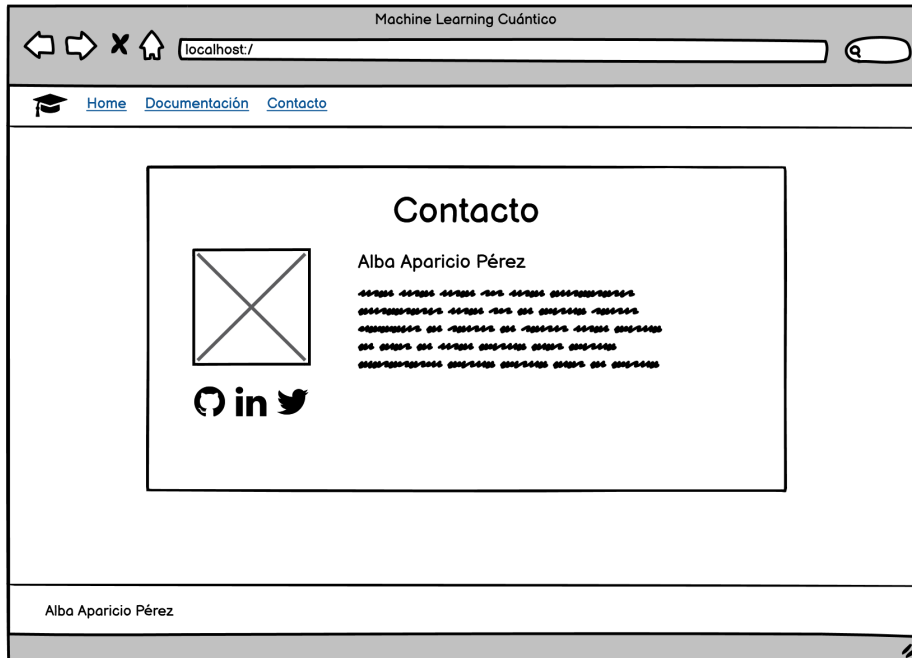


Figura 7.17: Boceto - Pantalla de contacto

7.5.1.6. Pantalla de contacto

En la Figura 7.17, podemos ver el boceto de la pantalla de contacto. En esta pantalla se mostrará una breve carta de presentación mía. También se incluyen algunos links a diversas redes sociales por si los usuarios estuviesen interesados en contactar conmigo por temas relacionados con este proyecto o similares.

Los usuarios podrán acceder a esta pantalla a través del botón **Contacto** que se encuentra en la barra de navegación.

7.5.2. Descripción del Comportamiento de la Interfaz

La interfaz ha sido diseñada de forma que fuese lo más sencilla y cómoda para los usuarios. De esta forma se consigue minimizar el nº de fallos potenciales.

Hay algunas pantallas que requieren que el usuario introduzca una serie de parámetros, por lo que estos se validarán lo antes posible.

En las pantallas de los algoritmos QSVM y QNN, de las figuras 7.11 y 7.12, se requiere que se introduzca el conjunto de datos deseado y el tipo de ejecución. Para evitar valores vacíos, los campos estarán inicializados. En el caso del conjunto de datos, se inicializa con el primer elemento de la lista, y en el caso del tipo de ejecución, se inicializa con la ejecución en local.

Para evitar errores, también se comprobará, en el backend, que el conjunto de datos seleccionado existe y es válido, y que el tipo de ejecución es local o en ibm, pero no otro valor.

En las pantallas de configuración de la ejecución, mostradas en las figuras 7.14 y 7.13, también se solicitarán diversos valores.

En el caso de que se seleccione la ejecución local, se solicitará al usuario que especifique el simulador cuántico a usar y el número de ejecuciones del algoritmo. Para validar el simulador, se comprobará que el seleccionado es alguno de los mostrados en la lista y que existe y es válido. Para el número de ejecuciones, se comprobará que es un número natural.

Para evitar errores por campos vacíos, ambos serán inicializados con un valor por defecto.

En el caso de que se seleccione la ejecución en IBM, se solicitará al usuario que especifique el TOKEN de autenticación de su cuenta de IBM y el número de ejecuciones del algoritmo. Para validar el TOKEN, el sistema se comunicará con la API de IBM para comprobar que este se corresponde con una cuenta real y es correcto. Para el número de ejecuciones, se comprobará que es un número natural.

En el resto de las pantallas mostradas no se requiere introducir ningún valor de entrada, por lo que no se implementa ningún convenio para la validación de datos.

En el caso de que surja algún problema o excepción imprevista, la aplicación le redirigirá al usuario a una pantalla de error, dónde se le mostrará la causa del error en el caso de que sea posible. El boceto de esta pantalla de error es el mostrado en la Figura 7.18.

7.5.3. Diagrama de Navegabilidad

En la Figura 7.19, podemos ver el diagrama de navegabilidad que muestra el paso de unas pantallas a otras. Como se puede ver, se podrá acceder tanto a la pantalla principal, como a la pantalla de documentación y la de contacto desde cualquier pantalla del sistema haciendo click en los links dispuestos en la barra de navegación.

Se accederá a la pantalla de error siempre y cuando se produzca un error de validación desde alguna de las pantallas de algoritmos o de ejecución, pero también si se intenta acceder a una página que no se encuentra definida en el sistema.

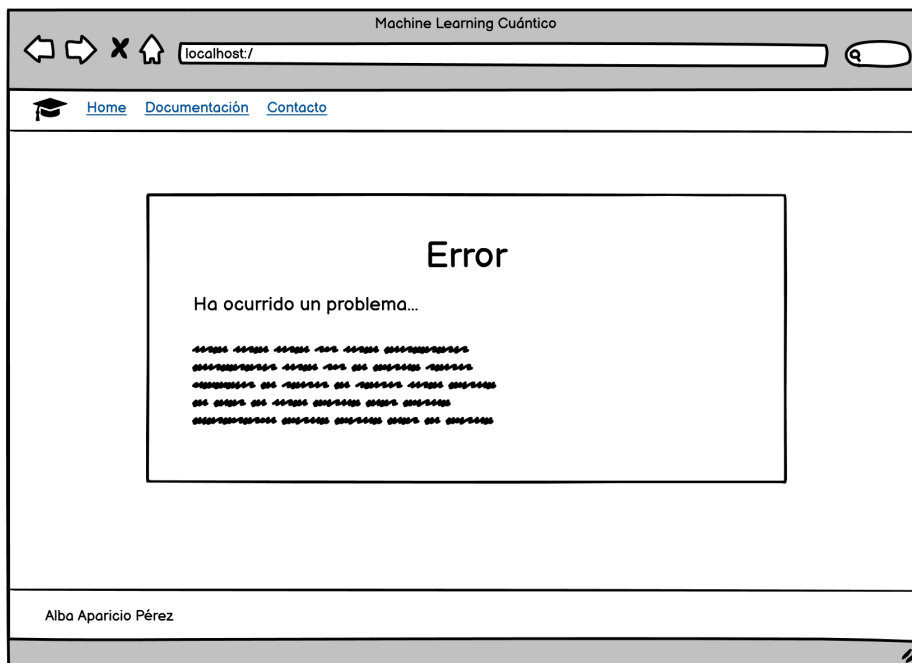


Figura 7.18: Boceto - Pantalla de mensaje de error

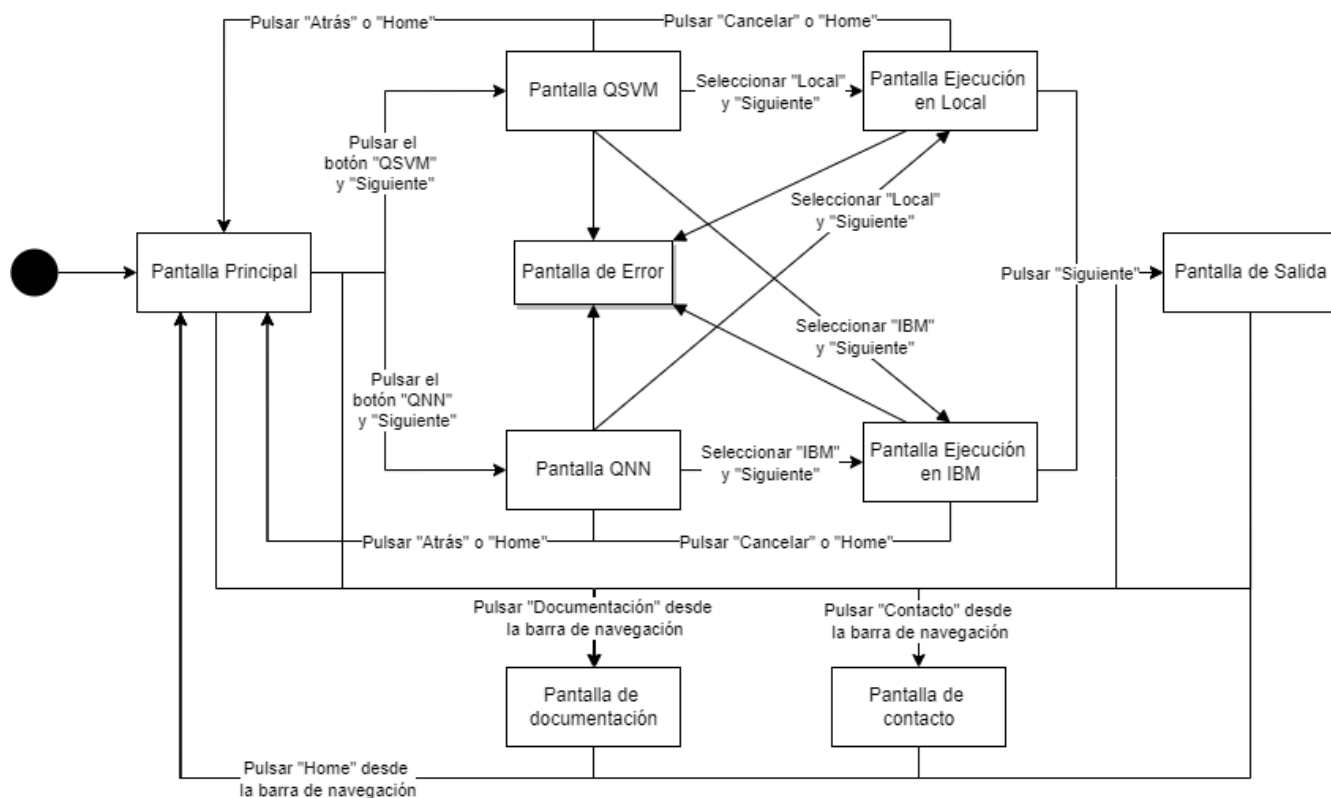


Figura 7.19: Diagrama de navegabilidad

7.6. ESPECIFICACIÓN DEL PLAN DE PRUEBAS

Este prototipo contará con un conjunto de pruebas unitarias agrupando únicamente el comportamiento esencial, de forma que queda bastante limitado.

Ya que la aplicación web no será alojada en un servidor, tan sólo se podrá ejecutar en local con la intención de comprobar que el sistema cuántico funciona correctamente. No serán realizadas pruebas de accesibilidad ni usabilidad con usuarios.

En el caso de que el prototipo se extendiera en un futuro, se podrían añadir más clases de pruebas de forma sencilla.

7.6.1. Pruebas unitarias

Las pruebas unitarias son aquellas que prueban comportamientos únicos de las clases que no tienen relación con sistemas externos y son esenciales para lograr el objetivo de la clase. Cada prueba comprobará que se obtienen los resultados esperados de un método concreto.

Se han realizado pruebas unitarias de los siguientes elementos:

- clase Dataset
 - `select_dataset`
 - `define_data`
- IBMExecutor
 - `is_authenticated`

Para estas pruebas se han empleado las herramientas **fixture** y **mock**, incluidas en el paquete `pytest`. Fixture sirve para generar datos antes de la creación de los tests que serán de utilidad para la ejecución de los mismos. En este caso, se crea un objeto `Dataset` que luego es utilizado en todas las pruebas unitarias.

Mocker es utilizado para reemplazar el comportamiento de los métodos que son llamados en aquel método que estamos probando. De esta forma, aseguramos que el método a probar funciona sin dependencias de otros lugares del código.

7.6.2. Pruebas de sistema

Las pruebas de sistemas están centradas en comprobar que los elementos externos al prototipo funcionan y se utilizan de forma correcta en este sistema.

Este tipo de pruebas agrupa todas aquellas situaciones a cubrir donde intervengan los simuladores cuánticos y los ordenadores cuánticos de IBM.

Se han realizado pruebas unitarias de los siguientes elementos:

- clase `LocalExecutor`

- create_backend
- IBMExecutor
 - validate_token
 - log_in_IBMQ
 - create_backend
- clase QuantumModel
 - run

7.6.3. Pruebas de integración

En este tipo de pruebas se tiene en cuenta que todos los elementos del prototipo funcionen correctamente en conjunto.

La clase QMLAlgorithm integra todas las clases mencionadas anteriormente para formar un algoritmo completo capaz de ejecutarse en un simulador u ordenador, por lo que las pruebas de integración se han realizado sobre los métodos de esta clase.

- clase QMLAlgorithm
 - create_quantum_instance
 - create_quantum_model
 - run

Capítulo 8

DISEÑO DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

DSI

8.1. DISEÑO DE CASOS DE USO REALES

8.1.1. Caso de Uso - Obtener un ordenador cuántico de IBM

8.1.1.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.1 podemos ver el diagrama de secuencia correspondiente al caso de uso obtener un ordenador cuántico de IBM.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador e introducirá su TOKEN de IBM para ejecutar un algoritmo en un ordenador cuántico. Este TOKEN pasará por QMLAlgorithm y IBMExecutor hasta llegar a la API de IBM Q. La API validará el token. En caso de que sea erróneo, se producirá una excepción y se alertará al usuario mostrándole la pantalla de errores. En caso de que el TOKEN sea válido, la API buscará el ordenador disponible que tenga la menor cola de espera y devolverá una instancia de dicho ordenador.

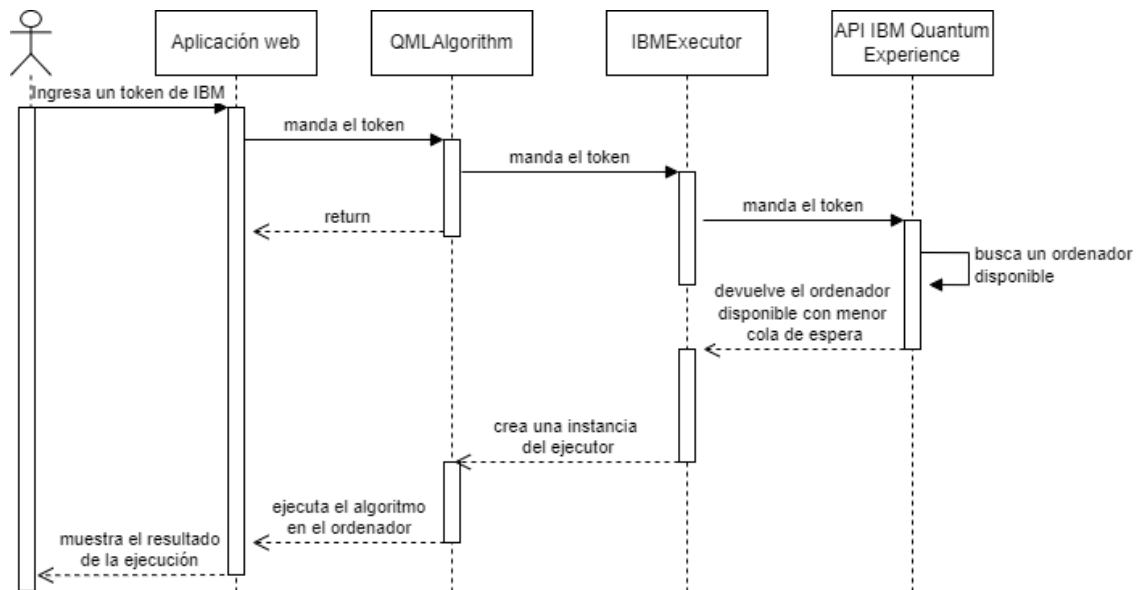


Figura 8.1: Diagrama de interacción - Obtener un ordenador cuántico de IBM

8.1.1.2. Diagramas de Estados de las Clases

En la Figura 8.2, podemos ver el diagrama de estados de la cuenta de IBM del usuario. Para este caso de uso en concreto, se tiene en cuenta el estado de dicha cuenta.

Para poder obtener acceso a un ordenador cuántico el usuario deberá disponer de una cuenta en IBM en el estado **validada y guardada en disco**. En caso contrario, no será posible realizar dicha acción.

El usuario, al ingresar en el sistema, comienza con su cuenta en dos posibles

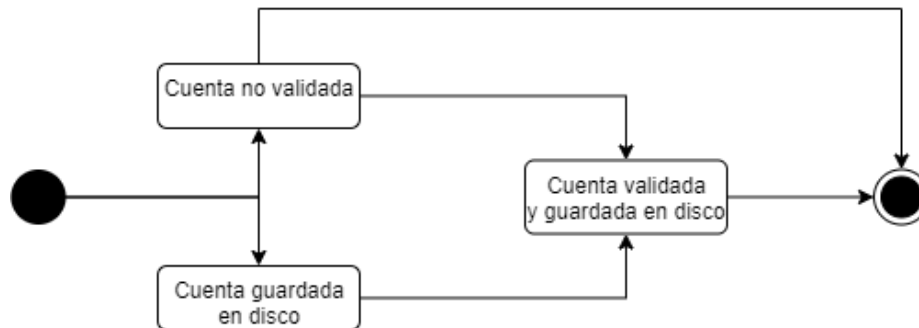


Figura 8.2: Diagrama de estados - Cuenta IBM

estados: **Cuenta no validada**, en el caso de que nunca hubiera ingresado su TOKEN en la aplicación web; o **Cuenta guardada en disco**, en el caso de que hubiera ingresado su TOKEN en una sesión anterior. En ambos casos la cuenta debe ser verificada a través de la API de IBM Q.

Tras la verificación, la cuenta se guardará en disco si no lo estaba ya y pasará a estado **Cuenta validada y guardada en disco**.

Si la API no logra verificar la cuenta, esta pasará al estado final y por lo tanto se abortará la ejecución en el ordenador.

8.1.2. Caso de Uso - Cargar la cuenta de IBM guardada en disco

8.1.2.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.3, podemos ver el diagrama de secuencia correspondiente al caso de uso cargar la cuenta de IBM guardada en disco.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador e introducirá su TOKEN de IBM para ejecutar un algoritmo en un ordenador cuántico. Este TOKEN pasará por QMLAlgorithm y IBMExecutor. En IBMExecutor se comprobará si el usuario ya dispone de una cuenta en disco. Si es así, se cargará dicha cuenta y será activada por la API.

Tras la activación, se crea la instancia del ordenador y posteriormente se ejecutará el algoritmo seleccionado.

8.1.2.2. Diagramas de Estados de las Clases

En la Figura 8.2, podemos ver el diagrama de estados de la cuenta de IBM del usuario. Para este caso de uso en concreto, se tiene en cuenta el estado de dicha cuenta.

Para poder obtener cargar la cuenta de IBM guardada en disco el usuario deberá disponer de una cuenta en IBM en el estado **Cuenta guardada en disco**. En caso contrario, no será posible realizar dicha acción.

El usuario, al ingresar en el sistema, comienza con su cuenta en dos posibles estados: **Cuenta no validada**, en el caso de que nunca hubiera ingresado su

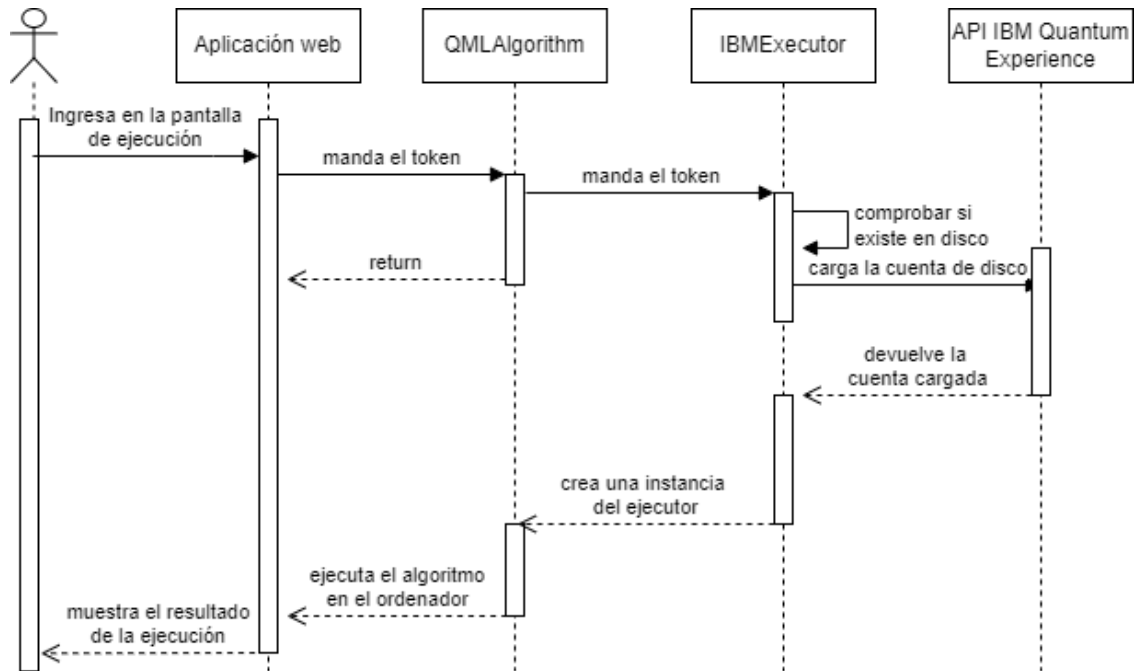


Figura 8.3: Diagrama de interacción - Cargar la cuenta de IBM guardada en disco

TOKEN en la aplicación web; o **Cuenta guardada en disco**, en el caso de que hubiera ingresado su TOKEN en una sesión anterior.

Si la cuenta existiera en disco, la API de IBM Q se encarga de recuperar dicha cuenta y validarla, pasando así al estado **Cuenta validada y guardada en disco**.

Si no se dispone de ninguna cuenta en disco, no se podría realizar esta acción.

8.1.3. Caso de Uso - Activar cuenta de IBM

8.1.3.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.3, podemos ver el diagrama de secuencia correspondiente al caso de uso Cargar la cuenta de IBM guardada en disco.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador e introducirá su TOKEN de IBM para ejecutar un algoritmo en un ordenador cuántico. Este TOKEN pasará por QMLAlgorithm y IBMExecutor. En IBMExecutor, se comprobará si el usuario ya dispone de una cuenta en disco. Si no dispone de ella, entonces se comprobará si el TOKEN introducido por el usuario es correcto. En ese caso, la API de IBM activará la cuenta y la almacenará en disco.

Tras la activación, se crea la instancia del ordenador y posteriormente se ejecutará el algoritmo seleccionado.

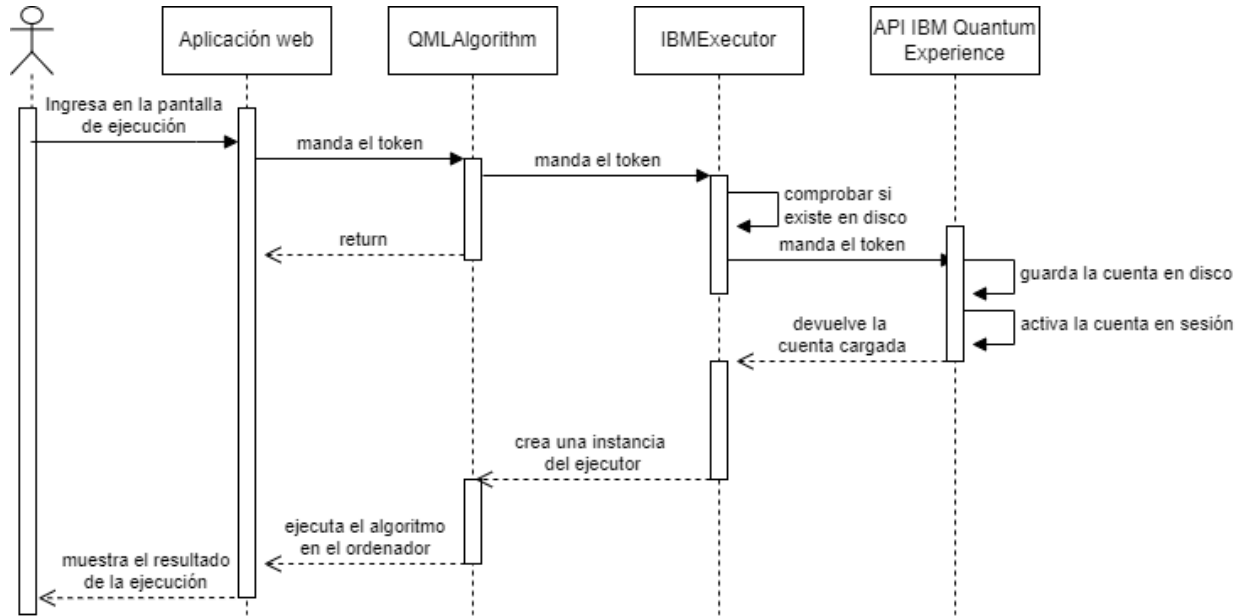


Figura 8.4: Diagrama de interacción - Activar cuenta de IBM

8.1.3.2. Diagramas de Estados de las Clases

En la Figura 8.2, podemos ver el diagrama de estados de la cuenta de IBM del usuario. Para este caso de uso en concreto, se tiene en cuenta el estado de dicha cuenta.

Para poder activar su cuenta de IBM, el usuario deberá disponer de una cuenta en IBM en el estado **no validada** o **guardada en disco**. En caso contrario, no será posible realizar dicha acción.

El usuario, al ingresar en el sistema, comienza con su cuenta en dos posibles estados: **Cuenta no validada**, en el caso de que nunca hubiera ingresado su TOKEN en la aplicación web; o **Cuenta guardada en disco**, en el caso de que hubiera ingresado su TOKEN en una sesión anterior.

En ambos casos, para activar la cuenta, esta debe ser verificada a través de la API de IBM Q.

Tras la verificación, la cuenta se guardará en disco si no lo estaba ya y pasará a estado **Cuenta validada y guardada en disco**.

Si la API no logra verificar la cuenta, esta pasará al estado final y por lo tanto se abortará la ejecución en el ordenador.

8.1.4. Caso de Uso - Ejecutar QSVM en un simulador cuántico

8.1.4.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.5, podemos ver el diagrama de secuencia correspondiente al caso de uso Ejecutar QSVM en un simulador cuántico.

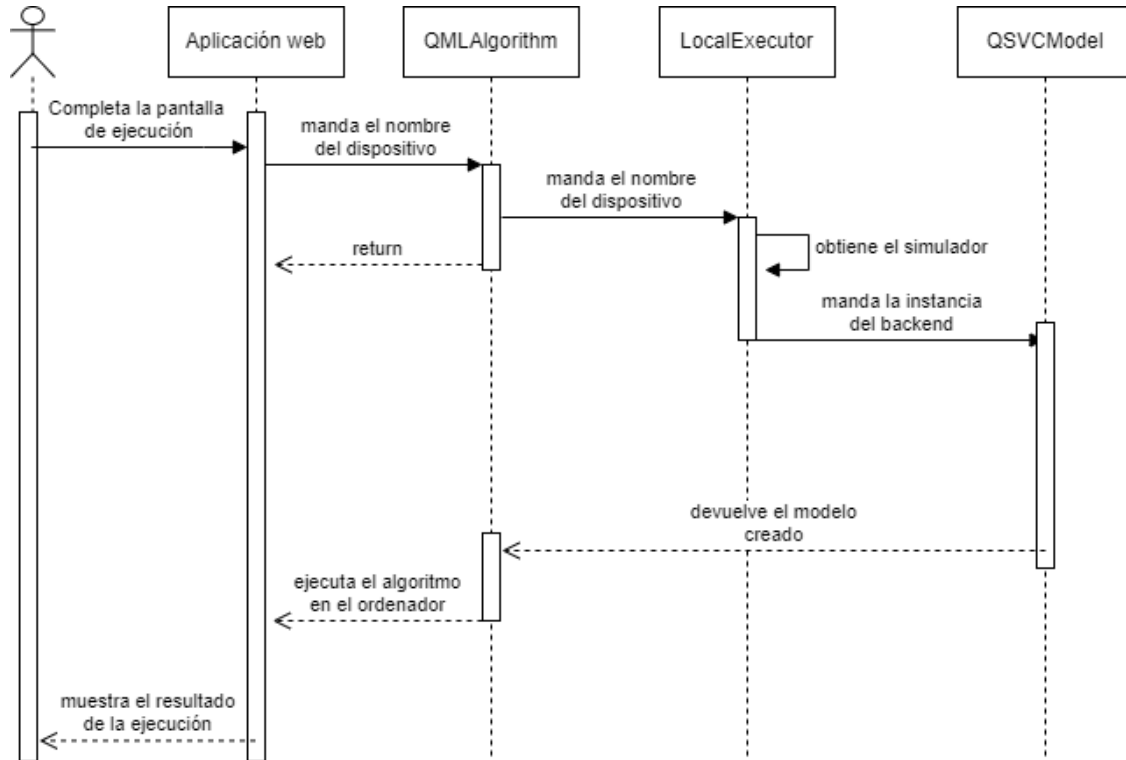


Figura 8.5: Diagrama de interacción - Ejecutar QSVM en un simulador cuántico

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador y seleccionará la opción de ejecutar el algoritmo QSVM en local. Una vez que se introduzcan los parámetros de ejecución, estos se enviarán a QMLAlgorithm, entre ellos el nombre del simulador seleccionado. El nombre se enviará a LocalExecutor, donde se obtendrá una instancia del simulador. QSVModel creará el modelo del algoritmo con el simulador incorporado y será todo enviado de vuelta a QMLAlgorithm para ser ejecutado.

8.1.4.2. Diagramas de Estados de las Clases

Para este caso de uso en concreto se tienen en cuenta los estados mostrados en la Figura 8.6, donde podemos ver el diagrama de estados de la ejecución de un algoritmo de aprendizaje.

Al acceder a la aplicación web, el usuario deberá seleccionar entre el algoritmo QSVM, entonces pasará al estado **QSVM seleccionado**, o QNN, y pasará a **QNN seleccionado**.

Una vez seleccionado el algoritmo QSVM, el usuario seleccionará la opción de ejecución en local, y por lo tanto pasará al estado **Simulador seleccionado**.

En el caso de que en alguno de los anteriores pasos se hubiera introducido un valor erróneo, se pasaría al estado final y, por lo tanto, no se completaría esta acción.

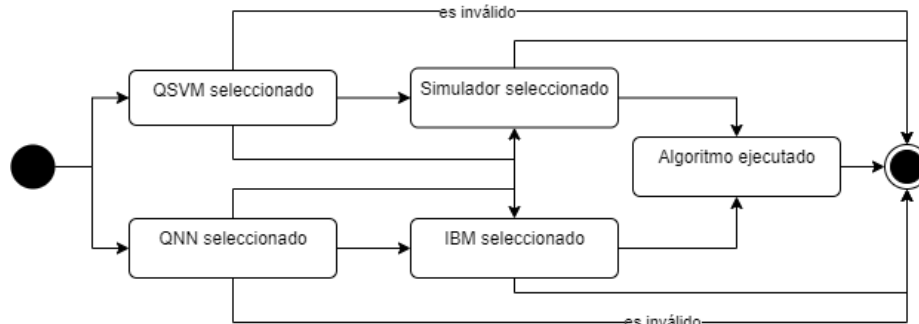


Figura 8.6: Diagrama de estados - Ejecución

Si todos los datos introducidos son correctos, se pasa al estado **Algoritmo ejecutado** y finaliza la ejecución para mostrar los resultados al usuario.

8.1.5. Caso de Uso - Ejecutar QSVM en un ordenador cuántico

8.1.5.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.7, podemos ver el diagrama de secuencia correspondiente al caso de uso ejecutar QSVM en un ordenador cuántico.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador e introducirá su TOKEN de IBM para ejecutar un algoritmo en un ordenador cuántico. Este TOKEN pasará por QMLAlgorithm y IBMExecutor. En IBMExecutor, se validará si el usuario dispone de cuenta en disco y en caso contrario lo autenticará. Una vez que la cuenta del usuario esté verificada, se enviará la instancia del backend del ordenador obtenido por la API a la clase QSVCModel, la cual creará un modelo del algoritmo QSVM con los parámetros especificados por el usuario y un ordenador cuántico como backend.

El modelo será ejecutado y el resultado obtenido será mostrado al usuario.

Tras la activación, se crea la instancia del ordenador y posteriormente se ejecutará el algoritmo seleccionado.

8.1.5.2. Diagramas de Estados de las Clases

Para este caso de uso en concreto se tienen en cuenta los estados mostrados en la Figura 8.6, donde podemos ver el diagrama de estados de la ejecución de un algoritmo de aprendizaje.

Al acceder a la aplicación web, el usuario deberá seleccionar entre el algoritmo QSVM, entonces pasará al estado **QSVM seleccionado**, o QNN, y pasará a **QNN seleccionado**.

Una vez seleccionado el algoritmo QSVM, el usuario seleccionará la opción de ejecución en IBM, y por lo tanto pasará al estado **IBM seleccionado**.

En el caso de que en alguno de los anteriores pasos se hubiera introducido un

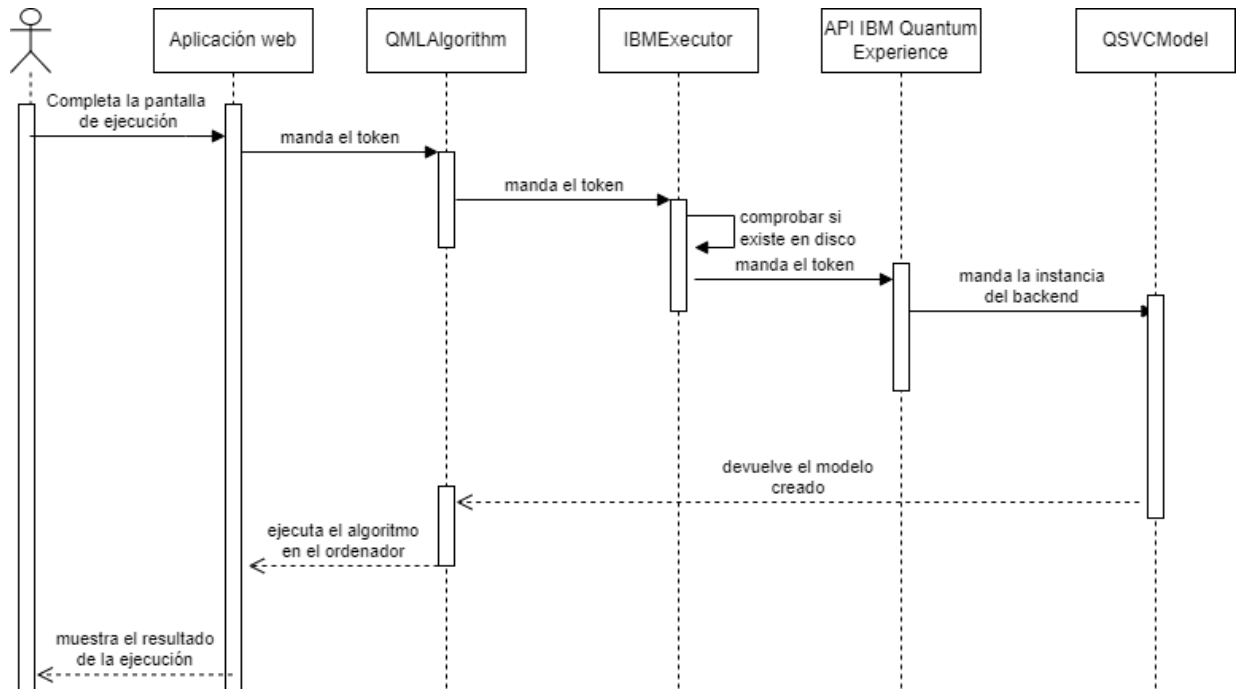


Figura 8.7: Diagrama de interacción - Ejecutar QSVM en un ordenador cuántico

valor erróneo, se pasaría al estado final y, por lo tanto, no se completaría esta acción.

Si todos los datos introducidos son correctos, se pasa al estado **Algoritmo ejecutado** y finaliza la ejecución para mostrar los resultados al usuario.

8.1.6. Caso de Uso - Ejecutar QNN en un simulador cuántico

8.1.6.1. Diagramas de Interacción (Comunicación y Secuencia)

En la Figura 8.8, podemos ver el diagrama de secuencia correspondiente al caso de uso Ejecutar QNN en un simulador cuántico.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador y seleccionará la opción de ejecutar el algoritmo QNN en local. Una vez que se introduzcan los parámetros de ejecución, estos se enviarán a QMLAlgorithm, entre ellos el nombre del simulador seleccionado. El nombre se enviará a LocalExecutor donde se obtendrá una instancia del simulador. QNNModel creará el modelo del algoritmo con el simulador incorporado y será todo enviado de vuelta a QMLAlgorithm para ser ejecutado.

8.1.6.2. Diagramas de Estados de las Clases

Para este caso de uso en concreto se tienen en cuenta los estados mostrados en la Figura 8.6, donde podemos ver el diagrama de estados de la ejecución de un algoritmo de aprendizaje.

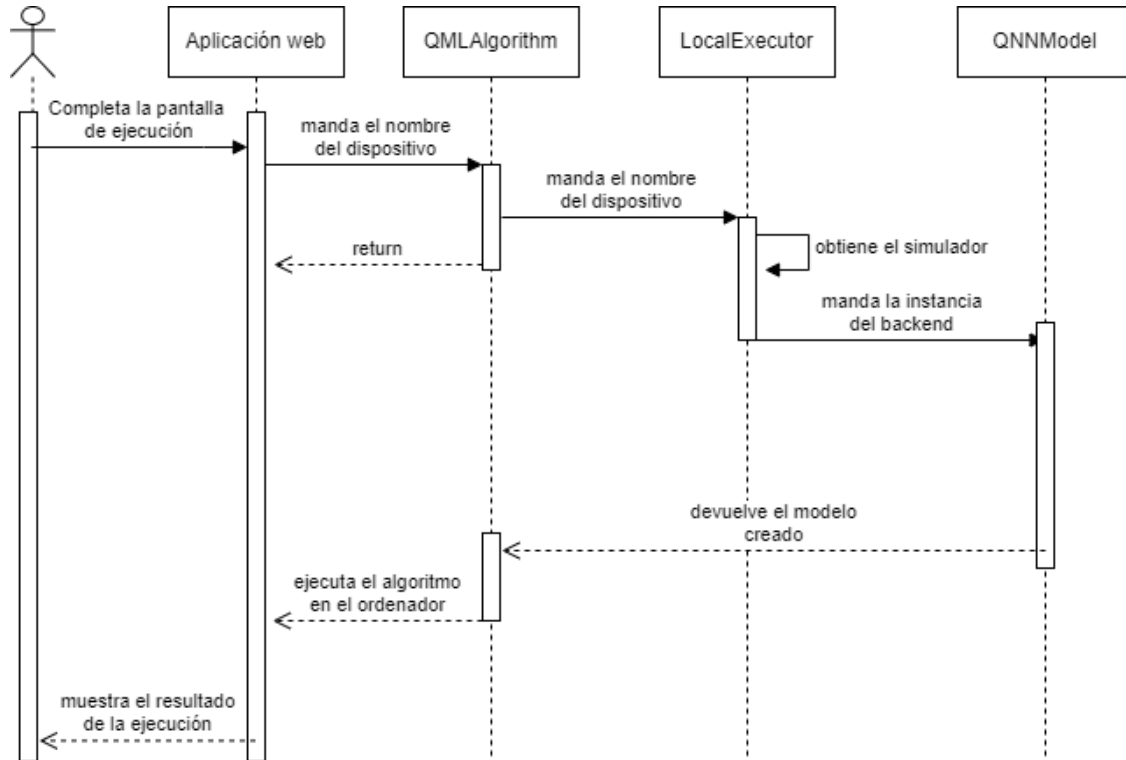


Figura 8.8: Diagrama de interacción - Ejecutar QNN en un simulador cuántico

Al acceder a la aplicación web, el usuario deberá seleccionar entre el algoritmo QSVM, entonces pasará al estado **QSVM seleccionado**, o QNN, y pasará a **QNN seleccionado**.

Una vez seleccionado el algoritmo QNN, el usuario seleccionará la opción de ejecución en local, y por lo tanto pasará al estado **Simulador seleccionado**.

En el caso de que en alguno de los anteriores pasos se hubiera introducido un valor erróneo, se pasaría al estado final y, por lo tanto, no se completaría esta acción.

Si todos los datos introducidos son correctos, se pasa al estado **Algoritmo ejecutado** y finaliza la ejecución para mostrar los resultados al usuario.

8.1.7. Caso de Uso - Ejecutar QNN en un ordenador cuántico

En la Figura 8.9, podemos ver el diagrama de secuencia correspondiente al caso de uso Ejecutar QNN en un ordenador cuántico.

En este diagrama, el actor principal es el usuario. El usuario accederá a la aplicación web desde su navegador e introducirá su TOKEN de IBM para ejecutar un algoritmo en un ordenador cuántico. Este TOKEN pasará por QMLAlgorithm y IBMExecutor. En IBMExecutor se validará si el usuario dispone de cuenta en disco y en caso contrario lo autenticará. Una vez que la cuenta del usuario esté verificada, se enviará la instancia del backend del ordenador obtenido por la

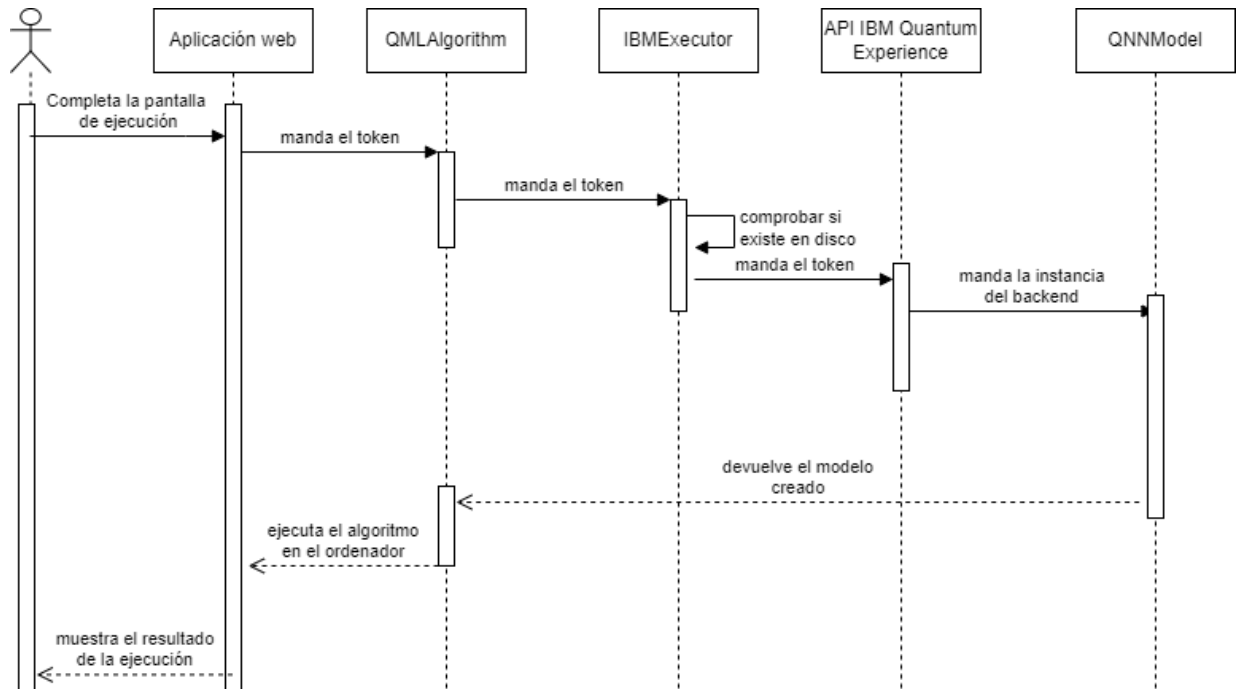


Figura 8.9: Diagrama de interacción - Ejecutar QNN en un ordenador cuántico

API a la clase QNNModel, la cual creará un modelo del algoritmo QNN con los parámetros especificados por el usuario y un ordenador cuántico como backend.

El modelo será ejecutado y el resultado obtenido será mostrado al usuario.

8.1.7.1. Diagramas de Interacción (Comunicación y Secuencia)

8.1.7.2. Diagramas de Estados de las Clases

Para este caso de uso en concreto se tienen en cuenta los estados mostrados en la Figura 8.6, donde podemos ver el diagrama de estados de la ejecución de un algoritmo de aprendizaje.

Al acceder a la aplicación web, el usuario deberá seleccionar entre el algoritmo QSVM, entonces pasará al estado **QSVM seleccionado**, o QNN, y pasará a **QNN seleccionado**.

Una vez seleccionado el algoritmo QNN, el usuario seleccionará la opción de ejecución en IBM y, por lo tanto, pasará al estado **IBM seleccionado**.

En el caso de que en alguno de los anteriores pasos se hubiera introducido un valor erróneo, se pasaría al estado final y por lo tanto no se completaría esta acción.

Si todos los datos introducidos son correctos, se pasa al estado **Algoritmo ejecutado** y finaliza la ejecución para mostrar los resultados al usuario.

8.2. DISEÑO DE CLASES

8.2.1. Diagrama de Clases

En este apartado se mostrara el diseño final de las clases implementadas. Ya se hizo un primer diseño pero menos extenso en el apartado 7.4.1 Diagrama de Clases, por lo que aquí se comentarán en específico algunos detalles y otros elementos que no aparecen en el anterior diagrama.

En la figura 8.10, podemos ver el diagrama de clases al completo. En este diagrama se han añadido nuevas excepciones personalizadas que se podrán utilizar desde cualquier parte de la aplicación. Estas excepciones estarán controladas exclusivamente para que se le muestre al usuario la pantalla de error 9.3 con el mensaje correspondiente.

La jerarquía de los modelos cuánticos se ha implementado siguiendo el patrón de diseño Strategy, de forma que sea sencillo modificar los modelos y añadir otros nuevos en el futuro, ya que es bastante probable que se quiera probar con algoritmos de regresión o incluso también con algoritmos de aprendizaje no supervisado.

La jerarquía de Executor se ha definido también siguiendo un patrón Strategy, debido a que es posible que se quieran añadir otros proveedores de ordenador cuánticos o simuladores. Actualmente, hay varias empresas que están haciendo numerosos avances por lo que hay una probabilidad de que se ofrezca al público la posibilidad de ejecutar diversos algoritmos en ordenadores cuánticos cada vez más potentes, por ejemplo los ordenadores de Google.

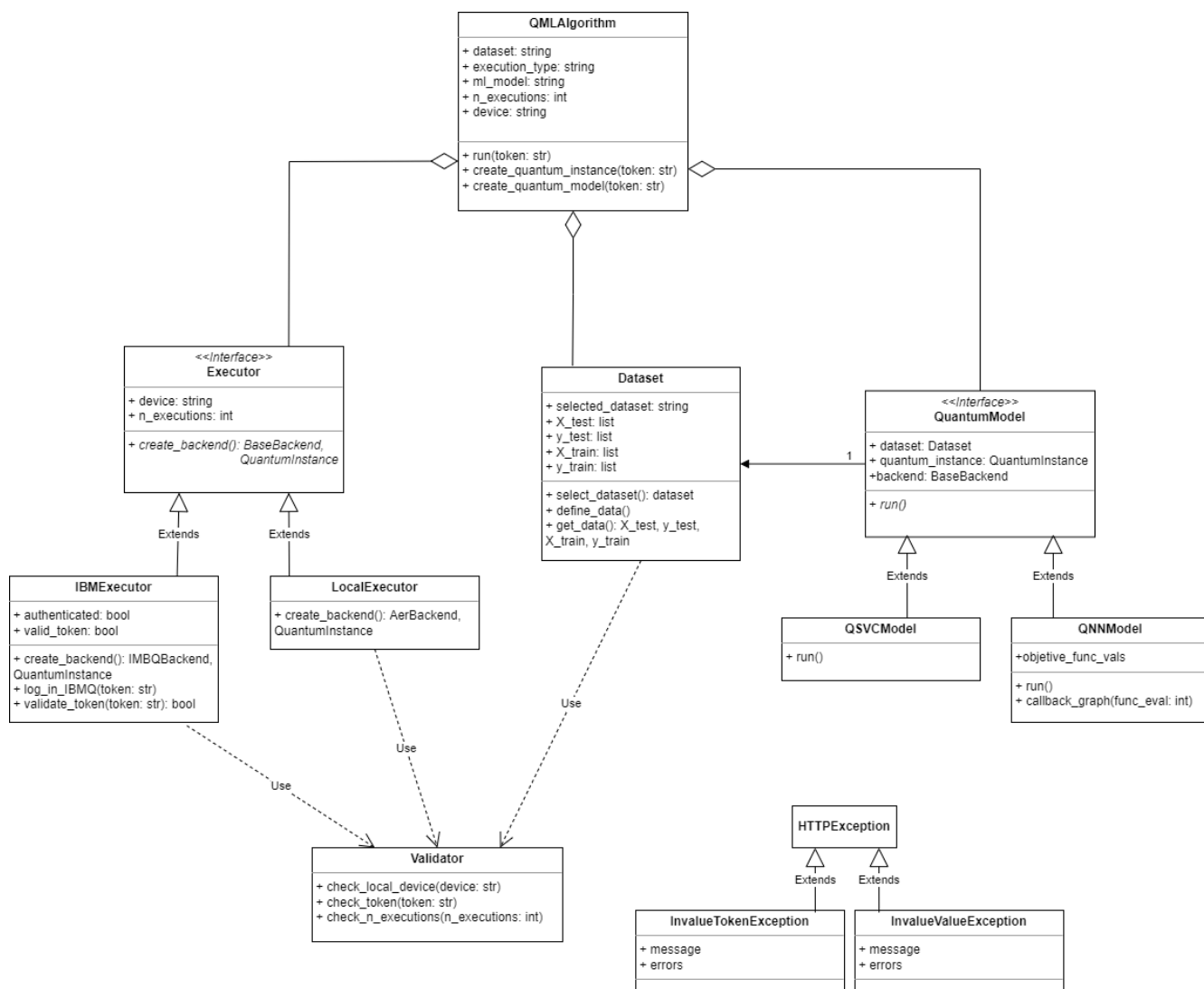


Figura 8.10: Diseño - Diagrama de clases

8.3. DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SISTEMA

8.3.1. Diseño de Módulos del Sistema

En la Figura 8.11 se muestra el diagrama de módulos del sistema.

src

Este módulo agrupa todo la parte de backend del proyecto. Aquí, se incluyen los algoritmos cuánticos y la conexión con simuladores y la API de IBM.

business

Este módulo, al igual que **src**, agrupa toda la funcionalidad del sistema, sin contar con la interfaz gráfica ni la aplicación web.

app

Este componente es el elemento principal que conforma la aplicación web. Será el encargado de importar el paquete Flask, crear una instancia de la aplicación web y manejar las peticiones HTTP de toda la aplicación.

templates

El grupo de componentes **templates** agrupa todas las pantallas de la interfaz de usuario, incluyendo una pantalla layout que servirá como plantilla base para el diseño de todas las pantallas.

static

En este componente se incluyen aquellos elementos estáticos que se utilizan en la aplicación web. Por ejemplo, en la pantalla de documentación se podrá ver esta memoria, por lo que la memoria deberá ser almacenada en este módulo.

tests

Este componente incluye todas las clases de prueba creadas. Agrupa las pruebas unitarias, de integración y de sistemas.

base

En el módulo base, se almacenan los componentes principales del sistema. Estos elementos podrán ser objetos únicos para todo el sistema, como Dataset. O podrán ser interfaces, de las cuáles surgirán diversos tipos de objetos, como Executor o QuantumModel.

extended

Este módulo se divide en dos submódulos, **executors** y **quantum_models**. En executors, se agrupan los componentes encargados de la ejecución de los algoritmos, en concreto los simuladores y la API de IBM Quantum Experience.

En quantum_models se agrupan los modelos de los algoritmos de aprendizaje automático cuántico diseñados para este proyecto.

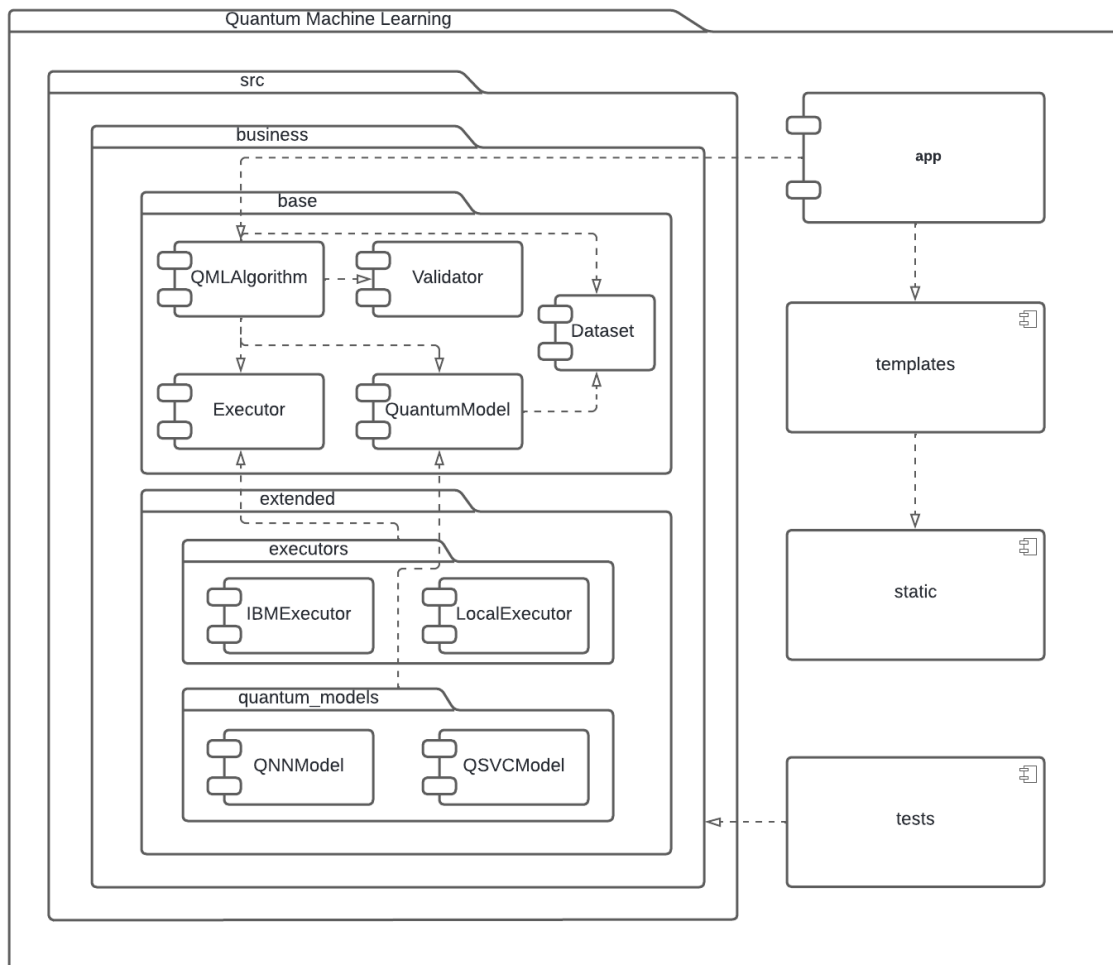


Figura 8.11: Diagrama de paquetes

8.3.2. Diseño de Comunicaciones entre Módulos

Para mostrar de forma clara de que forma se comunican unos módulos con otros, se ha creado un diagrama de componentes que puede verse en la Figura 8.13.

En este diagrama, se muestran los módulos principales donde se reúne la

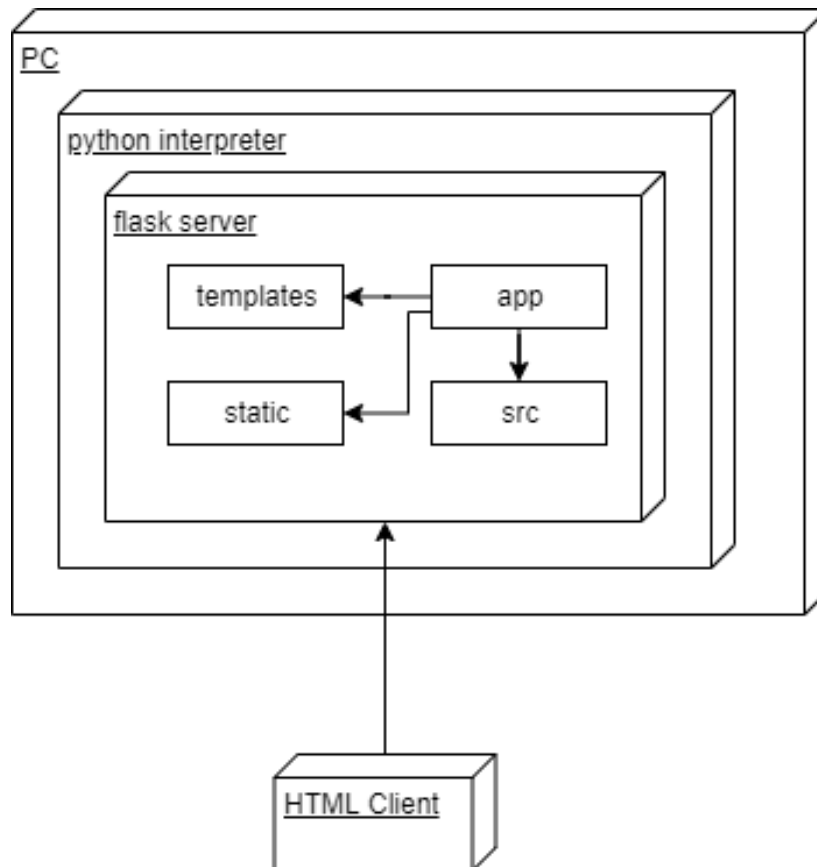


Figura 8.12: Diagrama de despliegue

mayoría de la lógica del sistema.

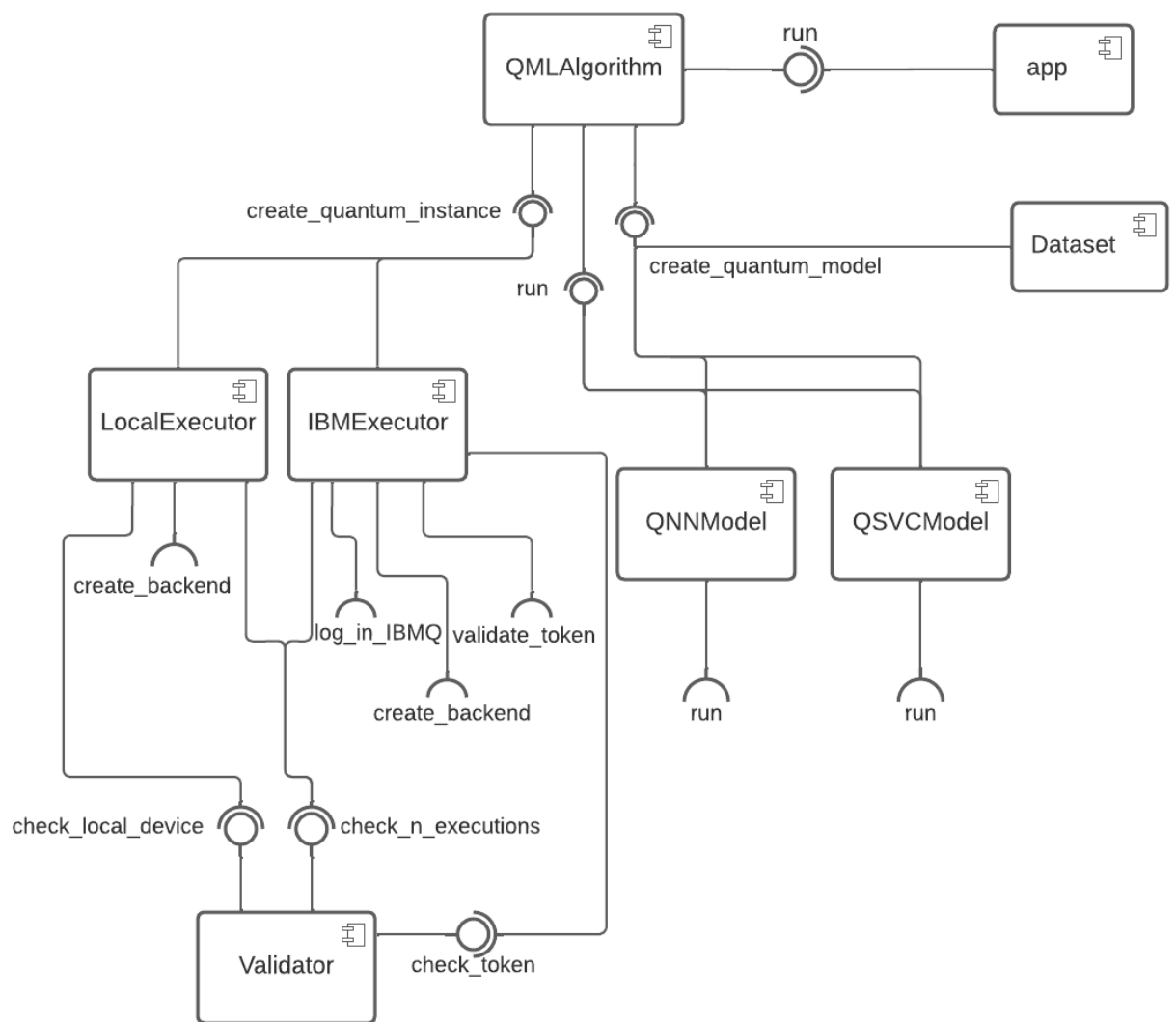


Figura 8.13: Diagrama de componentes



Figura 8.14: Diseño - Diagrama de Entidad-Relación

8.4. DISEÑO FÍSICO DE DATOS

8.4.1. Diagrama E-R

En este prototipo, no se usa ningún tipo de base de datos ni se almacena ningún tipo de información. Por ello, se ha decidido no incluir estos apartados. La única mención sería comentar la entidad que se almacena cuando el usuario ingresa su TOKEN de IBM. Si el usuario ingresa su token y este es validado, se almacena en disco, para que no sea necesario volver a ingresar el token cada vez que el usuario use la aplicación. Esta entidad se puede ver en la Figura 8.14.

8.5. ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS

8.5.1. Pruebas Unitarias

Para el desarrollo de las pruebas unitarias, como se ha comentado en el apartado 7.6.1 Pruebas unitarias, se han tenido en cuenta las situaciones de prueba mostradas en la tabla 8.1.

Tabla 8.1: Pruebas unitarias

Dataset	
select_dataset	<ul style="list-style-type: none"> ■ Se introduce el nombre de un conjunto de datos existente. ■ Se introduce el nombre de un conjunto de datos incorrecto.
define_data	<ul style="list-style-type: none"> ■ Se introduce el nombre de un conjunto de datos existente. ■ Se introduce el nombre de un conjunto de datos incorrecto. ■ Se introduce el nombre del conjunto de datos ad_hoc_data.
IBMExecutor	
is_authenticated	<ul style="list-style-type: none"> ■ No se dispone de cuenta en disco. ■ Se dispone de una cuenta en disco.

8.5.2. Pruebas de Integración

Para el desarrollo de las pruebas de integración, se han tenido en cuenta las situaciones de prueba mostradas en la tabla 8.2.

Tabla 8.2: Pruebas de integración

QMLAlgorithm	
create_quantum_instance	<ul style="list-style-type: none"> ■ Se crea un circuito cuántico en un simulador. ■ Se crea un circuito cuántico en un ordenador. ■ Se crea un circuito cuántico en un dispositivo incorrecto.
create_quantum_model	<ul style="list-style-type: none"> ■ Se crea un modelo del algoritmo QSVM. ■ Se crea un modelo del algoritmo QNN. ■ Se crea un modelo incorrecto.
run	<ul style="list-style-type: none"> ■ Se ejecuta un algoritmo en un simulador. ■ Se ejecuta un algoritmo en un ordenador.

8.5.3. Pruebas de Sistema

Para el desarrollo de las pruebas de sistema, se han tenido en cuenta las situaciones de prueba mostradas en la tabla 8.3.

Tabla 8.3: Pruebas de sistema

LocalExecutor	
create_backend	<ul style="list-style-type: none"> ■ Se crea un backend de un simulador Aer básico. ■ Se crea un backend de un simulador Aer diferente al básico. ■ Se crea un backend de un simulador incorrecto.
IBMExecutor	
validate_token	<ul style="list-style-type: none"> ■ El token a validar es correcto. ■ El token a validar es incorrecto.
log_in_IBMQ	<ul style="list-style-type: none"> ■ Se identifica y guarda la cuenta cuando no existe cuenta en disco. ■ Se carga la cuenta del disco cuando esta existe.
create_backend	<ul style="list-style-type: none"> ■ Se crea un backend de ibm con n^o de ejecuciones < 0 ■ Se crea un backend de ibm con n^o de ejecuciones > 20.000 ■ Se crea un backend de ibm con n^o de ejecuciones > 0 y < 20.000
QuantumModel	
run	<ul style="list-style-type: none"> ■ Se ejecuta el algoritmo QSVM en un simulador. ■ Se ejecuta el algoritmo QSVM en un ordenador cuántico. ■ Se ejecuta el algoritmo QNN en un simulador. ■ Se ejecuta el algoritmo QNN en un ordenador cuántico.

Capítulo 9

CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

CSI

9.1. PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONSTRUCCIÓN

9.1.1. Estándares y normas seguidos

Para los contenidos de este documento se ha seguido la Métrica V3 [23].

9.1.2. Lenguajes de programación

Lenguajes usados para la construcción del prototipo:

- Python 3.8
- Javascript (versión Node.js v14.16.0.)
- HTML5
- CSS3

9.1.3. Herramientas y programas usados para el desarrollo

Para el desarrollo del prototipo se han utilizado numerosas librerías o frameworks:

- Flask
- Qiskit
- Bootstrap
- pandas
- numpy
- matplotlib

Qiskit es un framework de Python que constituye el centro de este proyecto, ya que es el framework que nos permite crear circuitos cuánticos y ejecutarlos en un simulador o en un ordenador cuántico real.

Es un proyecto de código abierto que pertenece a IBM y pone a disposición de los usuarios de forma gratuita sus servicios mediante el sistema IBM Quantum Experience.

Programas:

Qiskit proporciona diferentes módulos que son usados en este proyecto con diferentes objetivos:

- Terra: este módulo contiene las herramientas y elementos para crear circuitos cuánticos, puertas lógicas, visualización del circuito y simuladores básicos, entre otras muchas cosas.

- Aer: este módulo contiene simuladores cuánticos, incluyendo simuladores de ruido y errores para asemejar el resultado a los ordenadores cuánticos reales.
- qiskit-machine-learning: este módulo contiene numerosos algoritmos de aprendizaje automático, así como circuitos cuánticos para implementarlos, conjuntos de datos, kernels cuánticos, y otras herramientas para la creación de sistemas inteligentes cuánticos.

Se han utilizados numerosos programas y páginas web a lo largo del proyecto para el desarrollo de este, con diferentes objetivos. Entre ellos, están:

Gestión del proyecto

- Microsoft Project 2022 (16.0.15225.20288)
- Microsoft Excel 2022 (18.2205.1091.0)
- Overleaf.com
- Notion 2.0.28
- Github
- GitKraken 8.4.0

Implementación del prototipo

- Pycharm 2022.1.2

Sistema sobre el que se ha creado

- Windows 10 professional
- Opera GX Browser (86.0.4363.70)

Diseño de interfaces/diagramas

- Lucidchart.com
- draw.io
- Balsamiq Wireframes 4.5.3

9.2. GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES Y PROCEDIMIENTOS

En esta sección se verá en profundidad la implementación de las clases más importantes que componen el sistema. Estas clases son **QMLAlgorithm**, **QuantumModel** y **Executor**, las cuales componen el núcleo de la funcionalidad del sistema.

9.2.1. QMLAlgorithm

La clase principal es QMLAlgorithm. En esta clase se instancian Dataset, QuantumModel y Executor para dar lugar a un algoritmo completo. En la tabla 9.1 se detallan los atributos de la clase y los métodos junto con sus entradas, salidas y una pequeña descripción.

Tabla 9.1: Descripción de diseño de QMLAlgorithm

QMLAlgorithm	
Descripción	
Es la encargada de formar la estructura principal del algoritmo con la configuración seleccionada por el usuario para su posterior ejecución.	
Atributos	
dataset: string	Es el nombre del dataset seleccionado por el usuario.
execution_type: string	Es el tipo de ejecución seleccionado por el usuario. Puede ser local o IBM.
device: string	Es el nombre del dispositivo seleccionado por el usuario en el caso de que se escoja ejecutar el sistema en local.
n_executions: int	Es número de veces que se va a ejecutar el algoritmo.
ml_model: string	Es tipo de algoritmo seleccionado por el usuario. Puede ser QSVM o QNN.
Métodos propuestos	
run: LLama al método <code>create_quantum_model</code> y ejecuta el algoritmo.	
create_quantum_instance(token: str) → BaseBackend, QuantumInstance: Crea el circuito cuántico y el entorno de ejecución del algoritmo (en local o en ibm). Devuelve una instancia del entorno, ya sea en local o ibm y una instancia del circuito con dicho entorno. En el caso de que <code>execution_type</code> sea local, se creará un simulador. En caso de que sea IBM, se validará el token pasado por parámetro y se creará la conexión a un ordenador cuántico. En caso de que <code>execution_type</code> tenga un valor inválido, se elevará una excepción.	
create_quantum_model(token: str) → QuantumModel: Llama al método <code>create_quantum_instance</code> y crea el modelo de aprendizaje con el conjunto de datos seleccionado en el atributo <code>m_model</code> . En el caso de que el valor de este sea inválido, se produce una excepción.	

9.2.2. QuantumModel

Esta clase se encargará de crear el modelo de aprendizaje automático seleccionado, junto con el conjunto de datos y la instancia del backend deseado. En la tabla 9.2, se detallan los atributos de la clase y los métodos junto con sus entradas, salidas y una pequeña descripción.

Tabla 9.2: Descripción de diseño de QuantumModel

QuantumModel		
Descripción		
Es la encargada de dar estructura a los modelos de aprendizaje automático, QSVM y QNN, seleccionados en este proyecto.		
Atributos		
dataset:	Dataset	Es conjunto de datos, ya procesado, seleccionado para que el modelo entrene y haga predicciones.
quantum_instance:	QuantumInstance	Quantum Instance es una clase de Qiskit que dispone de un backend de Qiskit Terra, así como una configuración para la ejecución de circuitos. Cuando se proporciona a un algoritmo, el algoritmo ejecutará los circuitos haciendo uso de esta instancia.
backend:	Base-Backend	Es la clase base, proporcionada por Qiskit, de los backends.
Métodos propuestos		
run()	→ dict:	Ejecuta el modelo de aprendizaje cuántico. Entrena sobre el dataset en el backend seleccionado y devuelve los resultados obtenidos y el grado de exactitud alcanzado.

9.2.3. Executor

Esta clase se encargará de crear backend deseado donde se ejecutará el modelo de aprendizaje. En la tabla 9.3, se detallan los atributos de la clase y los métodos junto con sus entradas, salidas y una pequeña descripción.

Tabla 9.3: Descripción de diseño de Executor

Executor	
Descripción	
Es la encargada de dar estructura a las clases de backend donde se ejecutarán los algoritmos, tanto simuladores como ordenadores cuánticos.	
Atributos	
device: string	Es el nombre que define al backend. En el caso de que se elija local, será el nombre del simulador, si se elige IBM, será el TOKEN del usuario.
n_executions: int	Es el número de veces que correrá el modelo en el backend, sin contar con la retroalimentación necesaria que se realiza en el proceso de entrenamiento del modelo.
Métodos propuestos	
create_backend() → BaseBackend, QuantumInstance : Obtiene el backend del proveedor necesario, instanciado en las subclases, y devuelve el backend creado y una instancia de QuantumInstance con la configuración de la ejecución a realizar.	

9.3. EJECUCIÓN DE LAS PRUEBAS

En esta sección se mostrarán los resultados obtenidos de la ejecución de todas las pruebas realizadas.

Para la creación y ejecución de las pruebas se ha utilizado el paquete `pytest` de Python. Para reemplazar el comportamiento de ciertas funciones se ha usado `pytest-mock`. Y para calcular el porcentaje de cobertura de las pruebas se utiliza el paquete `Coverage`, que viene integrado en el IDE `PyCharm 2022.1.2`.

Se han creado un total de 23 tests, entre ellos 5 tests son unitarios para comprobar el correcto funcionamiento de algunos métodos que no tienen contacto con ningún sistema exterior. El resto de tests se clasificarían como pruebas de integración y de sistema, ya que comprueban el correcto funcionamiento de la ejecución de los algoritmos, tanto en un simulador como en un ordenador cuántico real.

También se han realizado pruebas manuales sobre la interfaz gráfica. Para estas pruebas, se han probado todos los casos de uso definidos, los cuales se pueden ver en la Figura 7.1, además de comprobar que se obtiene la pantalla de error cuando se introducen valores erróneos. La pantalla de error que se muestra al usuario se puede ver en la Figura 9.3.

Además de comprobar el correcto manejo de errores y valores inválidos, principalmente se ha probado que el sistema puede ejecutarse tanto en un simulador como en un ordenador cuántico con ambos algoritmos.

Para ello, primero se ha escogido el algoritmo `QSVM`, seleccionando un conjunto de datos de entre los listados al azar. Este conjunto de datos posteriormente en dividido por el sistema en cuatro subconjuntos: `X_train`, `y_train`, `X_test` e `y_test`.

`X_train` e `y_train` conforman los datos de entrenamiento. `X_train` contiene todas las variables que se usarán para entrenar el modelo, mientras que `y_train` contiene las labels de dichas variables, es decir, la clase a la que pertenecen.

`X_test` e `y_test` conforman los datos de test. `X_test` contiene todas las variables que se usarán para hacer la predicción, mientras que `y_test` contiene las clases predichas por el modelo, la cual se usará para calcular el `accuracy`.

En ambos casos de ejecución en local y en IBM el proceso de entrenamiento del modelo es el mismo, por lo que a nivel de código el entrenamiento es independiente de la herramienta a utilizar para la ejecución.

Para el mapeo de los datos clásicos al circuito cuántico el `feature map` utilizado en ambos algoritmos es `ZZFeatureMap`, proporcionado por la librería `Qiskit`.

`ZZFeatureMap` es un circuito evolutivo de Pauli Z de segunda orden. Tiene como argumentos:

- `feature_dimensions`: es la dimensión de los datos clásicos. Es igual al número

de qubits.

- `reps`: es el número de veces que el circuito del feature map se repetirá.
- `data_map_func`: es la función que codifica los datos clásicos.
- `entanglement`: decide el tipo de entrelazamiento que se generará entre los qubits.

Existen otros diversos feature maps que tienen su propia forma de codificar los datos. Algunos de ellos son `ZFeatureMap` y `PauliFeatureMap` [24].

La ejecución, en cualquier caso que se seleccione, se llevará a cabo utilizando únicamente **2 qubits**, ya que la dimensión de los datos es igual a 2.

Una vez realizada la predicción, por ejemplo en local, obtendremos la salida correspondiente como se ve en la Figura 9.1. A la izquierda vemos el parámetro optimizado, que es θ , junto con el porcentaje de exactitud y los valores predecidos. En la parte de la derecha se muestra una gráfica con el conjunto de datos usado y una representación del circuito ejecutado.

Debido a los largos tiempos de espera para realizar una ejecución en un ordenador cuántico, no se han obtenido imágenes de los resultados obtenidos, pero sí que se ha llegado a ejecutarse correctamente obteniendo una accuracy del 80 %.

En cuanto al algoritmo QNN, el proceso seguido es el mismo que con QSVM, y se obtuvo en local la salida correspondiente a la que se ve en la Figura 9.2. A la izquierda vemos el porcentaje de exactitud y los valores predecidos. En la parte de la derecha se muestra una gráfica con la evolución de la función de pérdida por cada iteración realizada y una representación del circuito ejecutado.

Debido a que los tiempos de espera para realizar una ejecución en un ordenador cuántico también eran muy largos, no se han obtenido imágenes de los resultados obtenidos. Se ha llegado a ejecutarse correctamente obteniendo una accuracy del 60 %.

Salida

Trainable parameters: θ , [' $\theta[0]$ ']

Porcentaje de exactitud: 80.0%

Valores reales: [0 0 0 0 0 1 1 1 1 1]

Valores predecidos: [0 1 0 0 0 1 1 1 1 0]

}

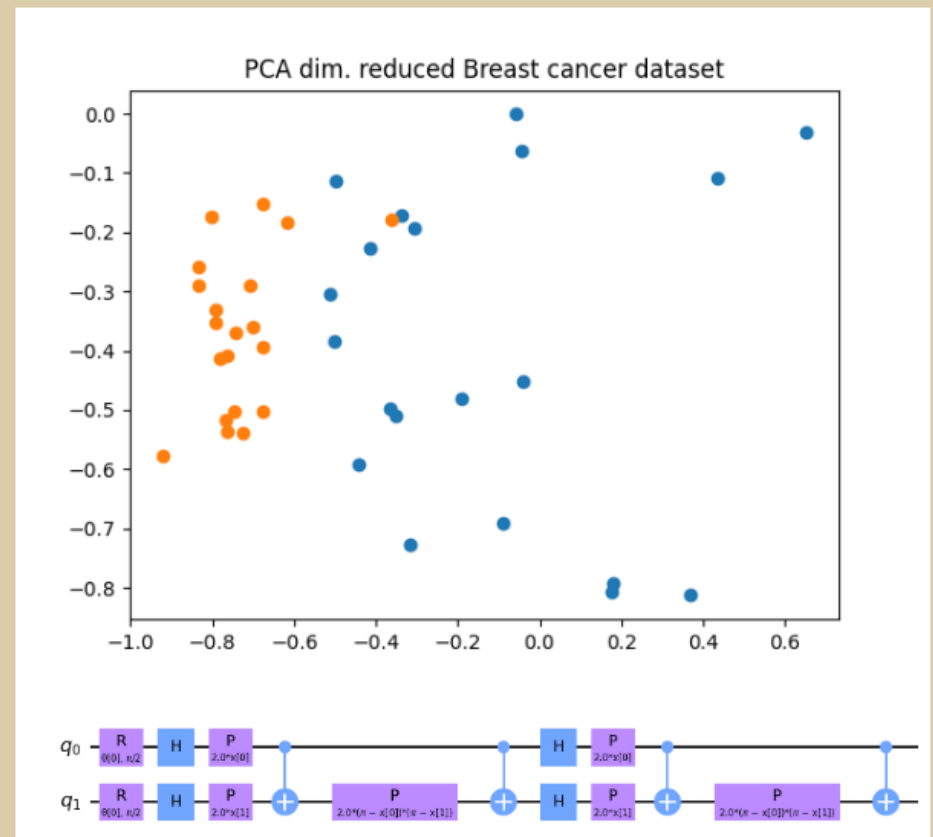


Figura 9.1: Ejemplo de ejecución en local. QSVM

Salida

```

Porcentaje de exactitud: 50.0%

Valores reales: [0 0 0 0 0 1 1 1 1 1]
Valores predecidos: ['0 1 1 1 1 1 1 1 1 1']
    
```

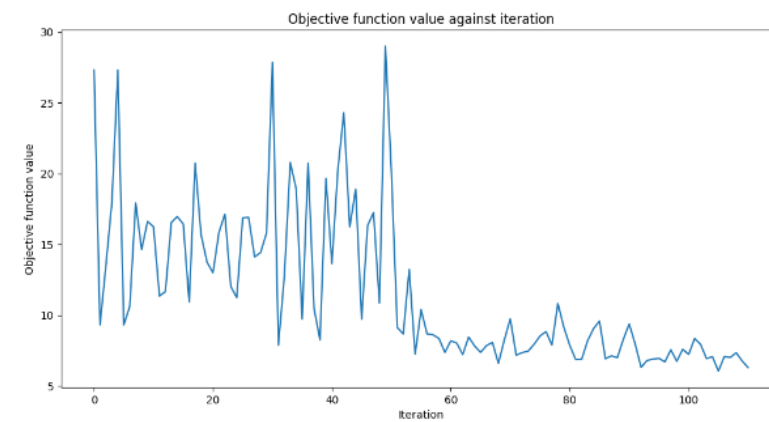


Figura 9.2: Ejemplo de ejecución en local. QNN



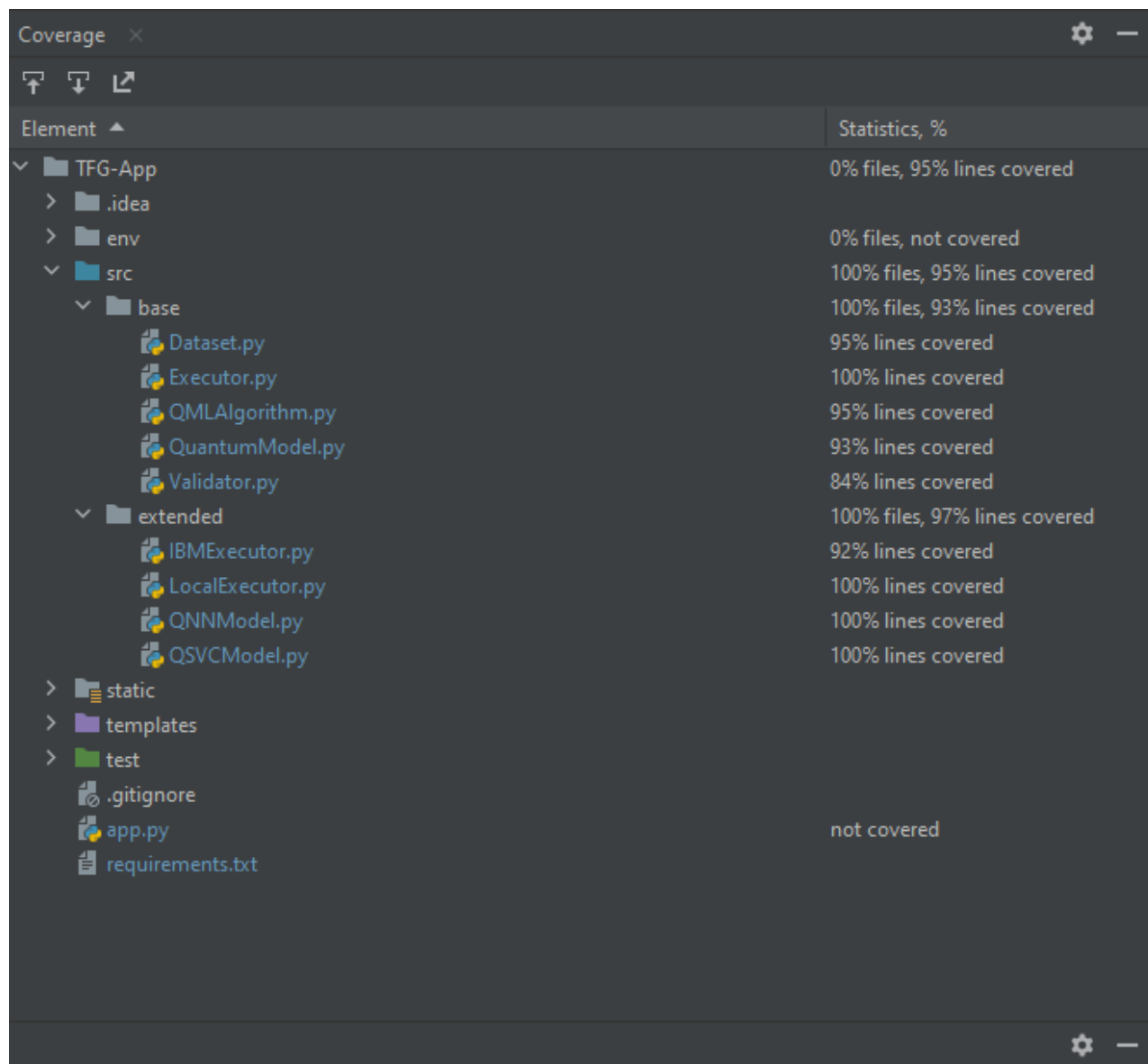
Figura 9.3: Pantalla de error

Durante la ejecución de las pruebas, se encontraron errores en la ejecución con simuladores, ya que algunos de ellos no soportaban las puertas cuánticas utilizadas en nuestro circuito, en concreto la puerta R_y , por lo que estos simuladores fueron descartados.

El objetivo de estas pruebas era asegurarse de que las partes fundamentales funcionan como se esperaba. También se quería abarcar una cobertura alta, por lo que se estableció un mínimo de 90 % de cobertura.

Para calcular el porcentaje de cobertura de pruebas se ha utilizado el paquete **coverage**. Con la ejecución de las pruebas haciendo uso de este paquete, el resultado obtenido fue el mostrado en la Figura 9.4.

Como se puede ver en la imagen, en la carpeta src, donde se encuentra todo el código del sistema, se ha logrado alcanzar un 95 % de cobertura, con un 100 % de archivos cubiertos.



The screenshot shows the Coverage tool window with a tree view of the project structure and a table of coverage statistics. The project is TFG-App, and the coverage is 95% for lines. The table lists the following elements and their coverage:

Element	Statistics, %
TFG-App	0% files, 95% lines covered
.idea	
env	0% files, not covered
src	100% files, 95% lines covered
base	100% files, 93% lines covered
Dataset.py	95% lines covered
Executor.py	100% lines covered
QMLAlgorithm.py	95% lines covered
QuantumModel.py	93% lines covered
Validator.py	84% lines covered
extended	100% files, 97% lines covered
IBMExecutor.py	92% lines covered
LocalExecutor.py	100% lines covered
QNNModel.py	100% lines covered
QSVCModel.py	100% lines covered
static	
templates	
test	
.gitignore	
app.py	not covered
requirements.txt	

Figura 9.4: Cobertura de pruebas

9.4. ELABORACIÓN DE LOS MANUALES DE USUARIO

9.4.1. Manual de Instalación

El sistema cuenta con una serie de dependencias que deben ser instaladas antes de la ejecución del mismo.

Tan sólo tendremos que instalar de forma manual **Python 3.8**, el resto de dependencias aparecen en un archivo **requirements.txt** que se usará más tarde para la instalación automática de las siguientes dependencias:

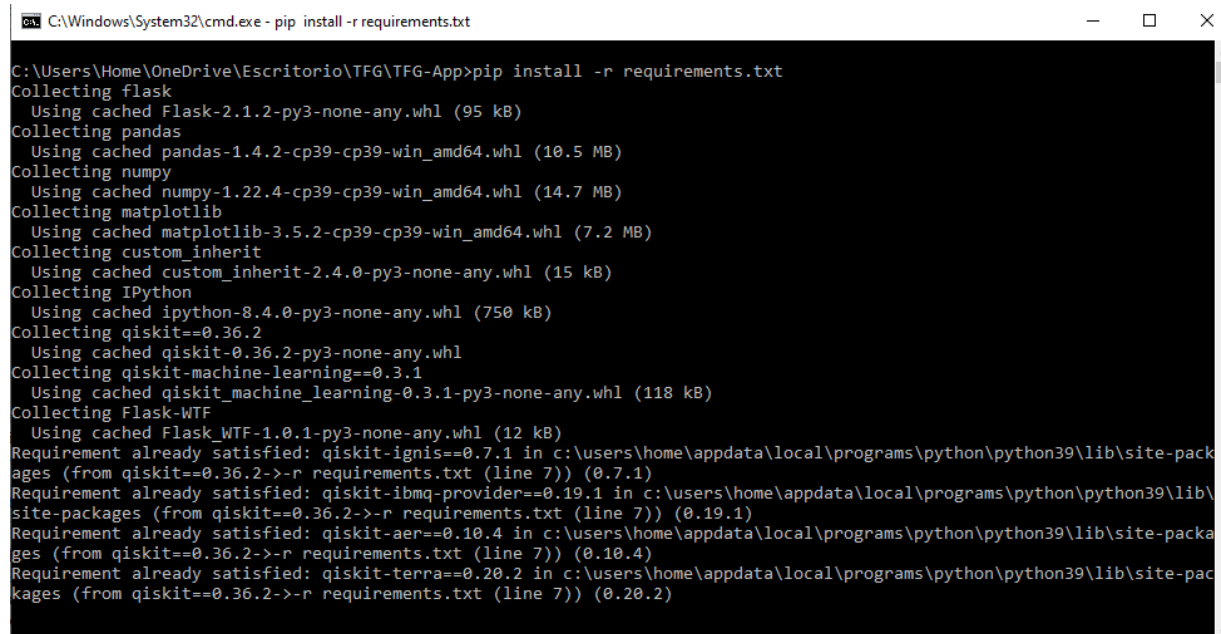
- flask
- pandas
- numpy
- matplotlib
- custom_inherit
- IPython
- qiskit
- qiskit-machine-learning
- Flask-WTF
- pytest
- pytest-mock
- coverage
- pylatexenc

Para instalar Python 3.8, es necesario descargarlo de su página oficial desde la siguiente url: <https://www.python.org/downloads/release/python-380/>. Una vez descargado, ejecutamos el instalador y seguimos las instrucciones hasta completar la instalación. Esta instalación también incluye la herramienta **pip**, que es un instalador de paquetes que nos servirá para instalar el resto de dependencias.

Una vez tengamos Python instalado, podremos adquirir el resto de paquetes. Para ello, accedemos al directorio del proyecto, donde encontraremos el archivo de texto **requirements.txt**.

Hay dos opciones para instalar y ejecutar el sistema: en local o en un entorno virtual.

En local:



```
C:\Windows\System32\cmd.exe - pip install -r requirements.txt

C:\Users\Home\OneDrive\Escritorio\TFG\TFG-App>pip install -r requirements.txt
Collecting flask
  Using cached Flask-2.1.2-py3-none-any.whl (95 kB)
Collecting pandas
  Using cached pandas-1.4.2-cp39-cp39-win_amd64.whl (10.5 MB)
Collecting numpy
  Using cached numpy-1.22.4-cp39-cp39-win_amd64.whl (14.7 MB)
Collecting matplotlib
  Using cached matplotlib-3.5.2-cp39-cp39-win_amd64.whl (7.2 MB)
Collecting custom_inherit
  Using cached custom_inherit-2.4.0-py3-none-any.whl (15 kB)
Collecting IPython
  Using cached ipython-8.4.0-py3-none-any.whl (750 kB)
Collecting qiskit==0.36.2
  Using cached qiskit-0.36.2-py3-none-any.whl
Collecting qiskit-machine-learning==0.3.1
  Using cached qiskit_machine_learning-0.3.1-py3-none-any.whl (118 kB)
Collecting Flask-WTF
  Using cached Flask_WTF-1.0.1-py3-none-any.whl (12 kB)
Requirement already satisfied: qiskit-ignis==0.7.1 in c:\users\home\appdata\local\programs\python\python39\lib\site-pack
ages (from qiskit==0.36.2->-r requirements.txt (line 7)) (0.7.1)
Requirement already satisfied: qiskit-ibmq-provider==0.19.1 in c:\users\home\appdata\local\programs\python\python39\lib\
site-packages (from qiskit==0.36.2->-r requirements.txt (line 7)) (0.19.1)
Requirement already satisfied: qiskit-aer==0.10.4 in c:\users\home\appdata\local\programs\python\python39\lib\site-packa
ges (from qiskit==0.36.2->-r requirements.txt (line 7)) (0.10.4)
Requirement already satisfied: qiskit-terra==0.20.2 in c:\users\home\appdata\local\programs\python\python39\lib\site-pac
kages (from qiskit==0.36.2->-r requirements.txt (line 7)) (0.20.2)
```

Figura 9.5: Instalación de dependencias

Abrimos una línea de comandos con permisos de administrador en ese mismo directorio y ejecutamos lo siguiente:

pip install -r requirements.txt

En la terminal saldrá algo similar a la Figura 9.5. Finalizada la instalación y si todo es correcto, ya podemos empezar a usar el prototipo.

En un entorno virtual:

Esta opción es preferible por si no queremos instalar las dependencias directamente en nuestro ordenador.

En la carpeta del proyecto ya está creado un entorno virtual. Se encuentra dentro de la carpeta **env**.

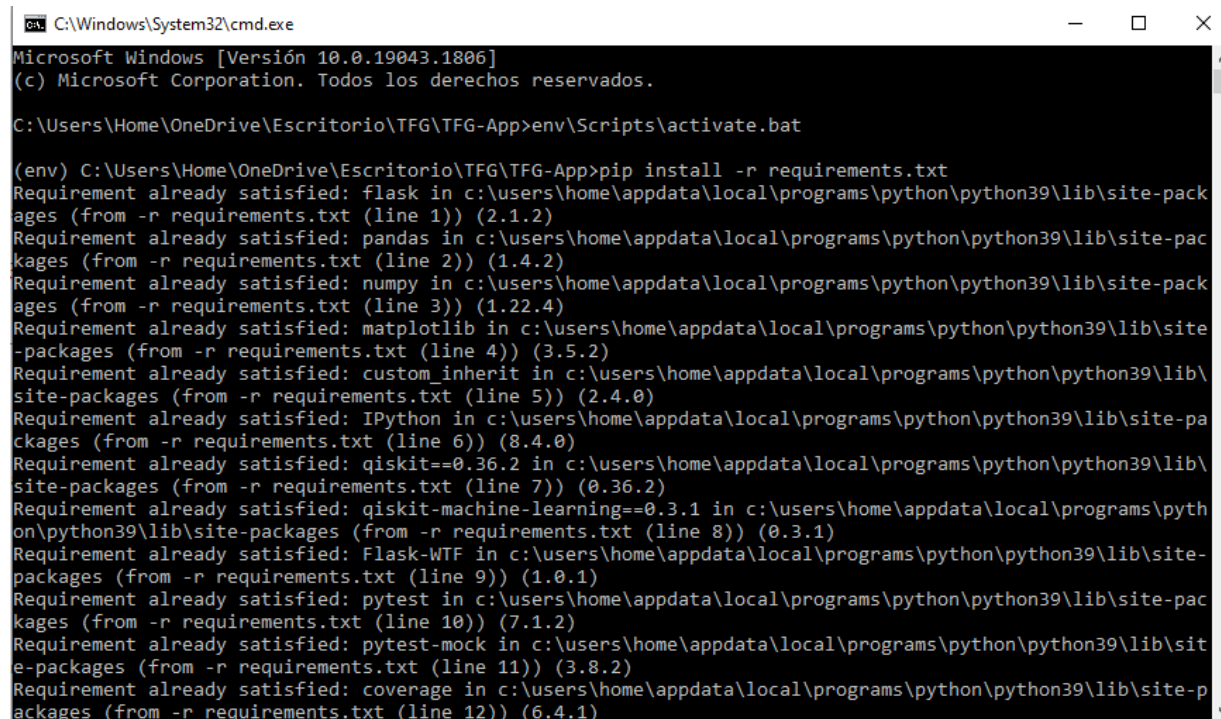
Abrimos una línea de comandos con permisos de administrador en el directorio principal del proyecto y ejecutamos lo siguiente:

env\Scripts\activate.bat

Con este comando se iniciará el entorno virtual, entonces para instalar las dependencias ejecutamos lo siguiente:

pip install -r requirements.txt

En la terminal saldrá algo similar a la Figura 9.6. Finalizada la instalación y si todo es correcto, ya podemos empezar a usar el prototipo desde el entorno virtual.



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1806]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Home\OneDrive\Escritorio\TFG\TFG-App>env\Scripts\activate.bat

(env) C:\Users\Home\OneDrive\Escritorio\TFG\TFG-App>pip install -r requirements.txt
Requirement already satisfied: flask in c:\users\home\appdata\local\programs\python\python39\lib\site-pack
ages (from -r requirements.txt (line 1)) (2.1.2)
Requirement already satisfied: pandas in c:\users\home\appdata\local\programs\python\python39\lib\site-pac
kages (from -r requirements.txt (line 2)) (1.4.2)
Requirement already satisfied: numpy in c:\users\home\appdata\local\programs\python\python39\lib\site-pack
ages (from -r requirements.txt (line 3)) (1.22.4)
Requirement already satisfied: matplotlib in c:\users\home\appdata\local\programs\python\python39\lib\site-
packages (from -r requirements.txt (line 4)) (3.5.2)
Requirement already satisfied: custom_inherit in c:\users\home\appdata\local\programs\python\python39\lib\
site-packages (from -r requirements.txt (line 5)) (2.4.0)
Requirement already satisfied: IPython in c:\users\home\appdata\local\programs\python\python39\lib\site-pa
ckages (from -r requirements.txt (line 6)) (8.4.0)
Requirement already satisfied: qiskit==0.36.2 in c:\users\home\appdata\local\programs\python\python39\lib\
site-packages (from -r requirements.txt (line 7)) (0.36.2)
Requirement already satisfied: qiskit-machine-learning==0.3.1 in c:\users\home\appdata\local\programs\pyth
on\python39\lib\site-packages (from -r requirements.txt (line 8)) (0.3.1)
Requirement already satisfied: Flask-WTF in c:\users\home\appdata\local\programs\python\python39\lib\site-
packages (from -r requirements.txt (line 9)) (1.0.1)
Requirement already satisfied: pytest in c:\users\home\appdata\local\programs\python\python39\lib\site-pac
kages (from -r requirements.txt (line 10)) (7.1.2)
Requirement already satisfied: pytest-mock in c:\users\home\appdata\local\programs\python\python39\lib\sit
e-packages (from -r requirements.txt (line 11)) (3.8.2)
Requirement already satisfied: coverage in c:\users\home\appdata\local\programs\python\python39\lib\site-p
ackages (from -r requirements.txt (line 12)) (6.4.1)

```

Figura 9.6: Instalación de dependencias en el entorno virtual

9.4.2. Manual de Ejecución

Una vez que hayamos completado la instalación como se explica en el 9.4.1 Manual de Instalación, podremos ejecutar el prototipo y comenzar a usarlo.

Ya que el entorno se ha preparado haciendo uso de Flask, deberemos tenerlo instalado y usar este para ejecutar el programa en local.

Para ello, accedemos al directorio del proyecto, donde encontraremos el archivo **app.py**. Este archivo será el script principal dónde se encuentra declarada la instancia de Flask.

Si hemos realizado la instalación en local entonces:

En este directorio abrimos una línea de comandos y ejecutamos con permisos de administrador:

flask run

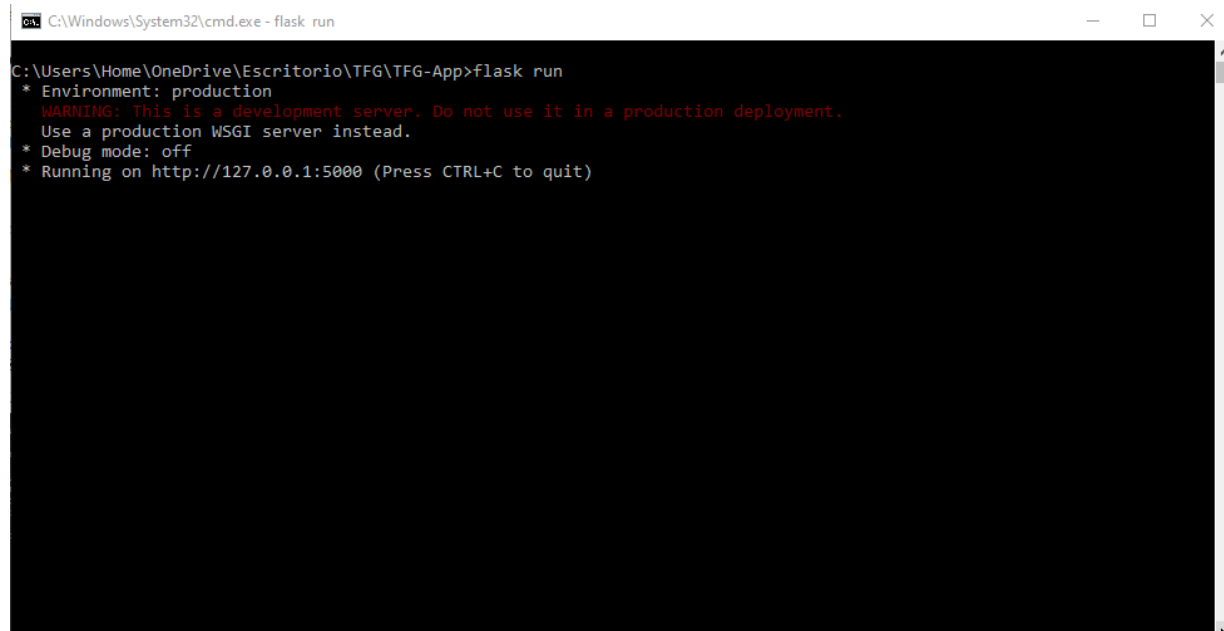
Si hemos realizado la instalación en un entorno virtual entonces:

En este directorio abrimos una línea de comandos y iniciamos el entorno virtual con permisos de administrador:

env\Scripts\activate.bat

Después, iniciamos la aplicación ejecutando:

flask run



```
C:\Windows\System32\cmd.exe - flask run
C:\Users\Home\OneDrive\Escritorio\TFG\TFG-App>flask run
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

Figura 9.7: Ejecución del entorno Flask

Podremos ver algo similar a lo mostrado en la Figura 9.5.

Esto hará que el programa se ejecute en local. Para acceder a él, simplemente ingresamos la url: <http://localhost:5000/> en cualquier navegador de nuestra preferencia y nos aparecerá la pantalla principal de la aplicación, que debería tener el aspecto mostrado en la Figura 9.8.

9.4.3. Manual de Usuario

Para hacer uso de la aplicación web, primero seguiremos los pasos definidos en el Manual de Instalación y en el Manual de Ejecución.

Nos dirigimos a la url: <http://localhost:5000/> desde el navegador.

Nos encontraremos en la página principal de la aplicación. Aquí, tendremos dos botones para escoger entre ejecutar el algoritmo QSVM o el QNN. También, en la parte superior de la página, tenemos la barra de navegación desde la cual podemos volver a este mismo punto pulsando Home, ir a la página de documentación o ir a la página de contacto.

Tanto si escogemos ejecutar QNN como QSVM, en la pantalla de configuración de los mismos tenemos idénticos parámetros a introducir.

En el caso de que se selecciona ejecutar el algoritmo en local, no será necesario el uso de ningún TOKEN, pero se podrá escoger un simulador de los listados.

En el caso de que se selecciona ejecutar el algoritmo en IBM, el usuario deberá disponer de un TOKEN de autenticación de IBM Quantum Experience. Para

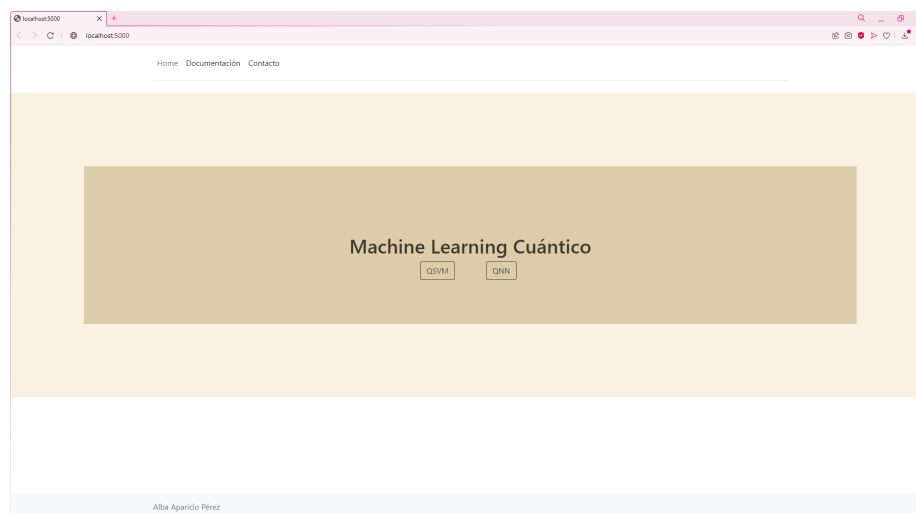


Figura 9.8: Pantalla principal

obtenerlo deberá acceder a <https://quantum-computing.ibm.com>. En esta página se creará una cuenta personal, y una vez creada, en el Dashboard se puede ver el valor de nuestro token, como se muestra en la Figura 9.9.

Este TOKEN deberá ser válido o, en caso contrario, se producirá un error en la aplicación que no nos permitirá ejecutar el algoritmo.

Una vez introducido el TOKEN correcto y pulsado el botón de siguiente, el algoritmo se ejecutará con todos los parámetros que hemos establecido. Es posible que este proceso tarde incluso unos minutos si los ordenadores cuánticos están ocupados en ese momento.

Y por último, una vez finalizada la ejecución, aparecerá la pantalla de salida con toda la información acerca de la ejecución.

9.4.4. Manual del Programador

Este prototipo ha sido diseñado de forma que puede ser ampliado fácilmente en el caso de que queramos incluir otros tipos de algoritmos de aprendizaje automático cuántico, incluyendo también algoritmos enfocados a problemas de regresión en vez de clasificación.

Para añadir nuevos algoritmos, crearemos una nueva clase que herede de la superclase **QuantumModel** existente en el proyecto y en el método `run()` implementaremos la lógica necesaria de este nuevo modelo. También se deberá modificar la clase **QMLAlgorithm** para que tenga en cuenta este nuevo algoritmo.

En cuanto a la interfaz de usuario, como se hace uso de un template, simplemente sería crear una nueva pantalla siguiendo el estilo del resto de pantallas.

Se puede hacer una ampliación similar con los sistemas de ejecución. Ya que

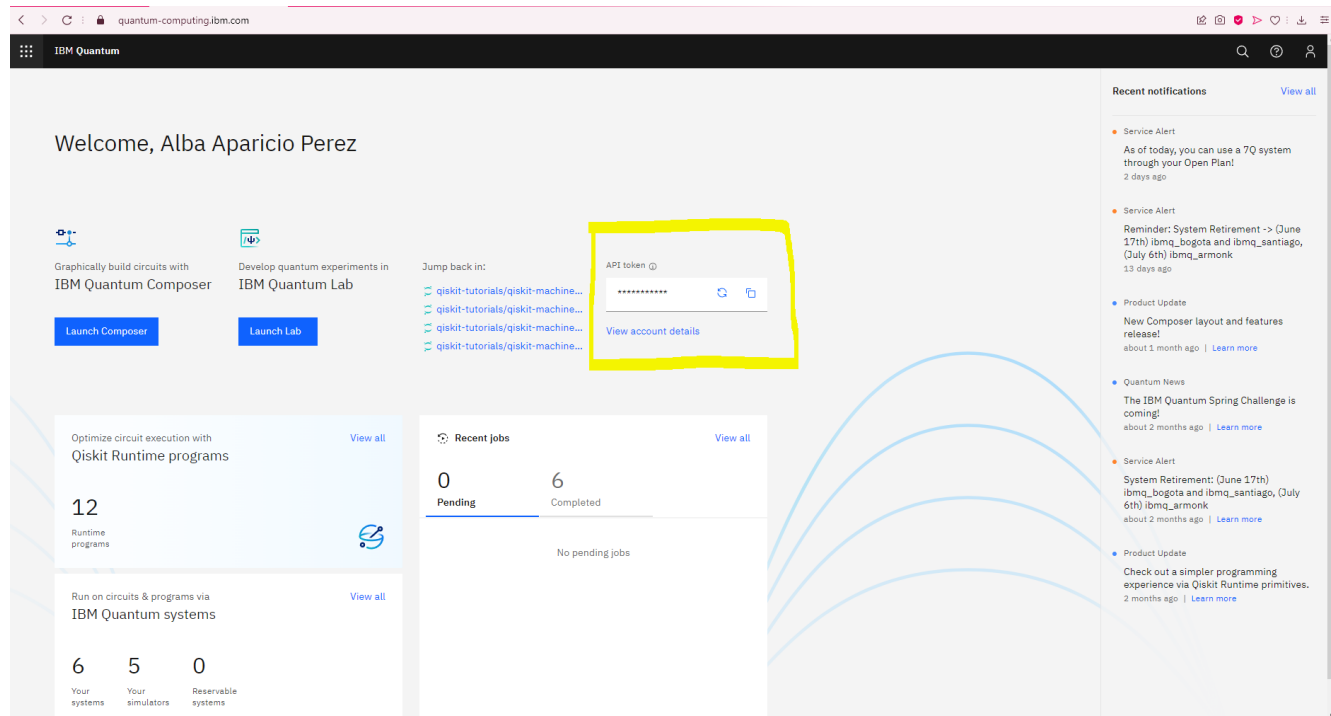


Figura 9.9: Obtención del TOKEN de IBM

el sistema dispone de la interfaz **Executor**, podemos implementar nuevas formas de ejecutar nuestros algoritmos, como, por ejemplo, usar otros proveedores de ordenadores cuánticos. El procedimiento sería el mismo que con los algoritmos. Se crea una nueva clase que herede de **Executor** y se añade dicha opción en la clase **QMLAlgorithm** para que tenga en cuenta este nuevo proveedor.

Capítulo 10

CONCLUSIONES Y AMPLIACIONES

10.1. CONCLUSIONES

A día de hoy, el paradigma de la computación cuántica está evolucionando rápidamente aunque aún se encuentra en sus comienzos y, por lo tanto, todavía no es útil para la gran mayoría de aplicaciones prácticas, y mucho menos para la comercialización [25].

Con esta evolución, el número de qubits cada vez será mayor.

Esto por lo tanto afectará al machine learning, ya que cada año se obtienen más y más datos que de alguna manera hay que procesar. Ahora, tenemos móviles inteligentes, televisiones e incluso coches que producen millones de datos, y si estos fueran usados para el entrenamiento de algoritmos de inteligencia artificial cuántica, la tecnología podría escalar más rápidamente. En esta sociedad donde el Big Data ya forma parte de nuestra vida, la aplicación de nuevos métodos cuánticos en áreas de investigación puede tener un gran impacto en nuestra vida e incluso en la economía mundial.

También hay que tener en cuenta que el paradigma cuántico es bastante complejo de entender y no es tarea fácil crear algoritmos que sean capaces de resolver los problemas de machine learning mejor que un ordenador convencional.

Además, el desarrollo de procesadores cuánticos se complica más cuanto mayor sea el número de qubits debido a las condiciones necesarias que se tienen que cumplir para evitar el ruido en los circuitos.

En este trabajo se ha implementado una parte muy sencilla de lo que tiene por ofrecer la computación cuántica, pero me ha servido para aprender más sobre este paradigma y reforzar las bases con las que ya contaba. Sin duda, también he aprendido mucho acerca del machine learning ya que era prácticamente desconocido para mí. Desde mi punto de vista creo, que es el quantum machine learning es una área de investigación bastante complicada de abordar, no sólo por su dificultad si no también por la escasa información que se tiene acerca de este tema, concretamente en español.

En definitiva, creo firmemente que la interacción entre la computación cuántica y el aprendizaje automático supondrá un gran avance para ambas ramas y será un campo de gran importancia. Pero aún queda mucho por desarrollar y errores que solucionar, sobretodo en el paradigma cuántico, lo que llevará bastante tiempo si queremos asentar unas bases prácticas y no sólo con estudios teóricos.

10.2. AMPLIACIONES

Este proyecto se centró principalmente en el desarrollo de algoritmos de clasificación binaria, ya que incluir otros tipos de problemas sería imposible por falta de tiempo.

Me gustaría haber implementado los algoritmos enfocados tanto a clasificación binaria como a multiclase, especialmente usando conjuntos de datos mucho más complejos, con miles de ejemplos, para ver si el porcentaje de exactitud se vería afectado.

En cuanto a la aplicación web, me habría gustado desplegarla en un servidor en la nube para que fuese accesible para todo el mundo. Se manejó esta posibilidad al poco de empezar el proyecto, pero se descartó ya que suponía implementar una gestión de usuarios y eso requería demasiado tiempo. De todas formas, el entorno de Google Cloud ofrece servicios para ello de forma gratuita y sería relativamente fácil desplegar la aplicación, por lo que es una ampliación bastante factible.

Finalmente, y como ampliación más esperada, me habría gustado tener a mi disposición ordenadores cuánticos con mucha mayor potencia que tan sólo los 7 qubits que ofrece IBM Quantum Experience. Quizás en unos años pueda disponer de mejor hardware y así poder hacer comparativas en cuanto a eficacia entre los diferentes paradigmas. O, incluso, incorporar otros dispositivos, de diferentes proveedores.

ANEXOS

PLAN DE GESTIÓN DE RIESGOS

Todo proyecto tiene riesgos asociados. El plan de gestión de riesgos se encarga de ofrecer una respuesta ante dichos potenciales riesgos, de forma que estos se puedan asumir o incluso transformar en riesgos positivos (oportunidades) que permitan potenciar los beneficios. Para ello se establecen unos pasos a seguir que permitan identificar, planificar, controlar y, finalmente, gestionar los riesgos. Para la realización de este plan se ha seguido la metodología PMBOK[26], en la que se encuentran las fases en la gestión de riesgos mostradas a continuación.

Planificación de la gestión de riesgos

Durante la planificación de la gestión de riesgos se define cómo realizar las actividades de gestión de riesgos de un proyecto.

Una planificación cuidadosa y explícita mejora la probabilidad de éxito de los otros procesos de gestión de riesgos. La planificación también es importante para proporcionar los recursos y el tiempo suficientes para las actividades de gestión de riesgos y para establecer una base acordada para la evaluación de riesgos. Tras seguir las fases descritas anteriormente, se actuará sobre aquellos riesgos que tengan un impacto mayor del 30 %, siguiendo el plan de respuesta que se describe más adelante.

Identificación de riesgos

Esta fase consiste en una identificación de posibles riesgos que puedan llegar a ocurrir a lo largo del proyecto. Estos riesgos pueden ser negativos o positivos y provienen de distintos tipos de categorías descritas en el PMBOK que se pueden ver en la Figura 10.1.

La gran mayoría de riesgos se identifican al inicio del proyecto, pero también es cierto que algunos otros han aparecido en el transcurso del mismo. Esto ocurre porque los riesgos son evolutivos, pudiendo llegar a aparecer nuevos o desaparecer los existentes a medida que avanza el ciclo de vida del proyecto. La identificación de los riesgos se realiza mediante el uso de distintas técnicas, siendo las utilizadas en este proyecto:

- **Tormenta de ideas (brainstorming):** Obtención de un listado de riesgos tras evaluar la situación del proyecto.
- **Análisis de listas de control:** Se utilizan listas de control de elementos de riesgo (listas de comprobación). Esta lista contiene preguntas obtenidas a lo largo de la experiencia que permiten determinar riesgos frecuentes.

Análisis de riesgos

Durante esta fase se calcula la probabilidad de que ocurra un riesgo y el impacto que podría llegar a ocasionar, dividido en diferentes partes del proyecto.

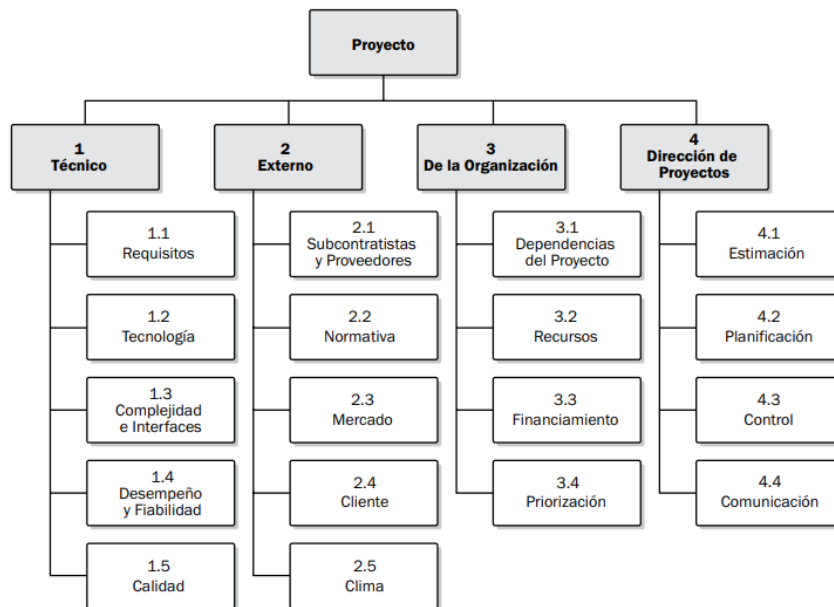


Figura 10.1: Clasificación de riesgos

Análisis cualitativo

En este análisis se priorizan los riesgos para realizar otros análisis posteriores, teniendo en cuenta la probabilidad de ocurrencia y el impacto que presentan.

Este tipo de análisis permite diferenciar los problemas más importantes a los que se enfrenta el proyecto.

Análisis cuantitativo

Este análisis consiste en obtener el impacto de los riesgos de manera numérica a partir de los datos obtenidos en el análisis cualitativo.

Para ello, se examinan los riesgos en detalle y se priorizan según su probabilidad de ocurrencia utilizando la matriz de impacto que se puede ver en la Figura 10.2.

Plan de respuesta a riesgos

Mediante el plan de respuesta a riesgos se desarrollan opciones para mejorar las oportunidades y reducir las amenazas del proyecto. La respuesta ofrecida antes cada riesgo es diferente y debe adaptarse a este, ajustándose a la relación coste/importancia adecuada.

Existen 4 estrategias de respuesta a riesgos que se pueden seguir:

- Eliminar el riesgo: No se expone al riesgo, eliminando los factores que lo producen.

Matriz de Probabilidad e Impacto										
Probabilidad	Amenazas					Oportunidades				
0,90	0,05	0,09	0,18	0,36	0,72	0,72	0,36	0,18	0,09	0,05
0,70	0,04	0,07	0,14	0,28	0,56	0,56	0,28	0,14	0,07	0,04
0,50	0,03	0,05	0,10	0,20	0,40	0,40	0,20	0,10	0,05	0,03
0,30	0,02	0,03	0,06	0,12	0,24	0,24	0,12	0,06	0,03	0,02
0,10	0,01	0,01	0,02	0,04	0,08	0,08	0,04	0,02	0,01	0,01
	0,05 Muy bajo	0,10 Bajo	0,20 Moderado	0,40 Alto	0,80 Muy Alto	0,80 Muy Alto	0,40 Alto	0,20 Moderado	0,10 Bajo	0,05 Muy Bajo
	Impacto Negativo					Impacto Positivo				

Cada riesgo es evaluado de acuerdo a la probabilidad de que ocurra y al impacto en algún objetivo si ocurriera. Los umbrales de tolerancia de cada organización se trasladan a la matriz, de manera que las áreas verde, amarilla y roja indiquen estos umbrales para la priorización de riesgos.

Figura 10.2: Matriz de probabilidad e impacto

- Transferir el riesgo: Alguna entidad externa que se contrate afrontará el riesgo.
- Mitigar el riesgo: Se reduce el impacto del riesgo.
- Asumir el riesgo: Se asumen los riesgos y se trabaja sobre ellos.

Las respuestas ante cada tipo de riesgo será decisión de la alumna responsable de este proyecto.

Monitorización y control de riesgos

La monitorización y control de riesgos permite implementar los planes de respuesta a riesgos previamente mencionados, rastrear los riesgos identificados y monitorear los riesgos residuales, además de evaluar la efectividad de las medidas tomadas. Si surgen nuevos riesgos, estos serán analizados y se tomarán las mismas medidas que con aquellos riesgos identificados previamente. Esta monitorización se enfocará de forma distinta según la importancia establecida, lo que afectará de distinta forma a lo largo del ciclo de vida del proyecto.

PRESUPUESTO

Definición de empresa

En esta sección se definirá la empresa ficticia que produce este proyecto. Este empresa es creada únicamente con el objetivo de crear un presupuesto realista en el caso de que se decidiera implantar el sistema desarrollado.

Todos los roles de esta empresa han sido desarrollados por la autora de este proyecto, Alba Aparicio Pérez.

Los costes indirectos (Figura 10.4), precio/hora (Figura 10.7 y Figura 10.8), costes de los medios de producción (Figura 10.5) y productividad se han establecido en referencia al proyecto realizado en la asignatura **Diseño y Planificación de Proyectos del Software**.

Nº	Concepto	Importe
1	Total de los costes directos	155.942,50 €
2	Total de los costes indirectos	100.068,97 €
3	Suma de los costes directos e indirectos	256.011,47 €
4	Beneficio deseado (25%)	64.002,87 €
5	Coste total (suma de los costos directos, indirectos y beneficios)	320.014,34 €
6	Facturación posible en función de las horas de producción y de los precios por hora	351.204,00 €
7	Margen entre el coste total y la facturación (relación entre 5 y 6)	8,88%

Figura 10.3: Presupuesto. Resumen.

Servicio	Coste mes	Coste año
Alquiler oficina coworking	265,00 €	3.180,00 €
Gestoría	45,00 €	540,00 €
Tributos / tasas diversas	700,00 €	8.400,00 €
Gastos de mantenimiento, reparación...	20,00 €	240,00 €
Pequeño utillaje y herramientas	5,00 €	60,00 €
Primas seguro	115,00 €	1.380,00 €
Portes y gastos de transporte	35,00 €	420,00 €
Pack Fibr+Móvil empresa	44,00 €	528,00 €
Asesoría	150	1.800,00 €
Consumo eléctrico	70,00 €	840,00 €
Gastos en comunicaciones	5,00 €	60,00 €
Material de oficina	20,00 €	240,00 €
Gasto mensajería	30,00 €	360,00 €
TOTAL		18.048,00 €

Figura 10.4: Presupuesto. Costes indirectos.

Equipo/Licencia	Unid.	Precio	Coste Total	Coste año	Tipo	Plazo
Portátil Lenovo	6	700,00 €	4.200,00 €	840,00 €	Amortización	5
Periféricos	7	20,00 €	140,00 €	28,00 €	Amortización	5
Monitor 27"	7	169,98 €	1.189,86 €	237,97 €	Amortización	5
Licencias de software	6	75,00 €	450,00 €	450,00 €	Alquiler	1
			1.555,97 €			

Figura 10.5: Presupuesto. Medios de producción.

Personal	Total	CD(%)	Coste Directo	CI(%)	Coste Indirecto
CEO	53.200,00 €	0%	0,00 €	100%	53.200,00 €
CTO / Jefe de proyecto	49.210,00 €	50%	24.605,00 €	50%	12.302,50 €
Investigador	46.550,00 €	100%	46.550,00 €	0%	0,00 €
Programador DevOps	99.750,00 €	85%	84.787,50 €	15%	14.962,50 €
Total	248.710,00 €		155.942,50 €		80.465,00 €

Figura 10.6: Presupuesto. Personal.

Personal	Precio / hora (sin beneficios)
Investigadora	28,61 €
Programador DevOps	21,64 €

Figura 10.7: Presupuesto. Personal. Precio por Hora.

Investigación

En esta sección es detallada la entrada de Investigación del presupuesto. Se puede ver en la Figura 10.9.

Personal	Precio / hora
Consultor de tecnología	25,00 €
Diseñador gráfico	20,00 €

Figura 10.8: Presupuesto. Personal Externo. Precio por Hora.

Investigación										
Item 1	Item 2	Item 3	Partida	Cantidad	Unidades	Importe	Subtotal(3)	Subtotal(2)	Subtotal(1)	Total
1			Investigación sobre física cuántica						314,71 €	
	01		Investigadora	1	día	28,61		314,71		
2			Formación sobre álgebra básica						629,42 €	
	01		Investigadora	2	días	28,61		629,42		
3			Investigación sobre computación cuántica						1.573,55 €	
	01		Investigadora	5	días	28,61		1573,55		
4			Investigación sobre machine learning						1.573,55 €	
	01		Investigar sobre algoritmos de machine learning de clasificación					629,42		
		001	Investigadora	2	días	28,61	629,42			
	02		Investigar sobre SVM					314,71		
		001	Investigadora	1	días	28,61	314,71			
	03		Investigar sobre Redes Neuronales					629,42		
		001	Investigadora	2	días	28,61	629,42			
5			Investigación sobre quantum machine learning						7.553,04 €	
	01		Investigar sobre QSVM					3.776,52 €		
		001	Investigadora	12	días	28,61	3.776,52 €			
	02		Investigar sobre QNN					3.776,52 €		
		001	Investigadora	12	días	28,61	3.776,52 €			
6			Investigación sobre simuladores y ordenadores cuánticos reales IBM Quantum Experience						629,42 €	
	01		Investigadora	2	días	28,61		629,42		
TOTAL										12.273,69 €

Figura 10.9: Presupuesto. Investigación

Desarrollo del prototipo

En esta sección es detallada la entrada de Desarrollo del prototipo del presupuesto. Se puede ver en las Figuras 10.10, 10.11 y 10.12.

AMPLIAR

Desarrollo del prototipo												
Item (1)	Item (2)	Item (3)	Item (4)	Partida	Cantidad	Unidades	Importe	Subtotal(4)	Subtotal(3)	Subtotal(2)	Subtotal(1)	Total
1				Análisis e identificación de casos de uso							238,04 €	
	01			Programador DevOps 3	11	horas	238,04 €			238,04 €		
2				Diseño de la arquitectura							476,08 €	
	01			Programador DevOps 1	22	horas	476,08 €			476,08 €		
3				Implementación de QSVM							908,88 €	
	01			Requisitos y casos de uso						43,28 €		
		001		Programador DevOps 2	2	horas	43,28 €		43,28 €			
	02			Implementar prototipo en Jupyter						779,04 €		
		001		Prototipo sin kernel					779,04 €			
			0001	Programador DevOps 2	11	horas	238,04 €	238,04 €				
				Prototipo con kernel								
			0001	Programador DevOps 3	22	horas	476,08 €	476,08 €				
				Adaptar a diferentes conjuntos de datos								
			0001	Programador DevOps 2	3	horas	64,92 €	64,92 €				
	03			Implementar prototipo en Python						86,56 €		
		001		Desacoplar el conjunto de datos del qsvm					43,28 €			
			0001	Programador DevOps 2	2	horas	43,28 €	43,28 €				
		002		Desacoplar el modo de ejecución del qsvm					43,28 €			
			0001	Programador DevOps 2	2	horas	43,28 €	43,28 €				
4				Implementación de QNN							597,49 €	
	01			Requisitos y casos de uso						43,28 €		
		001		Programador DevOps 1	2	horas	43,28 €		43,28 €			
	02			Estudio de precisión con diferentes tipos de QNN						258,95 €		
		001		Programador DevOps 2	8	horas	173,12 €		173,12 €			
		002		Investigadora	3	horas	85,83 €		85,83 €			
	03			Estudio de precisión con diferentes optimizadores						143,78 €		
		001		Programador DevOps 1	4	horas	86,56 €		86,56 €			
		002		Investigadora	2	horas	57,22 €		57,22 €			
	04			Implementar prototipo en Jupyter						64,92 €		
		001		Adaptar a diferentes conjuntos de datos					64,92 €			
			0001	Programador DevOps 1	3	horas	64,92 €	64,92 €				
	05			Implementar prototipo en Python						86,56 €		

PLACIONES

Figura 10.10: Presupuesto. Desarrollo del prototipo (1)

	05		Implementar prototipo en Python						86,56 €		
		001	Desacoplar el conjunto de datos del qnn					43,28 €			
		0001	Programador DevOps 3	2	horas	43,28 €	43,28 €				
		002	Desacoplar el modo de ejecución del qnn					43,28 €			
		0001	Programador DevOps 2	2	horas	43,28 €	43,28 €				
5			Implementación de las Entidades							1.298,40 €	
	01		Requisitos y casos de uso						43,28 €		
		001	Programador DevOps 1	2	horas	43,28 €		43,28 €			
	02		Implementación de Dataset						43,28 €		
		001	Programador DevOps 2	2	horas	43,28 €		43,28 €			
	03		Implementación de Executor						562,64 €		
		001	Implementación de IBMExecutor					476,08 €			
		0001	Programador DevOps 3	11	horas	238,04 €	238,04 €				
		0002	Programador DevOps 2	11	horas	238,04 €	238,04 €				
		001	Implementación de LocalExecutor					86,56 €			
		0001	Programador DevOps 1	4	horas	86,56 €	86,56 €				
	04		Implementación de Validator						43,28 €		
		001	Programador DevOps 1	2	horas	43,28 €		43,28 €			
	05		Implementación de QMLAlgorithm						605,92 €		
		001	Programador DevOps 1	14	horas	302,96 €		302,96 €			
		002	Programador DevOps 3	14	horas	302,96 €		302,96 €			
6			Aplicación web							432,80 €	
	01		Requisitos y casos de uso						43,28 €		
		001	Programador DevOps 2	2	horas	43,28 €		43,28 €			
	02		Formación sobre Flask						43,28 €		
		001	Programador DevOps 2	2	horas	43,28 €		43,28 €			
	03		Diseñar la interfaz gráfica						64,92 €		
		001	Programador DevOps 2	3	horas	64,92 €		64,92 €			
	04		Implementación de pantallas						281,32 €		

Figura 10.11: Presupuesto. Desarrollo del prototipo (2)

AMPLIACIONES

04			Implementación de pantallas						281,32 €		
	001		Implementación de la pantalla principal					64,92 €			
		0001	Programador DevOps 1	3	horas	64,92 €	64,92 €				
	002		Implementación de la pantalla de documentación					21,64 €			
		0001	Programador DevOps 1	1	horas	21,64 €	21,64 €				
	003		Implementación de la pantalla de contacto					21,64 €			
		0001	Programador DevOps 1	1	horas	21,64 €	21,64 €				
	004		Implementación de la pantalla de algoritmos					64,92 €			
		0001	Programador DevOps 2	3	horas	64,92 €	64,92 €				
	005		Implementación de la pantalla de ejecución					43,28 €			
		0001	Programador DevOps 2	2	horas	43,28 €	43,28 €				
	006		Implementación de la pantalla de salida					21,64 €			
		0001	Programador DevOps 1	1	horas	21,64 €	21,64 €				
	007		Implementación de la pantalla de error					43,28 €			
		0001	Programador DevOps 2	2	horas	43,28 €	43,28 €				
7			Integrar back-end y front-end							108,20 €	
	01		Programador DevOps 3	5	horas	108,20 €			108,20 €		
8			Pruebas de integración							108,20 €	
	01		Programador DevOps 1	5	horas	108,20 €			108,20 €		
9			Pruebas unitarias							108,20 €	
	01		Programador DevOps 2	5	horas	108,20 €			108,20 €		
TOTAL											4.276,29 €

IONES Y AMPLIACIONES

Figura 10.12: Presupuesto. Desarrollo del prototipo (3)

Documentación

En esta sección es detallada la entrada de Documentación del presupuesto. Se puede ver en las Figuras 10.13, 10.14 y 10.15.

Documentación									AMPLIAC
Item 1	Item 2	Item 3	Descripción	Cantidad	Unidades	Importe	Subtotal(3)	Subtotal(2)	Subtotal(1)
1			¿Qué es este trabajo?						281,32 €
	01		Resumen					86,56 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €		
	02		Palabras clave					86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €		
	03		Abstract					86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €		
	04		Keywords					21,64 €	
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €		
2			Aspectos teóricos						16.246,93 €
	01		Introducción					86,56 €	
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €		
	02		La computación cuántica					13.979,44 €	
		001	Programador DevOps 1	15	horas	324,60 €	4.869,00 €		
		002	Programador DevOps 2	15	horas	324,60 €	4.869,00 €		
		003	Programador DevOps 3	14	horas	302,96 €	4.241,44 €		
	03		La Inteligencia artificial					779,04 €	
		001	Programador DevOps 3	6	horas	129,84 €	779,04 €		
	04		Aprendizaje Automático Cuántico					1.401,89 €	
		001	Investigadora	7	horas	200,27 €	1.401,89 €		
3			Planificación del sistema de información						389,52 €
	01		Inicio del plan de sistemas de información					194,76 €	
		001	Programador DevOps 1	3	horas	64,92 €	194,76 €		
	02		Definición y organización del PSI					194,76 €	
		001	Programador DevOps 2	3	horas	64,92 €	194,76 €		
4			Definición de la arquitectura tecnológica						108,20 €
	01		Identificación de las necesidades de infraestructura tecnológica					86,56 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €		
	02		Selección de la arquitectura tecnológica					21,64 €	
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €		
5			Estudio de viabilidad del sistema						86,56 €
									AMPLIACIONES

Figura 10.13: Presupuesto. Documentación (1)

									AMPLIACIONES
5			Estudio de viabilidad del sistema						86,56 €
	01		Programador DevOps 2	2	horas	43,28 €		86,56 €	
6			Planificación y gestión del tfg						6.989,72 €
	01		Planificación del proyecto					1.060,36 €	
		001	Programador DevOps 3	7	horas	151,48 €	1.060,36 €		
	02		Ejecución del proyecto					194,76 €	
		001	Programador DevOps 2	3	horas	64,92 €	194,76 €		
	03		Cierre del proyecto					5.734,60 €	
		001	Programador DevOps 1	11	horas	238,04 €	2.618,44 €		
		002	Programador DevOps 3	12	horas	259,68 €	3.116,16 €		
7			Análisis del sistema de información						1882,68
	01		Definición del sistema					21,64 €	
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €		
	02		Establecimiento de requisitos					779,04 €	
		001	Programador DevOps 2	6	horas	129,84 €	779,04 €		
	03		Análisis de los casos de uso					346,24 €	
		001	Programador DevOps 1	4	horas	86,56 €	346,24 €		
	04		Análisis de clases					346,24 €	
		001	Programador DevOps 3	4	horas	86,56 €	346,24 €		
	05		Definición de interfaces de usuario					194,76 €	
		001	Programador DevOps 3	3	horas	64,92 €	194,76 €		
	06		Especificación del plan de pruebas					194,76 €	
		001	Programador DevOps 1	3	horas	64,92 €	194,76 €		
8			Diseño del sistema de información						3.981,76 €
	01		Diseño de casos de uso reales					779,04 €	
		001	Programador DevOps 1	6	horas	129,84 €	779,04 €		
	02		Diseño de clases					541,00 €	
		001	Programador DevOps 1	5	horas	108,20 €	541,00 €		
	03		Diseño de la arquitectura de módulos del sistema					2.618,44 €	
		001	Programador DevOps 2	11	horas	238,04 €	2.618,44 €		

Figura 10.14: Presupuesto. Documentación (2)

								AMPLIACION
	03		Diseño de la arquitectura de módulos del sistema					2.618,44 €
		001	Programador DevOps 2	11	horas	238,04 €	2.618,44 €	
	04		Diseño físico de datos					21,64 €
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €	
	05		Especificación técnica del plan de pruebas					21,64 €
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €	
9			Construcción del sistema de información					562,64 €
	01		Preparación del entorno de generación y construcción					86,56 €
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €	
	02		Generación del código de los componentes y procedimientos					86,56 €
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
	03		Ejecución de las pruebas unitarias					21,64 €
		001	Programador DevOps 1	1	horas	21,64 €	21,64 €	
	04		Ejecución de las pruebas de integración					86,56 €
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
	05		Ejecución de las pruebas de sistema					86,56 €
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
	06		Elaboración de los manuales usuario					194,76 €
		001	Programador DevOps 3	3	horas	64,92 €	194,76 €	
10			Conclusiones y ampliaciones					302,96 €
	01		Conclusiones					21,64 €
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €	
	02		Ampliaciones					21,64 €
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €	
11			Anexos					259,68 €
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €	
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
TOTAL								30.832,29 €
								Y AMPLIACIONES

Figura 10.15: Presupuesto. Documentación (3)

Otros costes								
Item 1	Item 2	Descripción	Cantidad	Unidades	Importe	Subtotal(2)	Subtotal(1)	Total
1		Personal externo					690,00 €	
	01	Consultor de tecnología	10	horas	25,00 €	250,00 €		
	02	Diseñador gráfico	22	horas	20,00 €	440,00 €		
2		Reuniones					1.300,00 €	
	01	Arranque	2	horas	50	100		
	02	Bisemanales	20	horas	50	1.000,00 €		
	03	Cierre	4	horas	50	200,00 €		
TOTAL								1.990,00 €

Figura 10.16: Presupuesto. Otros costes.

Otros costes

En esta sección es detallada la entrada de otros costes del presupuesto. Se puede ver en la Figura 10.16.

Presupuesto de costes

El presupuestos de costes resumido se puede ver en la Figura 10.17.

Presupuesto de costes		
Cod.	Partida	Total
01	Investigación	12.273,69 €
02	Desarrollo del prototipo	4.276,29 €
03	Documentación	30.832,29 €
04	Otros costes	1.990,00 €
TOTAL COSTES		49.372,27 €

Figura 10.17: Presupuesto de Costes

PRESUPUESTO FINAL

Definición de empresa

La empresa a definir es exactamente la misma que al comienzo del proyecto (ver Figura 10.3).

Investigación

La entrada de investigación del presupuesto final es la misma que en el presupuesto inicial (ver Figura 10.9).

Desarrollo del prototipo

La entrada de desarrollo del prototipo del presupuesto final es la misma que en el presupuesto inicial (ver Figura 10.10).

Documentación

En esta sección es detallada la entrada de documentación del presupuesto final. Se puede ver en las Figuras 10.18, 10.19 y 10.20.

Documentación									AMPLIA
Item 1	Item 2	Item 3	Descripción	Cantidad	Unidades	Importe	Subtotal(3)	Subtotal(2)	Subtotal(1)
1			¿Qué es este trabajo?						281,32 €
	01		Resumen					86,56 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €		
	02		Palabras clave					86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €		
	03		Abstract					86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €		
	04		Keywords					21,64 €	
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €		
2			Aspectos teóricos						16.246,93 €
	01		Introducción					86,56 €	
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €		
	02		La computación cuántica					13.979,44 €	
		001	Programador DevOps 1	15	horas	324,60 €	4.869,00 €		
		002	Programador DevOps 2	15	horas	324,60 €	4.869,00 €		
		003	Programador DevOps 3	14	horas	302,96 €	4.241,44 €		
	03		La Inteligencia artificial					779,04 €	
		001	Programador DevOps 3	6	horas	129,84 €	779,04 €		
	04		Apredizaje Automático Cuántico					1.401,89 €	
		001	Investigadora	7	horas	200,27 €	1.401,89 €		
3			Planificación del sistema de información						389,52 €
	01		Inicio del plan de sistemas de información					194,76 €	
		001	Programador DevOps 1	3	horas	64,92 €	194,76 €		
	02		Definición y organización del PSI					194,76 €	
		001	Programador DevOps 2	3	horas	64,92 €	194,76 €		
4			Definición de la arquitectura tecnológica						108,20 €
	01		Identificación de las necesidades de infraestructura tecnológica					86,56 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €		
	02		Selección de la arquitectura tecnológica					21,64 €	
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €		
5			Estudio de viabilidad del sistema						86,56 €
	01		Programador DevOps 2	2	horas	43,28 €		86,56 €	
									PLACIONES

Figura 10.18: Presupuesto final. Documentación (1)

								AMPLIAC
5			Estudio de viabilidad del sistema					86,56 €
	01		Programador DevOps 2	2	horas	43,28 €		86,56 €
6			Planificación y gestión del tfg					6.989,72 €
	01		Planificación del proyecto					1.060,36 €
		001	Programador DevOps 3	7	horas	151,48 €	1.060,36 €	
	02		Ejecución del proyecto					194,76 €
		001	Programador DevOps 2	3	horas	64,92 €	194,76 €	
	03		Cierre del proyecto					5.734,60 €
		001	Programador DevOps 1	11	horas	238,04 €	2.618,44 €	
		002	Programador DevOps 3	12	horas	259,68 €	3.116,16 €	
7			Análisis del sistema de información					1882,68
	01		Definición del sistema					21,64 €
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €	
	02		Establecimiento de requisitos					779,04 €
		001	Programador DevOps 2	6	horas	129,84 €	779,04 €	
	03		Análisis de los casos de uso					346,24 €
		001	Programador DevOps 1	4	horas	86,56 €	346,24 €	
	04		Análisis de clases					346,24 €
		001	Programador DevOps 3	4	horas	86,56 €	346,24 €	
	05		Definición de interfaces de usuario					194,76 €
		001	Programador DevOps 3	3	horas	64,92 €	194,76 €	
	06		Especificación del plan de pruebas					194,76 €
		001	Programador DevOps 1	3	horas	64,92 €	194,76 €	
8			Diseño del sistema de información					3.981,76 €
	01		Diseño de casos de uso reales					779,04 €
		001	Programador DevOps 1	6	horas	129,84 €	779,04 €	
	02		Diseño de clases					541,00 €
		001	Programador DevOps 1	5	horas	108,20 €	541,00 €	
	03		Diseño de la arquitectura de módulos del sistema					2.618,44 €
		001	Programador DevOps 2	11	horas	238,04 €	2.618,44 €	
	04		Diseño físico de datos					21,64 €
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €	
	05		Especificación técnica del plan de pruebas					21,64 €
								AMPLIACIONES

Figura 10.19: Presupuesto final. Documentación (2)

								AMPLIACIONES
	05		Especificación técnica del plan de pruebas				21,64 €	
		001	Programador DevOps 3	1	horas	21,64 €	21,64 €	
9			Construcción del sistema de información					10.127,52 €
	01		Preparación del entorno de generación y construcción				86,56 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €	
	02		Generación del código de los componentes y procedimientos				86,56 €	
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
	03		Ejecución de las pruebas unitarias				21,64 €	
		001	Programador DevOps 1	1	horas	21,64 €	21,64 €	
	04		Ejecución de las pruebas de integración				4.869,00 €	
		001	Programador DevOps 3	15	horas	324,60 €	4.869,00 €	
	05		Ejecución de las pruebas de sistema				4.869,00 €	
		001	Programador DevOps 3	15	horas	324,60 €	4.869,00 €	
	06		Elaboración de los manuales usuario				194,76 €	
		001	Programador DevOps 3	3	horas	64,92 €	194,76 €	
10			Conclusiones y ampliaciones					302,96 €
	01		Conclusiones				21,64 €	
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €	
	02		Ampliaciones				21,64 €	
		001	Programador DevOps 2	1	horas	21,64 €	21,64 €	
11			Anexos				259,68 €	
		001	Programador DevOps 1	2	horas	43,28 €	86,56 €	
		001	Programador DevOps 2	2	horas	43,28 €	86,56 €	
		001	Programador DevOps 3	2	horas	43,28 €	86,56 €	
TOTAL								40.397,17 €
								TES Y AMPLIACIONES

Figura 10.20: Presupuesto final. Documentación (3)

Otros costes

La entrada de otros costes del prototipo del presupuesto final es la misma que en el presupuesto inicial (ver Figura 10.16).

Presupuesto de costes

El presupuesto de costes final resumido se puede ver en la Figura 10.21.

Presupuesto de costes		
Cod.	Partida	Total
01	Investigación	12.273,69 €
02	Desarrollo del prototipo	4.276,29 €
03	Documentación	40.397,17 €
04	Otros costes	1.990,00 €
TOTAL COSTES		58.937,15 €

Figura 10.21: Presupuesto final de Costes

CONTENIDO ENTREGADO EN LOS ANEXOS

Contenidos

Además de este documento, se hace entrega de una carpeta comprimida “.zip” en la que ahora se describirán sus contenidos. Se estructurará también la organización del código fuente.

- **Planificación_TFG.mpp** → Archivo de Microsoft Project que contiene la planificación del proyecto entera.
- **Presupuesto_TFG.xlsx** → Archivo Microsoft Excel que contiene los cálculos del presupuesto del proyecto.
- **Diagramas** → Carpeta que contiene todos los diagramas utilizados en este documento.
 - *Diagrama_de_clases.png*
 - *Diagrama_de_componentes.png*
 - *Diagrama_de_despliegue.png*
 - *Diagrama_de_estados_Cuenta_IBM.png*
 - *Diagrama_de_estados_Ejecucion.png*
 - *Diagrama_de_interaccion_Caso_Obtener_ordenador_cuántico.png*
 - *Diagrama_de_interaccion_Caso_Cargar_cuenta_IBM.png*
 - *Diagrama_de_interaccion_Caso_Activar_cuenta_IBM.png*
 - *Diagrama_de_interaccion_Caso_QSVM_simulador.png*
 - *Diagrama_de_interaccion_Caso_QSVM_ordenador.png*
 - *Diagrama_de_interaccion_Caso_QNN_simulador.png*
 - *Diagrama_de_interaccion_Caso_QNN_ordenador.png*
 - *Diagrama_de_paquetes.png*
 - *Diagrama_navegabilidad.png*
 - *Diagrama_entidad_relacion.png*
 - *Diagrama_de_robustez_Ejecutar_QNN_ordenador.png*
 - *Diagrama_de_robustez_Ejecutar_QNN_simulador.png*
 - *Diagrama_de_robustez_Ejecutar_QSVM_ordenador.png*
 - *Diagrama_de_robustez_Ejecutar_QSVM_simulador.png*
 - *Diagrama_de_robustez_Obtener_ordenador.png*
 - *Diagrama_de_robustez_Activar_cuenta_IBM.png*
 - *Diagrama_de_robustez_Cargar_cuenta_IBM.png*
 - *Diagrama_de_robustez_Ejecutar_QSVM_Simulador.png*
- **TFG_codigo.zip** → Carpeta comprimida con todo el código fuente.

Ahora se mostrará el contenido de dicha carpeta comprimida que contiene todo el código fuente de la aplicación:

- **env** → Carpeta que contiene el entorno virtual en el que está creado la aplicación web.
- **src** → Carpeta que contiene el código para todos los componentes creados.
- **static** → Carpeta que contiene elementos visuales de la aplicación y las guías de estilo.
- **templates** → Carpeta que contiene los archivos html de todas las pantallas.
- **tests** → Carpeta que contiene las clases de las pruebas unitarias, de integración y de sistema.
- **app.py** → Clase principal y encargada de que comience la aplicación entera.
- **requirements.txt** → Archivo que contiene todas las dependencias del sistema.
- **Otros archivos** → Los demás archivos no son relevantes ya que muchos se generan por defecto y los demás son configuraciones propias de github.

Bibliografía

- [1] Aiqom, “Why is ai booming now?.” <https://aiqom.ai/en/blogs/why-is-ai-booming-now>, Aug 2021. Online; accessed 27 April 2022.
- [2] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O’Reilly, 2 ed., 2019.
- [3] S. Maths, “Representación de complejos en forma polar.” <https://www.sangakoo.com/es/temas/representacion-de-complejos-en-forma-polar>. Online; accessed 3 Mar 2022.
- [4] R. Larson, V. D. Oliver, and P. Pastrana, *Fundamentos de algebra lineal*. Cengage Learning, 2015.
- [5] V. Rodríguez Bouza *et al.*, “Sobre los cuaterniones, álgebras de lie, y matrices de pauli. teoría básica y aplicaciones físicas,” *Repositorio Institucional de la Universidad de Oviedo*, 2012.
- [6] L. de Broglie, “The wave nature of the electron.” <https://www.nobelprize.org/uploads/2016/04/broglie-lecture.pdf>, 1929. Online; accessed 20 Mar 2022.
- [7] R. Álvarez Nodarse, Jordi, and R. A. Nodarse, “Y las ondas se convirtieron en partículas.” <https://institucional.us.es/blogimus/2019/04/y-las-ondas-se-convirtieron-en-particulas/>, Sep 2019. Online; accessed 21 Mar 2022.
- [8] B. P. A. Einstein and N. Rosen, “Can quantum-mechanical description of physical reality be considered complete?,” *Physical Review*, vol. 47, no. 777, p. 3, 1935.
- [9] D. Quiñones, *Síntesis de Circuitos Cuánticos*. PhD thesis, 09 2012.
- [10] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers. Machine Learning*. Springer, 2018.
- [11] J. Preskill, “Quantum Computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, Aug. 2018.
- [12] P. N. Roldán, “Modelo de regresión - definición, qué es y concepto.” <https://economipedia.com/definiciones/modelo-de-regresion.html>, 09 2016. Online; accessed 31 May 2022.

- [13] S. Bhattacharyya, I. Pan, A. Mani, S. De, E. Behrman, and S. Chakraborti, *Quantum Machine Learning*, p. 73. De Gruyter, 2020.
- [14] J. M. Heras and J. G. V. Portella, “Máquinas de vectores de soporte (svm).” <https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/>, May 2019. Online; accessed 27 May 2022.
- [15] R. Misra, “Support vector machines-soft margin formulation and kernel trick.” <https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe>, Nov 2021. Online; accessed 27 May 2022.
- [16] M. Schuld, “Supervised quantum machine learning models are kernel methods,” *ArXiv*, no. 2101.11020, 2021.
- [17] Olive.zhao, Thibay, Helen.F, Unicef, MahMush, Zebra, LucianoNhantumbo, VinceD, and Yencao, “Machine learning algorithms: Support vector machine (svm).” <https://forum.huawei.com/enterprise/en/machine-learning-algorithms-support-vector-machine-svm/thread/722247-895/>. Online; accessed 29 May 2022.
- [18] A. Matic, M. Monnet, J. M. Lorenz, B. Schachtner, and T. Messerer, “Quantum-classical convolutional neural networks in radiological image classification,” *ArXiv*, no. 2204.12390, 2022.
- [19] A. Mohanty, “Multi layer perceptron (mlp) models on real world banking data.” <https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f>, May 2019. Online; accessed 10 Jun 2022.
- [20] “Gradient descent - qiskit documentation.” <https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.GradientDescent.html>. Online; accessed 27 Jun 2022.
- [21] PennyLane, “Documentación pennylane.” <https://pennylane.readthedocs.io/en/stable/>, 2022. Online; accessed 27 Jun 2022.
- [22] Qiskit, “Documentación simuladores qiskit.” https://qiskit.org/documentation/locale/es_UN/tutorials/simulators/1_aer_provider.html, 2022. Online; accessed 17 Jun 2022.
- [23] G. de España, “Métrica v.3.” https://administracionelectronica.gob.es/pae/Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html, 2001. Online; accessed 1 Jul 2022.
- [24] S. Agnihotri, “Quantum machine learning 102-qsvm using qiskit.” <https://shubham-agnihotri.medium.com/quantum-machine-learning-102-qsvm-using-qiskit-731956231a54>, Sep 2020. Online; accessed 7 Jul 2022.

- [25] N. Mishra, M. Kapil, H. Rakesh, A. Anand, N. Mishra, A. Warke, S. Sarkar, S. Dutta, S. Gupta, A. Prasad Dash, R. Gharat, Y. Chatterjee, S. Roy, S. Raj, V. Kumar Jain, S. Bagaria, S. Chaudhary, V. Singh, R. Maji, P. Dalei, B. K. Behera, S. Mukhopadhyay, and P. K. Panigrahi, *Quantum Machine Learning: A Review and Current Status*. Singapore: Springer Singapore, 2021.
- [26] PMI, *Guía de los fundamentos para la dirección de proyectos (guía del PM-BOK®)*. Newtown Square, Pensilvania: Project Management Institute, 5 ed., 2012.
- [27] Jose Manuel Redondo, “Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo.” https://www.researchgate.net/publication/327882831_Plantilla_de_Proyectos_de_Fin_de_Carrera_de_la_Escuela_de_Informatica_de_Oviedo, 2019. Online; accessed 13 Jul 2020.
- [28] Jose Manuel Redondo, “Creación y evaluación de plantillas para trabajos de fin de grado como buena práctica docente,” *Revista de Innovación y Buenas Prácticas Docentes*, vol. pp, no. pp, p. pp, 2020.
- [29] J. M. Requena, “El consejero de Universidad pide apostar por la innovación y generar conocimiento.” <https://www.lne.es/asturias/2019/08/13/consejero-universidad-pide-apostar-innovacion/2514937.html>, 2019. Online; accessed 13 Jul 2020.
- [30] Francesco Rodella, “¿Qué es y cómo funciona la computación cuántica en la nube.” <https://www.sacyr.com/-/-que-es-y-como-funciona-la-computacion-cuantica-en-la-nube-.html>, 2021. Online; accessed 04 Mar 2022.
- [31] C. Bernhardt, *Quantum Computing for everyone*. Cambridge, MA: The MIT Press, 1 ed., 2019.
- [32] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, ch. 2. Cambridge, UK: Cambridge University Press, 7 ed., 2010.
- [33] S. Pattanayak, *Quantum Machine Learning with Python*. Bangalore, India: Apress Media LLC, 1 ed., 2021.
- [34] P. C. Scott Pakin, “The problem with quantum computers.” <https://blogs.scientificamerican.com/observations/the-problem-with-quantum-computers/#:~:text=Quantum%20computers%20are%20exceedingly%20difficult,chance%20to%20run%20to%20completion.>, 2019. Online; accessed 27 Jun 2022.
- [35] E. F. Combarro, “Una introducción práctica al quantum machine learning y a los algoritmos variacionales cuánticos,” Nov 2021.