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

Julio 2022

Esta versión de la plantilla para Trabajos de Fin de Grado ha sido posible gracias a la donación de la exalumna Elena Allegue González de su documentación de Trabajo de Fin de Grado, que ha servido como base para elaborar esta versión. Aquí podréis encontrar todos los títulos y subtítulos de las secciones, pero las explicaciones se mantendrán en la versión *Word* de la plantilla (se proporciona una versión PDF de la misma para facilitar el acceso a las mismas). No obstante, del trabajo de Elena se han conservado ejemplos de como hacer elementos clave como imágenes, tablas, etc.

Desarrollar una versión *Latex* de la plantilla desde cero es una trabajo bastante largo, pero gracias al trabajo de Elena se ha podido equiparar esta versión con las de *Word* mucho más rápidamente.

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Ejemplos de elementos comúnes en Latex para mejorar el uso de la plantilla

Esta sección contiene ejemplos de como incluir ciertos elementos comunes en un documento Latex, extraídos del proyecto de **Elena Allegue Gonzalez**, que donó su documentación para poder hacer esta plantilla de trabajos de fin de grado y que su uso fuese mucho más sencillo. Esta sección contiene ejemplos de dichos elementos basados en lo que ella hizo en su proyecto (la aplicación *GuardMe* [3], para protección de personas en situaciones de peligro con el móvil), por lo que todos ellos deben ser entendidos en ese contexto. Por favor, no dejes ni esta sección ni ninguno de estos elementos en tu documentación porque son solo ejemplos que tienen sentido en el contexto de su proyecto. Ten en cuenta además que en el cuerpo del propio documento hay algún ejemplo más en la sección en la que corresponde, que debe adaptarse y eliminarse si no es de aplicación.

Paquetes de Latex usados en la plantilla

Paquetes de LaTeX

Para generar **documentación en LaTeX** se requiere del **uso de paquetes**, los cuales son introducidos al mismo mediante el comando \`usepack
age{name}`, que son instalados en caso de no estarlo ya en el propio editor de texto para su posterior uso.

Se han usado un **gran número de librerías**, las cuales clasificaré a continuación por tipo de licencia, mencionando en todas ellas su correspondiente autor/autora y funcionalidad:

- Licencia MIT
  - `enumitem` sirve para el control de las listas numeradas pudiendo, entre otras, diseñar el tipo de enumerado que aparece en el documento generado. Autor copyright: Javier Bezos (`https://github.com/jbezos/enumitem`).
- LaTeX Project Public License[1]
  - `nameref` sirve para crear una referencia basada en el nombre del título del capítulo, sección, subsección o lo que corresponda. Autor copyright: Oberdiek Package Support Group.
  - `hyperref` sirve para que las referencias, ya sean de tipo *ref* o *nameref* sean clicables y enlazadas con la referencia a la que apuntan, el índice también funciona de esta manera. Además, las urls de sitios web o documentos se abran en el navegador de Internet. Autor copyright: Oberdiek Package Support Group (`https://github.com/ho-tex/hyperref`).
  - `titlesec` sirve para poder crear diferentes estilos en los títulos de las secciones así como poner estilos concretos a una página. Autor copyright: Javier Bezos.
  - `float` sirve para que una imagen se mantenga en una posición concreta siempre en el documento. LaTeX si no se provee de esta configuración, coloca la imagen donde mejor entre, aunque esto suponga estar en la sección siguiente de donde debería estar. Autor: NaN.(Mantenimiento: Anselm Lingnau).
  - `fancyhdr` se utiliza para el control de los estilos y personalizaciones de posición de los encabezados y pies de páginas de LaTeX. Autor copyright: Piet van Oostrum.
  - `enumerate` se utiliza para cambiar el estilo de los marcadores de las listas enumeradas o no. Autor copyright: David Carlisle.

[1] La distribución de copias esta permitida, aunque no es posible modificar los documentos bajo esta licencia. LPPL es la licencia que posee el kernel del propio LaTeX además de ser la más común para la distribución sus paquetes.

- `makeidx` paquete ya instalado con MiKTeX con el cual se genera el índice del documento. Autor copyright: The LaTeX Team.
- `graphicx` sirve para poder darle tamaño y otro tipo de propiedades que el paquete básico `graphic` no incluye. Esta instalado con MiKTeX. Autor copyright: David Carlisle, Sebastian Rahtz.
- `url` sirve para dar formato a hipervínculos de páginas web, direcciones de ficheros, direcciones de correo electrónico, etc. Es el formato que se usa en el documento para las referencias y todo tipo de links. Autor copyright: NaN. (Mantenimiento: Donald Arseneau).
- `xurl` sirve para que el anterior paquete que crea las urls pueda tener un estilo en el que entre en la página, es decir, el anterior paquete no tiene saltos de linea de la url menos en una \, lo cual hacía que las urls fuesen demasiado largas e incluso se saliesen de la página, cosa que gracias a esta librería no pasa. Autor copyright: Nan (Mantenimiento: Herbert Voss).
- `footmisc` sirve para poder hacer notas a pie de página y poder darles formato. Autor copyright: Robin Fairbairns. (Mantenimiento: Frank Mittelbach).
- `report` es el paquete usado para esta plantilla. Es parecido al estilo de un libro, pero con pequeñas diferencias para ser un libro profesional. Autor: The LaTeX Team.
- `inputenc` sirve para especificar el tipo de codificación en la creación del documento. En este caso se utilizó *utf8*. Autor copyright: NaN. (Mantenimiento: The LaTeX Team, Frank Mittelbach).
- `babel` determina el tipo de lenguaje que se utiliza en el documento. Soporta más de 200 lenguajes y en este caso se utilizó español. Autor copyright: Javier Bezos and Johannes L. Braams (`https://github.com/latex3/babel`).
- `xcolor` sirve utilizar colores al igual que sombreados, ...Se ha utilizado para las instrucciones de la linea de comandos poniendo negros los cuadros de texto y blanca la letra. Autor copyright: Uwe Kern.
- `booktabs` se ha utilizado para las tabulaciones de las tablas que se generan con **??** y que es necesario para que estas compilen. Les ofrecen una mayor calidad y más componentes que no ofrece la tabla por sí sola. Autor copyright: NaN. (Mantenimiento: Danie Els).
- `multirow` sirve entre otras cosas para poder dividir una fila en varias como se puede ver por ejemplo en las tablas de la descripción de clases (**??**). Autor copyright: Piet van Oostrum.
- `amsmath` sirve para poder hacer uso de los diferentes estilos de operaciones matemáticas dentro de LaTeX. Autor copyright: LaTeX3 Project and American Mathematical Society.
- `listings` sirve para poder escribir código en el documento. Puede incluso detectar el tipo de lenguaje que se está usando para poder especificar colores. Autor copyright: Brooks Moses, Jobst Hoffmann.

- Dominio público
  - `titlepic` sirve para poder introducir una o varias imágenes a la portada del documento. La intalación de MiKTeX contiene este paquete por defecto. Autor copyright: NaN. (Mantenimiento: Thomas ten Cate).

- Free Licence (no especificada)
  - `eurosym` sirve para poder escribir el símbolo € a continuación de el número que se requiera. Autor copyright: NaN. (Mantenimiento: Henrik Theiling).

excel2latex

Un descubrimiento que realicé cuando estaba desarrollando la documentación fue excel2latex. Se trata de un complemento de *Excel* cuya funcionalidad se basa en **convertir cualquier tabla** que se encuentre en un archivo `.xls` (*Excel*) **en código *LaTeX***.

Para poder hacer uso de este, hay que hacer una descarga en `https://www.ctan.org/tex-archive/support/excel2latex/` del archivo *Excel2LaTeX.xla*. A continuación nos dirigiremos a *Excel* y con un documento abierto (no importa que esté vacío o no) nos dirigiremos a la siguiente ruta: *Archivo>Opciones>Complementos*.

Una vez la pantalla de Complementos esté abierta, en la parte inferior deberemos marcar: *Administrar:Complemento* *Excel*, y seguidamente clicar en `Ir...`. Ahi examinaremos hasta donde haya descargado el complemento, seleccionándolo. A continuación y después de aceptar, se recomienda reiniciar *Excel*.

El tipo de licencia que usa esta herramienta es la determinada por **LaTeX Project Public License**.

Figura 1: Complemento excel2latex en Excel

Simplemente habrá que seleccionar una tabla en *Excel*, y a continuación clicar en `Convert Table To LaTeX` que resultará en una nueva ventana con el código en el que se puede copiar e incluso realizar otro tipo de acciones.

Por último, destacar que la licencia por la que se rige es *The LaTeX Project Public License 1.3* y que a pesar de que no se identifica el autor del copyright, como mantenimiento se encuentran Chelsea Hughes y Kirill Müller.

Ejemplos de cómo hacer figuras

Ejemplo de dos figuras lado a lado

Figura 2: Figura izquierda

Figura 3: Figura derecha

Ejemplo de figura

Figura 4: Figura de ejemplo

Listas de items

Lista de items estándar

- El **uso de la aplicación** por parte de una gran cantidad de usuarios **sin importar el sistema operativo** del que dispongan.
- La posibilidad de realizar una **llamada a emergencias al (*112*)de manera rápida e intuitiva**, al mismo tiempo que se avisa a los protectores del usuario que solicite auxilio.
- La **monitorización de localización** por parte de los protectores de los usuarios que posean como protegidos.
- La **comunicación entre protector y protegido** a través de un servicio de mensajería dentro de la aplicación.

Lista estándar multinivel

- Prueba: el usuario se registrar por primera vez en la aplicación.
- Resultados
  - Esperado: el usuario aparece tanto en la lista de usuarios registrados de Firebase Authentication como en la base de datos de la aplicación.
  - Obtenido: el usuario parece inscrito en ambos registros.

Lista multinivel para Ingeniería de Requisitos

Llamada de emergencia

Requisitos no funcionales

RNF-1. El usuario deberá ser capaz de utilizar todas las funcionalidades desarrolladas en la aplicación sin problema.

RNF-2. La aplicación sera accesible para los usuarios a través de un portal de descarga.

RNF-2.1. La aplicación será multiplataforma.

RNF-2.1.1. La aplicación requiere mínimo las versiones:

RNF-2.1.1.1. 5.0 para Android

RNF-2.1.1.2. 10.0 para iOS

RNF-2.1.2. Las versiones están condicionadas por los requisitos de Expo.

RNF-3. Los servicios que utiliza la aplicación deberán mantenerla disponible el mayor tiempo posible.

RNF-3.1. El sistema estará disponible siguiendo el protocolo de los tres nueves: 99.9 %.

RNF-3.1.1. El sistema no estará disponible 43,8 minutos/mes u 8,76 horas/año.

RNF-4. Los usuarios de la aplicación no deberán tener conocimientos tecnológicos avanzados.

RNF-4.1. El nivel básico será requerido, lo que incluye haber tratado con un teléfono móvil alguna vez.

RNF-5. El sistema se conectará con la base de datos que albergará todos los datos asociados a los usuarios registrados y sus datos.

RNF-5.1. Las base de datos estará alojada en la nube.

RNF-5.2. Los tiempos de carga de datos no deberán sobrepasar los 10 segundos.

RNF-6. El sistema se comunicará con:

RNF-6.1. Google Maps API

RNF-6.2. Google Firebase Cloud Firestore

RNF-6.3. Google Firebase Authentication

RNF-6.4. Whatsapp

Ejemplos de cómo hacer tablas

Tabla descripción clases

Tabla 1: Análisis de LoginScreen

LoginScreen	
Descripción	
Es la encargada de las acciones y la renderización de la pantalla de inicio de sesión.	
Atributos propuestos	
-	
Métodos propuestos	
componentWillMount	
signIn	Inciar sesión en Firebase
render	

Tabla típica

Tabla 2: Planificación de Formación

Num. esq.	Nombre de tarea	Duración
1.1	**Formación propia**	**34 horas**
1.1.1	*Aprender sobre Expo*	*19,5 horas*
1.1.2	*Aprender sobre React-Native*	*8 horas*
1.1.3	*Aprender sobre base de datos*	*16,5 horas*
1.1.4	*Aprender sobre Typescript*	*3 horas*
1.1.5	*Aprender sobre Scaledrone*	*3 horas*
1.1.6	*Aprender sobre LaTeX*	*5,5 horas*

Tabla ejemplo para especificar un presupuesto

Tabla 3: Resumen del presupuesto

COD.	Descripción	Subtotal
001	*Formación*	*1.110,00€*
002	*Estudio de Alternativas*	*400,00€*
003	*Análisis y Diseño*	*1.840,00€*
004	*Desarrollo*	*10.860,00€*
005	*Documentación*	*3.985,00€*
	TOTAL	**18.195,00€**

Otros elementos

URLs

`www.javascript.com/learn/`

Fracciones

El portátil se trata de un Lenovo 80E5 i7 con 8 GB RAM y con 4 años de uso, el cual costó entonces 650€, por lo tanto la amortización será de 26€ al año según:

$$\frac{650 - 545}{4} \equiv 26$$

Capítulo 1

¿Qué es este trabajo?

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 espectacular fusión entre la mecánica cuántica y la ciencia de la computación, el cuál 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.

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, sin estar programado explícitamente para ello.

El objetivo de este trabajo es unir 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 ejecutarlo tanto en simuladores como en computadores cuánticos reales para ver su eficacia en comparación con ordenadores clásicos. 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.

La computación cuántica nos permite resolver problemas que con la computación clásica nos parecían impensables. Por ello, con esto se busca conseguir resolver problemas de Aprendizaje Automático que sean muy costosos o imposibles en ordenadores clásicos, contribuyendo así al desarrollo 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 spectacular 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 models.

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.

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, without being explicitly programmed to do so.

The goal of this work is to unite 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 to see how effective they are in comparison with classical computers. Specifically, Supervised Learning problems will be solved, which are those in which we have a set of input data from which our computer will learn what is necessary to solve the problem.

Quantum computing allows us to solve problems that seemed unthinkable with classical computing. Therefore, the aim is to solve Machine Learning problems that are very computational expensive or impossible with classical computers, thus contributing to the development 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 esta compuesto por una serie de bits, que pueden estar en un 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 está en una **combinación** de los estados 0 y 1. Esta es la diferencia fundamental y la base 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 maquina tiene tamaño n, la cual es una colección de n 1 o 0. Sin embargo, en un ordenador cuántico con n **qubits**, un estado concreto de la máquina es una combinación de todas las posibles colecciones de n 1 y 0. El caso es que hay 2^n posibles combinaciones, osea 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, y uniendo muchas de estas podemos crear 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 a día de hoy con un ordenador clásico, son muy costosos. Uno de esos problemas, que es el principal objetivo de este trabajo, son la creación y entrenamiento de algoritmos de Machine Learning.

La diferencia en cuanto a cambios de estados es que en la computación cuántica, están basados en probabilidades, por lo que no podemos conocer a priori cuál será el resultado de una ejecución aunque conozcamos todos los cambios de estado y las entradas, tan sólo conoceremos la **probabilidad** de los resultados.

Esto sucede porque la mecánica cuántica se rige por probabilidades. En la teoría cuántica, según la interpretación de Copenhague[1], se dice que las propiedades de una partícula no están determinadas hasta el mismo momento de ser medidas.

[1]Para 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`

Por otro lado, Machine Learning es una rama de la computación que aún está en desarrollo, por lo que un algoritmo de aprendizaje automático necesita grandes cantidades de datos para funcionar correctamente. Incluso para problemas simples como la clasificación, necesitaremos miles de ejemplos para que el sistema puede aprender lo necesario. Para problemas más complejos como el reconocimiento de voz se necesitan millones de ejemplos, 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.

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 álgebra lineal.

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, por lo que la evolución del sistema cuántica se puede representar mediante matrices de números complejos.

Se define la unidad imaginaria $\boldsymbol{i}$ como la raíz cuadrada del número real negativo $i = \sqrt{-1}$

Un número complejo $\boldsymbol{z}$, es la suma de un número real $\boldsymbol{a}$ más un número real $\boldsymbol{b}$ multiplicado por la unidad imaginaria $\boldsymbol{i}$:

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

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 las mismas operaciones matemáticas que ya conocemos con este conjunto, como 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 eje horizontal, y los números imaginarios se representan en el eje vertical, de forma que un número complejo, por ejemplo $c = 1 + 3i$, se representaría como:

figura del plano complejo, explicar también qe la linea del origen de coordenadas al puntos es el modulo de c

En la representación polar, el número complejo $z = a + b \cdot i$, en lugar de quedar determinado por sus componentes real e imaginaria, a y b, puede quedar fijado mediante su módulo y su argumento.

El argumento de un número complejo z es 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^o y 360^o.

figura del 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*.

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.

Álgebra lineal es el estudio de los *espacios vectoriales* y las operaciones lineales realizadas en dichos espacios [4].

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 $(z_1, ..., z_n)$, siendo n un entero positivo que indica el número de elementos que tienen sus vectores. Podemos representarlos usando la notación matricial:

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

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:

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

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:

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

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

Bases

Una base es un conjunto de vectores linealmente independientes y que son capaces de generar cualquier vector de dicho espacio. En nuestro estudio del plano, una base estará formada por dos vectores linealmente independientes.

Para cambiar de base se hace uso de las matrices de transición. Una **matriz de transición** o **matriz de Markov**, es una matriz cuadrada que describe las probabilidades de pasar de un estado a otro en un sistema dinámico. En cada fila están las probabilidades de pasar del estado representado por esa fila a los otros estados.

Una matriz de transición especialmente importante es la matriz de Hadamard, representada por:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Su importancia se debe a que una de las puertas lógicas cuánticas más usadas en la computación cuántica está basada en esta matriz.

Espacio de Hilbert

En mecánica cuántica, los estados de un sistema cuántico se representar en los **espacios de Hilbert**.

Un espacio de Hilbert es un espacio vectorial que es completo con respecto a la normal vectorial definida por el producto interno. Definiciones como espacios vectoriales completos o normas vectoriales entran en demasiado detalle para la materia que se va a presentar en este trabajo. Lo más importante que se debe conocer es que los espacios infinito-dimensionales son espacios de Hilbert, y estos, normalmente son sucesiones de números complejos o de funciones. Por lo que, en la mecánica cuántica, un conjunto de partículas se puede describir por un espacio de Hilbert que contenga las funciones de onda correspondientes a estas. [5]

Matrices de Pauli

Las matrices de Pauli o 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 cambio, para la computación cuántica son las bases de las puertas cuánticas que usaremos para transformar nuestros qubits. Las cuatro puertas simples más sencillas son las matrices de Pauli y 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. Mientras que la puerta I se corresponde con la matriz identidad y no cambia el estado del qubit. [6]

Producto tensorial

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

El espacio de estados de un sistema combinado es el **producto tensorial** $H_1 \otimes H_2$, donde H_1 y H_2 son los subsistemas a combinar, de forma que:

$$H_1 = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \; H_2 = \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

$$H_1 \otimes H_2 = \begin{bmatrix} a_{00} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} & a_{01} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \\ a_{10} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} & a_{11} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \end{bmatrix}$$

$$H_1 \otimes H2 = \begin{bmatrix} a_{00} * b_{00} & a_{00} * b_{01} & a_{01} * b_{00} & a_{01} * b_{01} \\ a_{00} * b_{10} & a_{00} * b_{11} & a_{01} * b_{10} & a_{01} * b_{11} \\ a_{10} * b_{00} & a_{10} * b_{01} & a_{11} * b_{00} & a_{11} * b_{01} \\ a_{10} * b_{10} & a_{10} * b_{11} & a_{11} * b_{10} & a_{11} * b_{11} \end{bmatrix}$$

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*[7]. Este postulado se basó en el conocido *efecto fotoeléctrico*, publicado por Albert Einstein, que sugiere la naturaleza cuántica de la luz.

Esta hipótesis de que la materia se comporta como una onda se confirmó gracias al experimento de la doble rendija.

En este experimento realizado por Thomas Young, se intentaba demostrar la naturaleza ondulatoria de la luz y lo consiguió. 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.
insertar foto experimento

Este experimento marcó un antes y un después en el estudio de la mecánica cuántica, ya que se consideró fundamental a la hora de demostrar la *dualidad onda corpúsculo*, la cual 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. Esto quiere decir que sera **imposible** calcular sucesos futuros, ya que el simple hecho de calcularlo ya lo está modificando.

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 un momento dado.

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 (el colapso de función de onda) era incompleta, ya que violaba el **principio de localidad**.

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 un 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 era incorrecta.

2.2.2.1. Estado cuántico

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

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.

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 probabilísticas correspondientes a los estados $|0\rangle$ y $|1\rangle$.

2.2.3. Teoría de la computación cuántica

circuitos, puertas logicas, medicion, aplicaciones y usos, limitaciones

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 $|1\rangle$, los cuales son los equivalentes a los estados 0 y 1 de un bit en computación clásica. También en forma de vector:

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

Un qubit, antes de medirlo, se escribirá como una combinación linear de los vectores base, es decir, tendrá esta forma: $a_0|b_0\rangle + a_1|b_1\rangle$. Si medimos el qubit, su estado automáticamente se colapsará a cualquiera de los dos posibles estados $|0\rangle$ y $|1\rangle$. La probabilidad de que al medir obtengamos $|0\rangle$ es ${d_0}^2$; la probabilidad de que al medir obtengamos $|1\rangle$ es ${d_1}^2$. Por lo tanto, es resultado final tras la medición, es el mismo que en un sistema clásico, al final obtenemos un valor de 0 o 1.

d_0 y d_1 es lo que se denomina *amplitudes*. El cuadrado de la amplitud es la probabilidad que hay de que el qubit colapse al valor correspondiente del vector, y este valor al ser una probabilidad, siempre será positivo. En cambio, la amplitud no tiene por qué ser positiva.

Esfera de Bloch

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

En computación cuántica, la esfera de Bloch es una representación geométrica de radio 1 de un qubit. Se trata de una esfera en la cual cada punto de la superficie corresponde a un posible estado del qubit. Por lo que cada par de puntos diametralmente opuestos sobre la esfera corresponde a dos estados orto-normales en el espacio de Hilbert.

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, por lo que 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 **X** podemos ver la esfera de Bloch.

Circuitos cuánticos

Un circuito cuántico, similar a los circuitos clásicos, es un modelo para la computación cuántica, en los 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.

FIGURA

Como se puede ver en la figura **X**, 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 al símbolo de medición.

FIGURA

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 lineas horizontales.

FIGURA

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.

Puerta lógicas cuánticas

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

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

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, estas 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, y casualmente, son las matrices de Pauli explicadas anteriormente.

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:

Tabla 2.1: Representación Puerta I

Puerta	**En circuito**	**En Matriz**
I	$\|0\rangle$ —[I]— $\|0\rangle$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

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 $|1\rangle$.[9] Actúa de forma:

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

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

Tabla 2.2: Representación Puerta X

Puerta	En circuito	En Matriz	Dirac
X	$\|0\rangle$ —[X]— $\|1\rangle$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\|1\rangle\langle 0\| + \|0\rangle\langle 1\|$

Puerta Y

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

Tabla 2.3: Representación Puerta Y

Puerta	En circuito	En Matriz	Dirac
Y	$\|0\rangle$ —[Y]— $i\,\|1\rangle$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$i\|1\rangle\langle 0\| - i\|0\rangle\langle 1\|$

Puerta Z

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

Tabla 2.4: Representación Puerta Z

Puerta	En circuito	En Matriz	Dirac
Z	$\|1\rangle$ —[Z]— $-\|1\rangle$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\|1\rangle\langle 0\| - \|0\rangle\langle 1\|$

Puerta H o de Hadamard

La puerta de Hadarmd 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 **X** podemos ver que en la esfera de Bloch esta operación representa una rotación de la esfea de 90^{o} en el eje y y 180^{o} en el eje x.

Tabla 2.5: Representación Puerta H

Puerta	En circuito	En Matriz	Dirac
H	$\|0\rangle$ —[Z]— $\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle)\langle 0\| + \frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\langle 1\|$

Puerta S

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

Tabla 2.6: Representación Puerta S

Puerta	En circuito	En Matriz	Dirac
S	$\|1\rangle$ — S — $i\,\|1\rangle$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\|0\rangle\langle 0\| + i\|1\rangle\langle 1\|$

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. [9]

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
$SWAP$	$\|0\rangle$ — ✕ — $\|1\rangle$ $\|1\rangle$ — ✕ — $\|0\rangle$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

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, entonces esta puerta crea **entrelazamiento cuántico** entre ambos qubits.

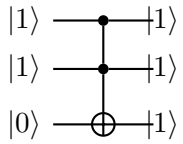
Se representa con un punto negro en la línea del qubit de control y un circulo con una línea horizontal y otra vertical en el qubit objetivo.

Tabla 2.8: Representación Puerta CNOT

Puerta	En circuito	En Matriz
$CNOT$	$\|1\rangle$ — ● — $\|1\rangle$ $\|1\rangle$ — ⊕ — $\|0\rangle$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

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:



2.2.4. Hardware de los ordenadores cuánticos

ibm, simuladores, ordenadores cuanticos reales, próximamente

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 la cual te permite programar para sus ordenadores cuánticos.

IBM, considerado el pionero en la creación de estos super 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 local: Desde el propio IDE podremos ejecutar en local el programa, aunque lógicamente el resultado obtenido será una simple simulación sobre el hardware normal que usamos.

En un simulador: Tanto Google como IBM ofrecen u servicio de simuladores online en los cuales podrás ejecutar tus programas. Estos simuladores no se ejecutan sobre ordenadores cuánticos reales pero te dan una aproximación acerca de los resultados y el tiempo que tardaría uno real en ejecutarlo.

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 de que cada ordenador tendrá una cola de trabajo, por lo que tendremos que esperar a nuestro turno para poder usarlo.

2.3. La Inteligencia Artificial

2.3.1. Introducción

La Inteligencia Artificial 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 que no se les ha encontrado un algoritmo capaz de resolverlos.

Un ejemplo de Inteligencia Artificial es el famoso filtro de spam utilizado en todos 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.

Machine Learning es una disciplina que crea sistemas que aprenden automáticamente, es decir, identifican patrones a partir de un conjunto de datos para predecir comportamientos futuros y actuar en consecuencia.

Dentro del Machine Learning disponemos de diversas técnicas, las cuáles podemos clasificar 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.

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 [9]. 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.

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).

Insertar imagen de regresión

Algunos algoritmos de regresión también pueden ser usados para problemas de clasificación. Por ejemplo, la **Regresión Logística**, como su nombre indica sería un algoritmo de regresión, pero es utilizado para problemas

de clasificación, ya que como salida nos devuelve un valor que sería la probabilidad que tiene un valor de corresponder a una clase o otra.

Los algoritmos de aprendizaje supervisado más importantes son:

- K vecinos más próximos
- Regresión Linear
- Regresión Logística
- Árboles de decisión y Bosques Aleatorios
- Máquinas de Vectores de Soporte
- Redes Neuronales

Siendo estos dos últimos los que se implementarán.

2.3.3. Cómo construir un sistema inteligente

Para poder construir un modelo de aprendizaje automático 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* [10], 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, Perfeccionar los parámetros del 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]En 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`

2.3.3.2. 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 con sólo 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.

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 decisión. 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 Regresión Logística, y Máquinas de Vectores de Soporte o Árboles de decisión como con los problemas de regresión.

La mayoría de modelos sirven tanto para problemas de regresión y clasificación (salvo la Regresión Lineal y la Regresión Logística). Por lo que a la hora de decidir cuál debemos usar, lo mejor sería entrenar varios modelos y comparar los resultados.

Tras entrenar los modelos podemos obtener una puntuación sobre la precisión de la predicción. Si obtenemos una baja puntuación es bastante probable que las características de los datos no proporcionan la suficiente información para hacer buenas predicciones, o es que el modelo no es 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, lo más probable es que se trate del caso contrario, **sobreajuste** o **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.
- 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 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 a 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.

Por lo tanto, suena bastante conveniente que la potencia que la hace falta al aprendizaje automático para procesar grandes cantidades de datos, se la brinde la computación cuántica.

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 para comprobar si realmente son más eficientes que sus versiones clásicas y si es que tienen alguna limitación.

2.4.2. QSVM

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

Es un modelo muy potente y versátil, el cuál es capaz de realizar clasificaciones lineares o no lineares y regresiones.

En este método, situaremos nuestros datos en un plano, por ejemplo: Tenemos un conjunto de correos y queremos clasificarlos en spam o no. Según sus características se posicionará un punto por cada correo en un determinado lugar del plano. La maquina de vector soporte lo que hará sera dibujar una línea, llamada **hiperplano**, entre los correos que sean spam y los que no, con el mínimo número de errores, de forma que la distancia entre los puntos spam y los que no sea la mayor posible (ver Fig. **X)**.

En este caso asumimos que es un problema de clasificación binario, con tan sólo dos posibles soluciones, y que los datos son linealmente separables.

La mayor complejidad de este modelo reside en la optimización, ya que pese a ser bastante sencillo en general, es más difícil encontrar un hiperplano que clasifique todos los datos de forma correcta y además lo haga con el mayor margen entre los vectores posible.

El algoritmo posicionará cada punto del plano según una serie de parámetros (w) definidos de forma que las dos clases de datos queden a cada lado del hiperplano. Esta operación se puede definir como una fórmula matemática [9]:

$$f(x^m; w)y^m > 0$$

También se puede aplicar a otra serie de problemas donde los datos no sean linealmente separables añadiéndole funciones polinómicas, como se puede ver en la figura **X** [10].

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 computación cuántica este proceso se llama *preparación del estado* (ver Fig. X pag 140 libro supervised...) de un algoritmo de aprendizaje automático. Este proceso de adaptar los datos al hardware no se suele usar en computación clásica, aunque aquí es imprescindible.

A su vez, este proceso de conversión de datos forma parte del algoritmo de aprendizaje automático, por lo que aunque la ejecución sea más veloz en un ordenador cuántico, este paso aporta una complejidad extra.

En este caso, como queremos que los datos que reciba el algoritmo sean supervisados, es decir, que tengan una etiqueta dónde se muestra de qué clase son, deberemos también codificar estas etiquetas en qubits que estén en entrelazamiento con los qubits dónde se guarden los respectivos datos.[9]

Existen diversas técnicas de codificar los datos, como por ejemplo mediante amplitudes o la Hamiltoniana, pero para hacer más sencillo este proceso recurriremos a un, representado como: **Quantum Feature Map** o Mapa de funciones cuánticas $V(\Phi(x^{\rightarrow}))$.

$\Phi()$ es una función que aplicaremos a los datos normales, mientras que $V(\Phi(x^{\rightarrow}))$ es el circuito cuántico que se encargará de la transformación de los datos.

Debido a que encontrar un buen hiperplano dados los datos de entrada no es el método más óptimo, a menudo se hace uso de métodos **kernel**, que pueden ayudar a encontrar un hiperplano en un espacio vectorial de más dimensiones sin tener que definir dicho espacio de forma explícita.

Dicho de otra forma, 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 en problema de una forma no lineal. Se ve representada la diferencia entre usar un QSVM con kernel y sin kernel en la **Figura**

Una de las ventajas del QSVM frente al SVM clásico es precisamente el uso de funciones Kernel óptimas, ya que se puede dar que en ordenadores clásicos dichas funciones no se puedan estimar de forma eficiente, mientras que con la ventaja cuántica se facilitaría mucho el proceso.

2.4.3. QNN

Las redes neuronales artificiales cuánticas, al igual que las clásicas, fueron inspiradas por la arquitectura de los cerebros humanos. Científicos intentaron imitar los cerebros para crear una máquina que fuera capaz de pensar, de ser **inteligente**.

Las redes neuronales artificiales *(ANN 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. **??**

Una red neuronal se compone por un numero de dispositivos básicos de computación, a los que se les conoce como neuronas, y estas neuronas están conectadas unas con otras forma una gran red de comunicación. Las

aristas que unen los nodos (neuronas) unen el valor de salida, al cual llamaremos **peso**, de un nodo con el valor de entrada de otro, por lo que, la entrada de un nodo será obtenida cogiendo la suma de todos los pesos que apunten a dicho nodo.

Perceptrón

El perceptrón es la red neuronal más básica que existe.

Consiste de 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 de un límite concreto, entonces el resultado será la clase positiva, en caso contrario será la clase negativa. **??**

Debido a la simplicidad del perceptrón, este no nos permitirá resolver problemas complejos, pero en cambio, si juntasemos una gran cantidad de perceptrones divididos en diferentes capas, es 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 perceptrones, las cuales serán las *capas ocultas*, y una capa final la cual será las salidas. **ver Figura Multi-Layer Perceptron**.

El algoritmo utilizado para entrenar las redes neuronales es el **descenso del gradiente**.

El descenso del gradiente es el algoritmo de optimización genérico el cual es capaz de encontrar las soluciones óptimas para un gran número de problemas.

Consiste en que mide la función de error local, y guarda en un parámetro dicho valor, va iterando y en cada iteración modifica los parámetros del algoritmo, de forma que cada vez que se ejecuta, se calcule la función de error hasta que sea la mínima posible, y así de esta forma se habrá minimizado el coste de la función.

El descenso del gradiente se aplica a las redes neuronales usando una técnica de computación que lo calcula de forma automática, propagándose por todos los nodos de forma repetida hasta que se encuentra la mejor solución.

Las **redes neuronales cuánticas** funcionan de forma parecida. En la mayoría de casos se usan redes híbridas, es decir, parte del algoritmo se comportará exactamente igual que un red neuronal normal, pero por otro lado existirán ciertas capas ocultas que estarán parametrizadas en un circuito cuántico.

Estos circuitos estarán formados por puertas cuánticas que operan sobre los qubits que definen las capas mencionadas.

Estas puertas cuánticas se encargarán de rotar los estados de los qubits. De forma que los ángulos de rotación $h_1 y h_2$ se aplican a las puertas $R_1 y R_2$ en el circuito, y cambia los estados iniciales $|\psi\rangle_1$ y $|\psi\rangle_2$ **??**:

$$|\psi\rangle_3 = R(h_1)|\psi\rangle_1$$
$$|\psi\rangle_3 = R(h_1)|\psi\rangle_1$$

Al igual que en redes neuronales clásicas, en las cuánticas también se usa el algoritmo de descenso del gradiente mediante propagación a la hora de entrenar el modelo.

El gradiente de los pesos en una capa en concreto estará determinado por función de error con respecto a los pesos y ejecuciones en las capas anteriores. Si se asume que el objetivo de costo es C y que los parámetros

del circuito cuántico son el vector θ, puede calcular el gradiente del objetivo con respecto al uso de la regla de cambio de parámetros, como se demuestra aquí:

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

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

A la hora de medir los qubits, la información cuántica colapsa a información clásica que es la que se usa para predecir las salidas del algoritmo.

Capítulo 3

PSI: PLANIFICACIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE PLANIFICACIÓN

PSI

3.1. PSI 1: INICIO DEL PLAN DE SISTEMAS DE INFORMACIÓN

3.1.1. PSI 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 dónde 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. PSI 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 2. Aspectos Técnicos.

Para visualizar los resultados de las ejecuciones se creara 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 (ya sea simulador o ordenador cuántico)
- Menú de configuración donde se concretan ciertos aspectos extras de la ejecución.
- Menú 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 su versión clásica en local, y en versión cuántica en un simulador cuántico y ordenadores cuánticos reales para posteriormente analizar las diferencias entre todos ellos.

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. PSI 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. PSI 2: DEFINICIÓN Y ORGANIZACIÓN DEL PSI

3.2.1. PSI 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, donde los datos deban separarse por dos tipos de clase diferente. A este tipo se les llama Quantum Support Vector Classifier o Clasificador de vectores de soporte cuánticos.

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
- 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. 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 **no lo se aun**.

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 **no lo se**. Además, mostrará unas gráficas para poder visualizar el resultado final.

Objetivos de la fase:

- Hacer uso de Qiskit para la implementación
- Que el algoritmo **no lo se**
- 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 identificación, ni por lo tanto, de registro.

Objetivos de la fase:

- Que se pueda escoger entre usar QSVM o QNN
- Que se pueda escoger entre 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 ha 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 dispositivos disponibles y permitirle escoger uno
- Que se muestren los resultados obtenidos junto con las correspondientes gráficas

Capítulo 4

PSI 7: DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA

FASE DE PLANIFICACIÓN

PSI

Figura 4.1: Esquema de la infraestructura tecnológica

4.1. PSI 7.1: 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 ASI 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, tanto en simulador como real.
- 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, Qiskit, Strawberry Fields, 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.

4.2. PSI 7.2: 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 AYAX**: 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. EVS 4, 5, 6: ESTUDIO Y VALORACIÓN DE ALTERNATIVAS DE SOLUCIÓN. SELECCIÓN DE ALTERNATIVA FINAL

5.1.1. Sistema 1

5.1.2. Sistema 2

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:
  - Jefe de proyecto, Desarrollador del software, Tester, Arquitecto 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 el estudiante en este proyecto.

6.1.2.2. PBS (To-Do)

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

- Aplicación web
- Sistema desarrollado
  - Algoritmo QNN
  - Algoritmo QSVM
  - Datasets
  - Sistema de ejecución en simulador cuántico
  - Sistema de ejecución en ordenador cuántico real

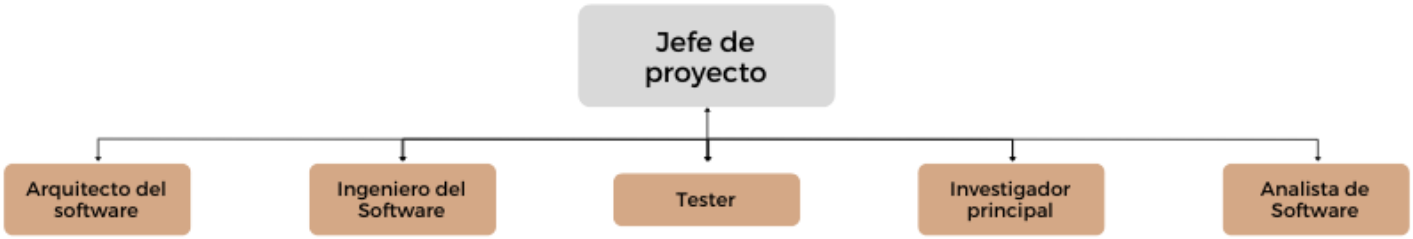


Figura 6.1: OBS

- Memoria del proyecto
- Planificación del proyecto
- Presupuesto del proyecto
- Plan de Gestión de Riesgos

6.1.3. Planificación Inicial. WBS

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 (ver 11.2)

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.

Disponibilidad de ordenadores cuánticos con un nº elevado de qubits

El objetivo de este proyecto es ejecutar una seria 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

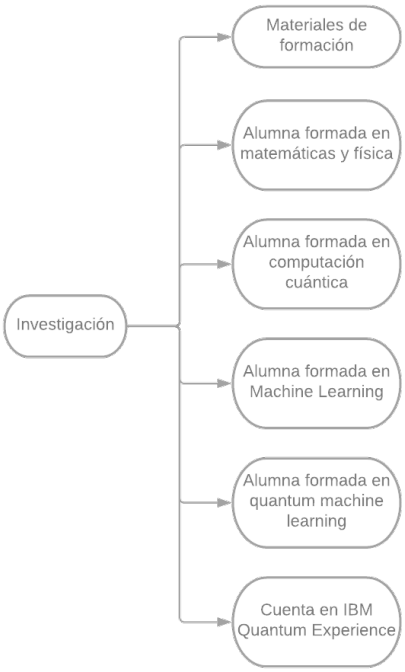


Figura 6.2: PBS Investigación

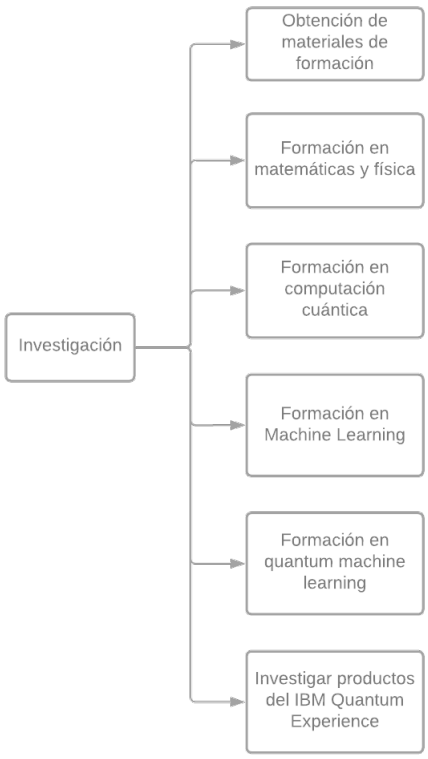


Figura 6.3: WBS Investigación

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.

Resultados poco orientativos por falta de potencia cuántica

Para que este estudio sea efectivo deberemos obtener diferencias en el tiempo de ejecución y resultados entre ordenadores cuánticos y ordenadores clásicos. Para que esta diferencia sea notable necesitaremos realizar las ejecuciones en unos ordenadores cuánticos con la potencia suficiente capaz de superar en potencia a un ordenador clásico.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Alta	Bajo	Bajo	Crítico	Crítico	0,63

- **Estrategia:** Mitigar el riesgo
- **Respuesta:** El sistema por defecto tendrá la opción de ejecutar en el ordenador con mayor potencia disponible.

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 decido 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 idónea 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.

Fiabilidad de los resultados de los simuladores cuánticos

La ejecución del sistema se realizará tanto en simuladores como en ordenadores cuánticos reales. Los simuladores cuánticos tienen un margen de fallo, por lo que puede haber disimilitudes con ejecuciones en ordenadores cuánticos.

Categoría	Probabilidad	Presupuesto	Planificación	Alcance	Calidad	Impacto
Técnico	Baja	Inapreciable	Inapreciable	Medio	Crítico	0,27

- **Estrategia:** Asumir el riesgo
- **Respuesta:** Los simuladores son proporcionados por empresas externas, por lo que la fiabilidad de los resultados va a depender de estos y no queda otra opción que asumir el resultado obtenido.

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 des-actualizado 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

6.1.5.1. Presupuesto de Costes

6.1.5.2. Presupuesto de Cliente

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 las 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 cuál 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

Idealmente relacionarla con la actualización de la planificación.

6.2.3. Riesgos

seguimiento de al menos 5 riesgos, con las hojas de riesgos correspondientes
12/06/2022

- Disponibilidad de ordenadores cuanticos con un nº elevado de qubit: El máximo permitido son 5 qubits
- Resultados poco orientativos por falta de potencia cu antic: con 2 qubits no podemos comprobar la eficacia en cuanto a tiempo, simplemente que funciona y ya
- Errores en las librerıas a usar: la libreria qiskit.aqua ha desaparecido y da errores
- Librerías en desarrollo: qiskit.terra está en desarrollo, por lo que la documentación es rara y escasa.
- Largos periodos de espera en la cola de ejecucion: hay ejecuciones que se pueden demorar hasta 20 minutos

6.3. CIERRE DEL PROYECTO

6.3.1. Planificación Final

Incluir la planificación final del proyecto realizado, haciendo un análisis de la evolución de la planificación

6.3.2. Informe Final de Riesgos

6.3.3. Presupuesto Final de Costes

6.3.4. Informe de Lecciones Aprendidas

Capítulo 7

ANÁLISIS DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

ASI

7.1. ASI 1: DEFINICIÓN DEL SISTEMA

7.1.1. Determinación del Alcance del Sistema

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

7.2. ASI 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.

ReqQSVM1.1. El sistema tendrá la capacidad de resolver problemas de clasificación binaria.

ReqQSVM1.2. El sistema tendrá la capacidad de resolver problemas de clasificación multiclase.

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 **digits**.

ReqQSVM3.4. El conjunto de datos proporcionado por la librería *qiskit* llamado **gaussian**.

ReqQSVM3.5. El conjunto de datos proporcionado por la librería *qiskit* llamado **iris**.

ReqQSVM3.6. El conjunto de datos proporcionado por la librería *qiskit* llamado **wine**.

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.**

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.

ReqQNN1.1. El sistema tendrá la capacidad de resolver problemas de clasificación binaria.

ReqQNN1.2. El sistema tendrá la capacidad de resolver problemas de clasificación multiclase.

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 **digits**.

ReqQNN2.4. 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.**

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 nº 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 tantos qubits cómo clases de datos haya en el conjunto de datos seleccionado por el usuario.

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 precisión alcanzado en la predicción mencionada en **ReqCuántica4.**.

Requisitos de la salida esperada

Salida1. El sistema mostrará un esquema del circuito ejecutado.

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 predicción 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

ametrización1. 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.

ametrización2. 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.

ametrización3. 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.

ametrización4. El sistema validará que el nº de ejecuciones sea un número natural.

ametrización5. El nº de ejecuciones por defecto de los algoritmos serán 1024.

Requisitos de la aplicación web

eqAppWeb-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 dato personal de los usuarios.

Requisitos de la API IBMQ

IBMQ1. El sistema hará uso de la API IMBQ 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 nº 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, estos serán 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**.

Por último, 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 este 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 imitar la ejecución de un dispositivo real. Los simuladores proporcionados, y usados por este sistemas serán imitaciones de ordenadores cuánticos de 32 qubits.

Los circuitos cuánticos que se ejecutan en estos simuladores pueden contener compuertas, mediciones, restablecimientos, condicionales y otras instrucciones personalizadas del simulador que se discutirán en otro cuaderno. [11]

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. El diagrama 7.1 contiene todos los casos de uso contemplados.

Ejemplo de tabla para especificación de casos de uso

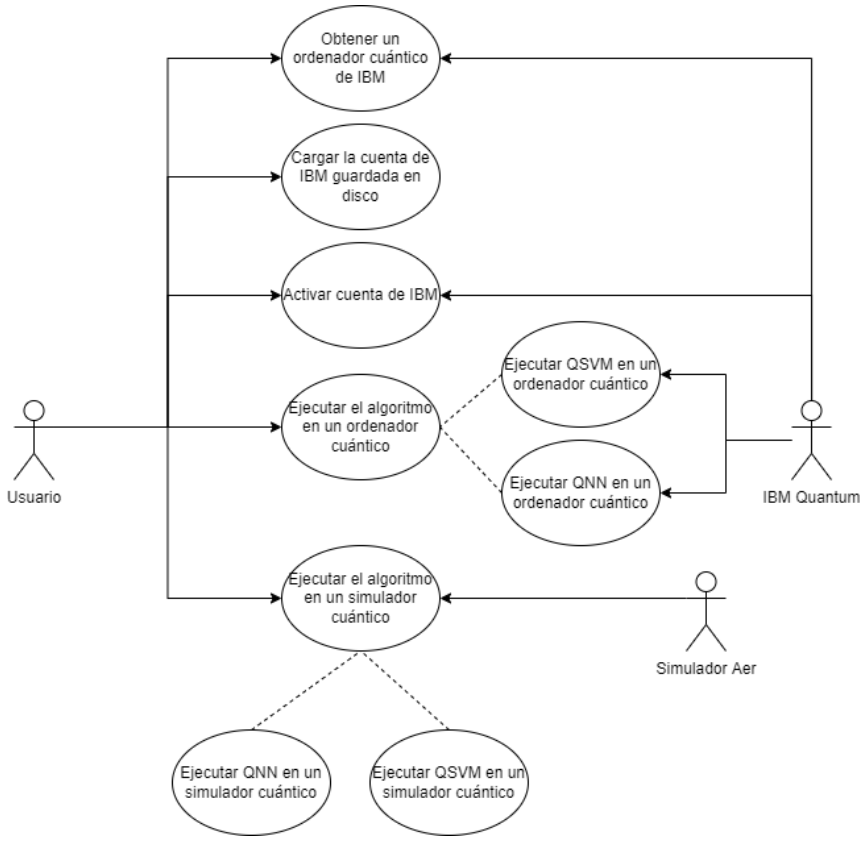


Figura 7.1: Casos de uso

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.

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 el algoritmo QSVM en un circuito cuántico 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 el algoritmo QSVM en un circuito cuántico 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 el algoritmo QNN en un circuito cuántico 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 el algoritmo QSVM en un circuito cuántico para ejecutarlo en un ordenador cuántico real.

7.3. ASI 3: IDENTIFICACIÓN DE SUBSISTEMAS DE ANÁLISIS

N/A

7.3.1. Descripción de los Subsistemas

N/A

7.3.2. Descripción de los Interfaces entre Subsistemas

N/A

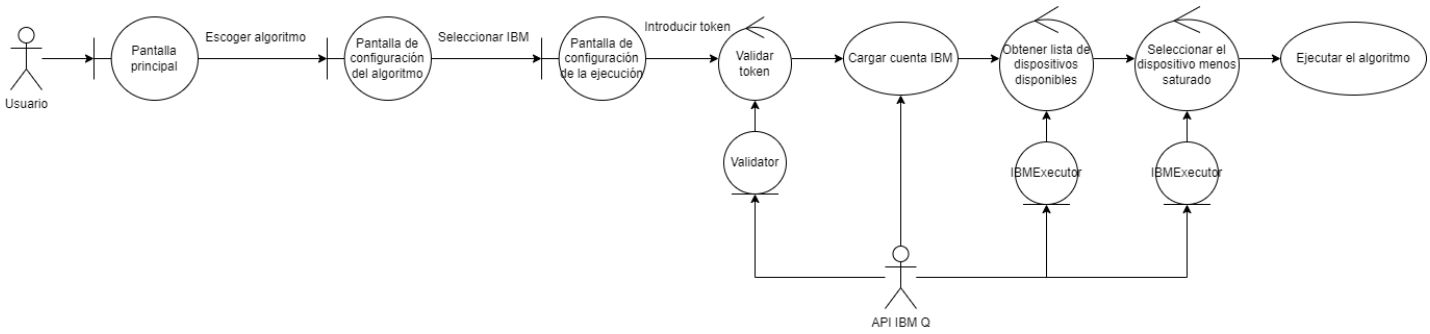


Figura 7.2: Diagrama de robustez - Obtener un ordenador cuántico de IBM

7.4. ASI 4: ANÁLISIS DE LOS CASOS DE USO

Caso de Uso - Obtener un ordenador cuántico de IBM Caso de Uso - Cargar la cuenta de IBM guardada en disco Caso de Uso - Activar cuenta de IBM Caso de Uso - Ejecutar QSVM en un simulador cuántico Caso de Uso - Ejecutar QSVM en un ordenador cuántico Caso de Uso - Ejecutar QNN en un simulador cuántico Caso de Uso - Ejecutar QNN en un ordenador cuántico

7.4.1. 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 **??** recoge el análisis de este caso de uso.

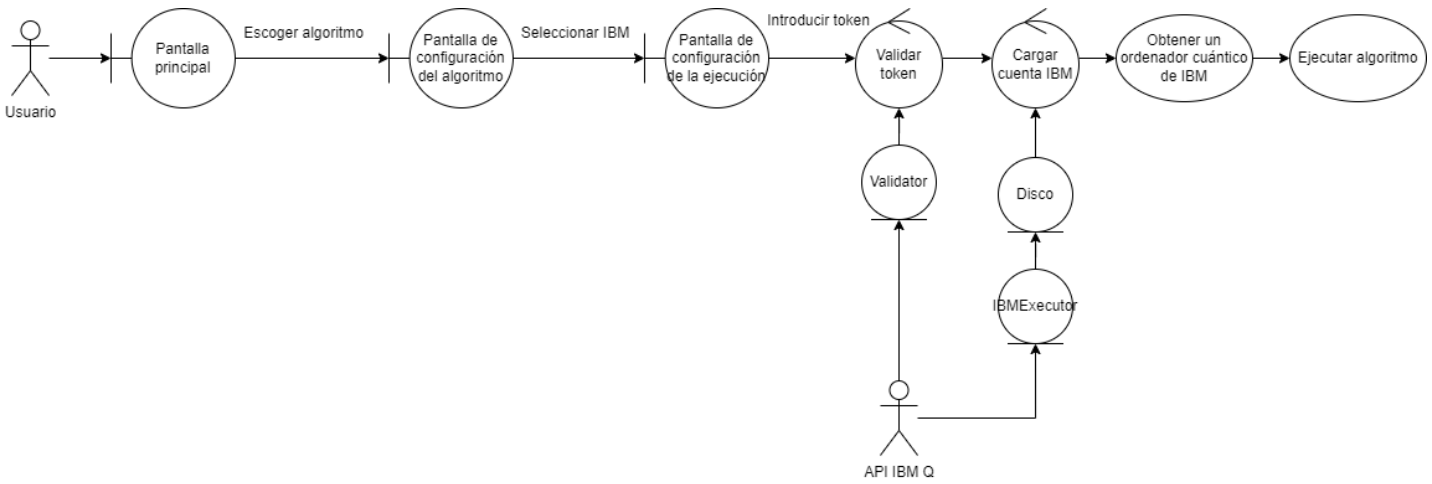


Figura 7.3: Diagrama de robustez - Cargar la cuenta de IBM guardada en disco

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	-

7.4.2. 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 **??** recoge el análisis de este caso de uso.

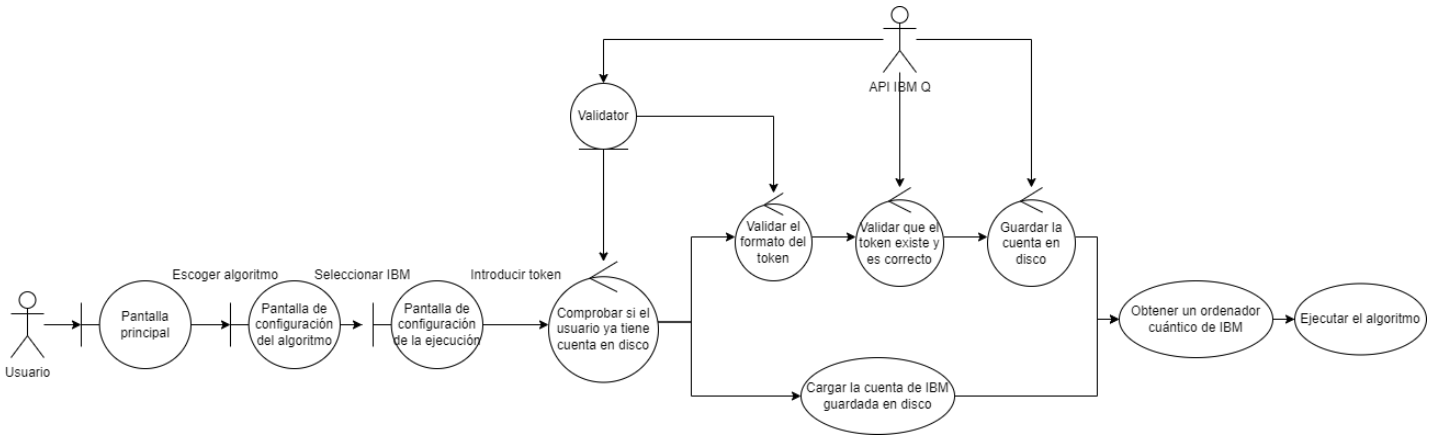


Figura 7.4: Diagrama de robustez - Activar cuenta de IBM

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	-

7.4.3. 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 **??** recoge el análisis de este caso de uso.

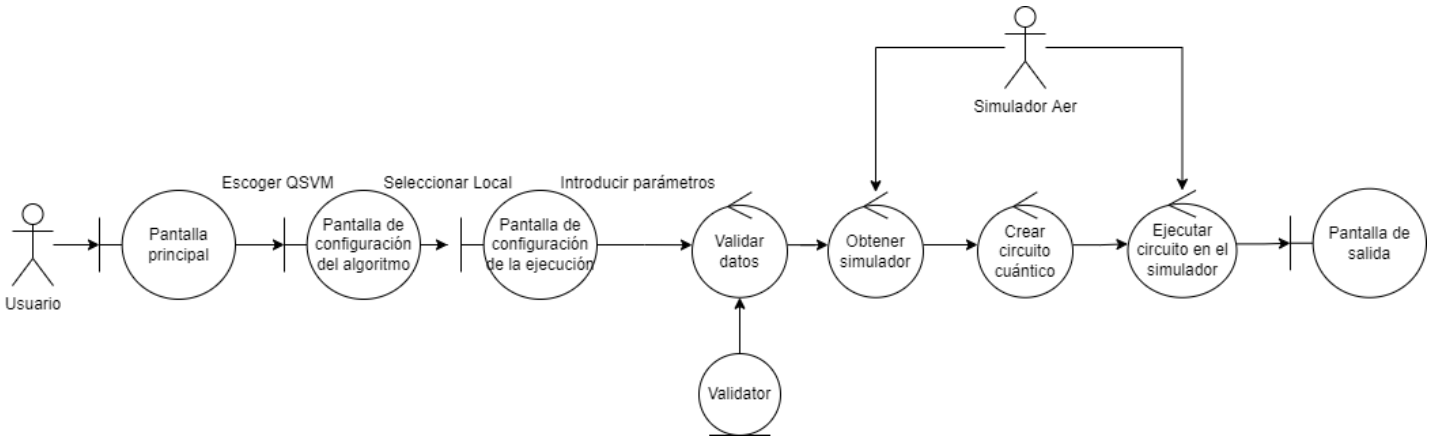


Figura 7.5: Diagrama de robustez - Ejecutar QSVM en un simulador cuántico

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	-

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

El diagrama de robustez de la figura 7.7 se recoge el correspondiente al caso de uso: Ejecutar QSVM en un simulador cuántico. La tabla **??** recoge el análisis de este caso de uso.

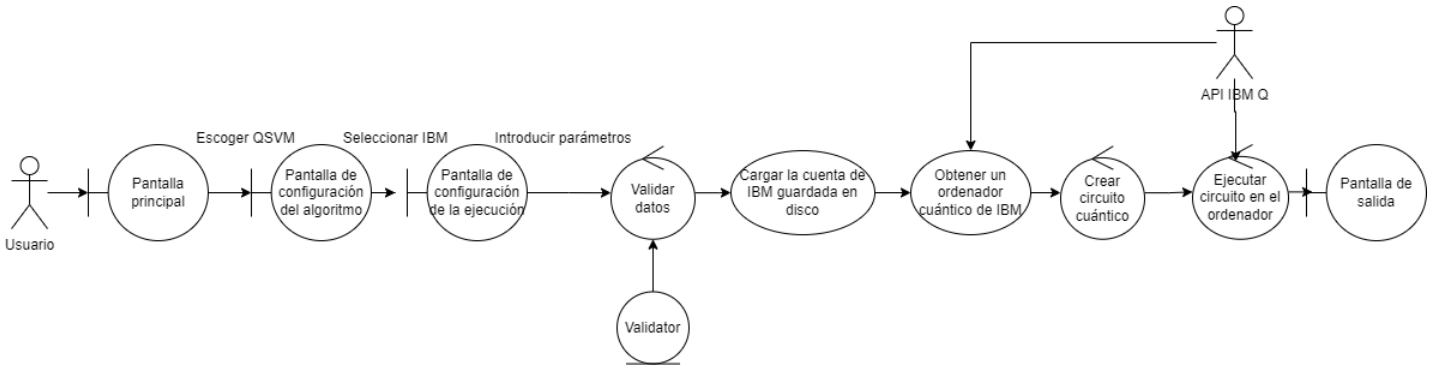


Figura 7.6: Diagrama de robustez - Ejecutar QSVM en un ordenador cuántico

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	-

7.4.5. 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 **??** recoge el análisis de este caso de uso.

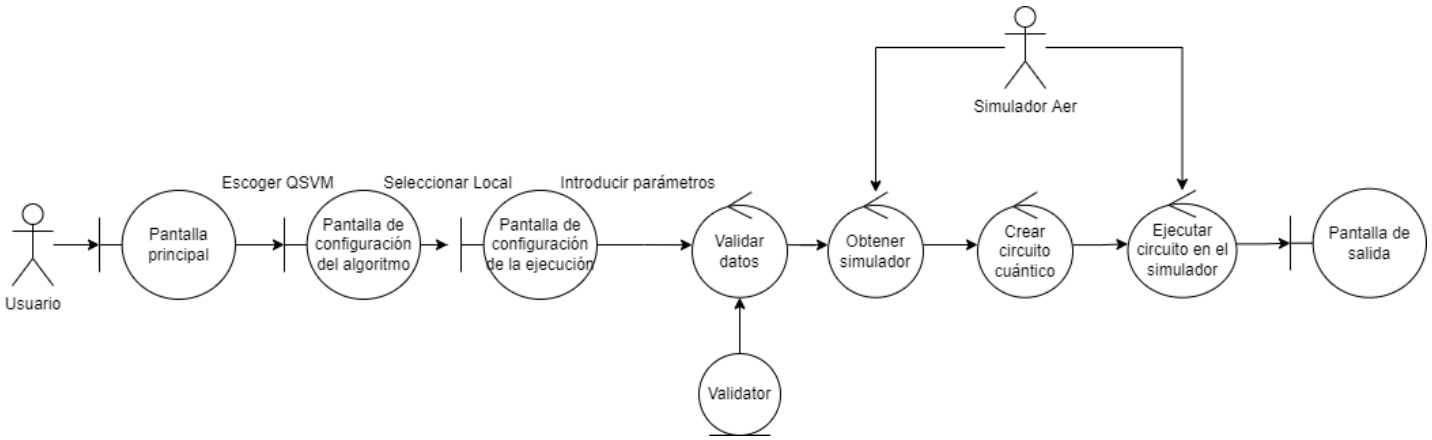


Figura 7.7: Diagrama de robustez - Ejecutar QSVM en un simulador cuántico

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	-

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

El diagrama de robustez de la figura 7.8 se recoge el correspondiente al caso de uso: Ejecutar QNN en un simulador cuántico. La tabla **??** recoge el análisis de este caso de uso.

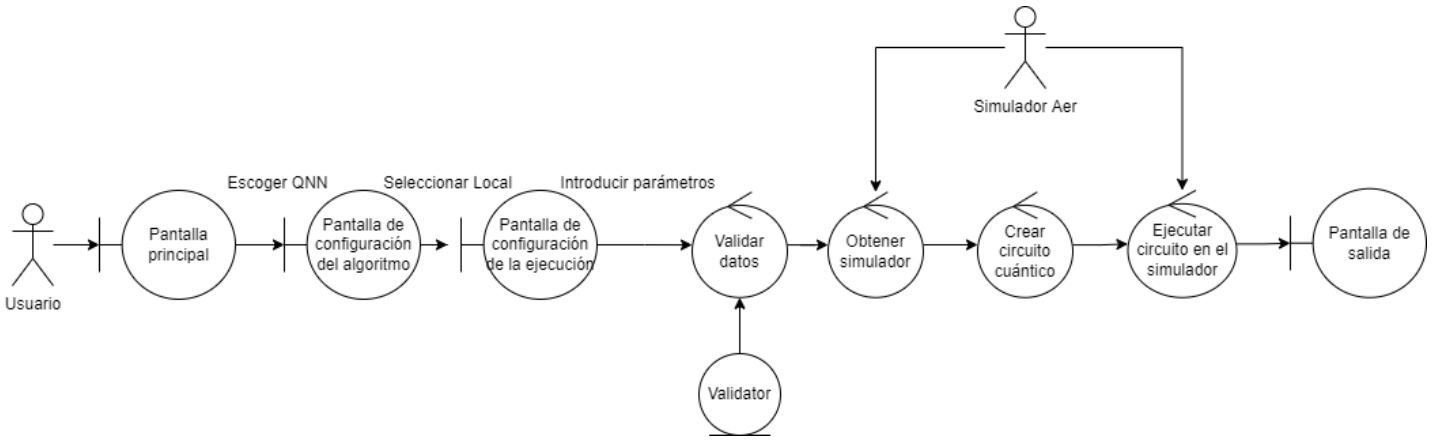


Figura 7.8: Diagrama de robustez - Ejecutar QNN en un simulador cuántico

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	-

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

El diagrama de robustez de la figura 7.9 se recoge el correspondiente al caso de uso: Ejecutar QNN en un ordenador cuántico. La tabla **??** recoge el análisis de este caso de uso.

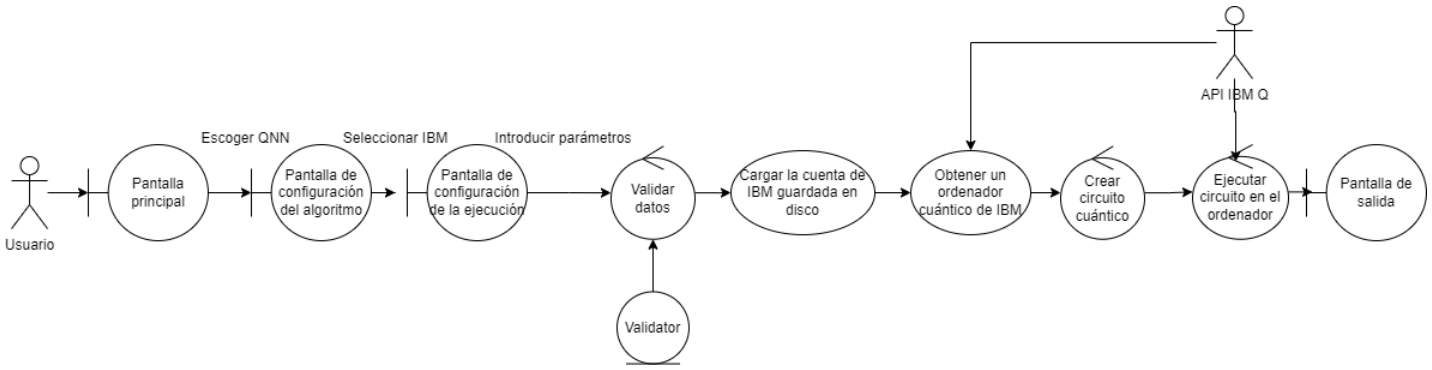


Figura 7.9: Diagrama de robustez - Ejecutar QNN en un ordenador cuántico

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 nº 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	-

7.5. ASI 5: ANÁLISIS DE CLASES

7.5.1. Diagrama de Clases

En la figura 7.10 se puede ver el Diagrama de Clases del sistema implementado.

7.5.2. Descripción de las Clases

En esta sección se describirá de forma breve la composición de cada clase representada en el apartado 7.5.1 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.

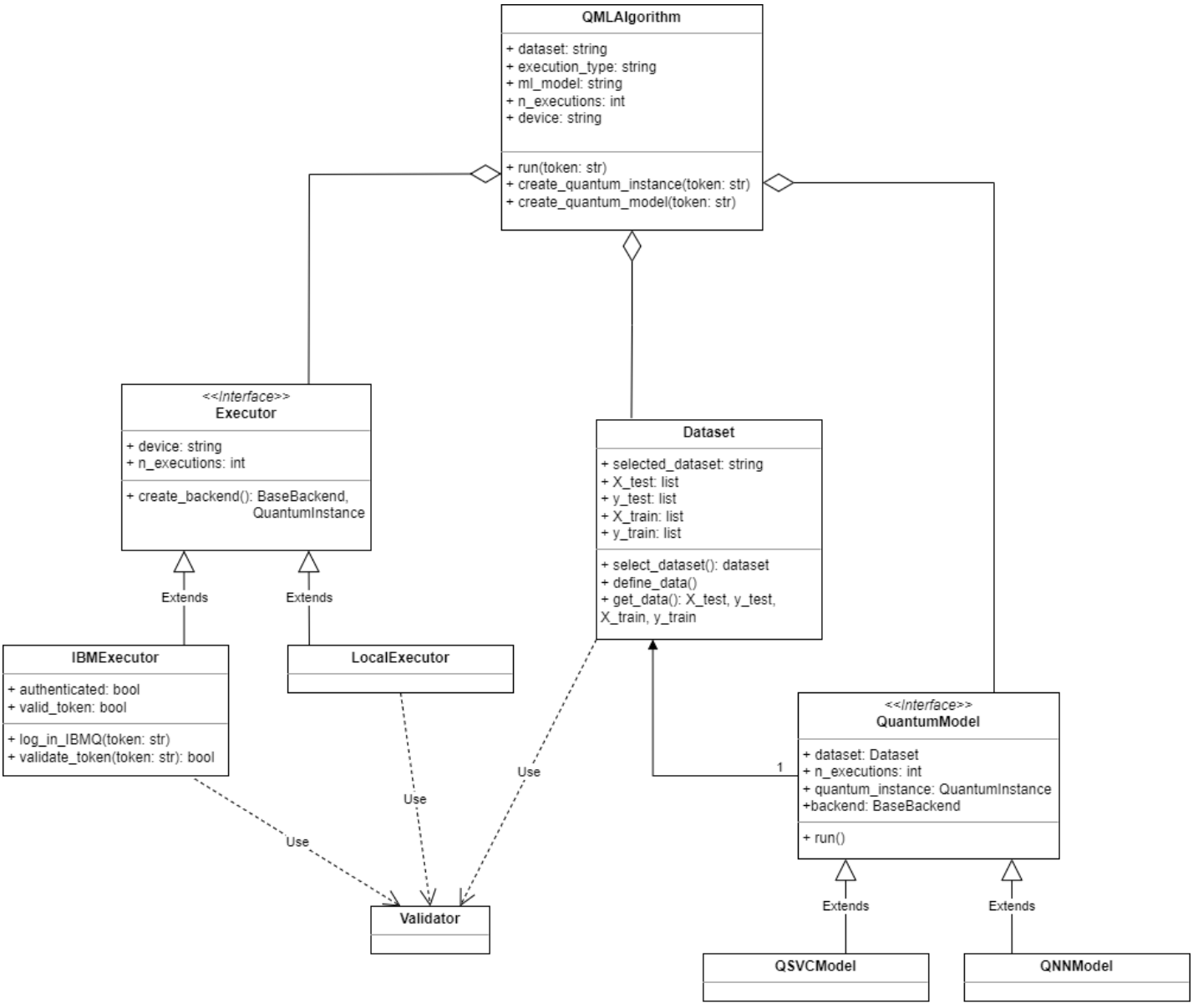


Figura 7.10: Diagrama de Clases

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 Ejecutor 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 devuelve el resultado obtenido.

Tabla 7.22: Descripción de diseño de QSVCModel

QSVCModel
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

7.6. ASI 8: DEFINICIÓN DE INTERFACES DE USUARIO

7.6.1. 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 mostraran 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 sólo en local.

7.6.1.1. Menú principal

7.6.1.2. Menú del algoritmo

7.6.1.3. Menú de configuración de la ejecución

7.6.1.4. Menú de ejecución

7.6.2. Descripción del Comportamiento de la Interfaz

7.6.3. Diagrama de Navegabilidad

7.7. ASI 10: ESPECIFICACIÓN DEL PLAN DE PRUEBAS

Capítulo 8

DISEÑO DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

DSI

8.1. DSI 3: DISEÑO DE CASOS DE USO REALES

8.1.1. Caso de Uso 1.1

8.1.1.1. Diagramas de Interacción (Comunicación y Secuencia)

8.1.1.2. Diagramas de Estados de las Clases

8.1.1.3. Diagramas de Actividades

8.1.2. Caso de Uso 1.2

8.2. DSI 4: DISEÑO DE CLASES

8.2.1. Diagrama de Clases

8.3. DSI 5: DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SISTEMA

8.3.1. DSI 5.1 Diseño de Módulos del Sistema

8.3.2. DSI 5.2 Diseño de Comunicaciones entre Módulos

8.3.3. DSI 5.3 Revisión de la Interfaz de Usuario

8.4. DSI 6: DISEÑO FÍSICO DE DATOS

8.4.1. Descripción del SGBD Usado

N/A

8.4.2. Integración del SGBD en Nuestro Sistema

N/A

8.4.3. Diagrama E–R

8.5. DSI 9: DISEÑO DE LA MIGRACIÓN Y CARGA INICIAL DE DATOS

N/A

8.6. DSI 10: ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS

8.6.1. Pruebas Unitarias

8.6.2. Pruebas de Integración y del Sistema

8.6.3. Pruebas de Usabilidad y Accesibilidad

N/A

8.6.3.1. Diseño de Cuestionarios

N/A

8.6.3.2. Actividades de las Pruebas de Usabilidad

N/A

8.6.4. Pruebas de Accesibilidad

N/A

8.6.5. Pruebas de Rendimiento

N/A

Capítulo 9

CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

CSI

9.1. CSI 1: PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONSTRUCCIÓN

9.1.1. Estándares y normas seguidos

9.1.2. Lenguajes de programación

9.1.3. Herramientas y programas usados para el desarrollo

9.2. CSI 2: GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES Y PROCEDIMIENTOS

Ejemplos de tablas descripción de clases

Tabla 9.1: Descripción de diseño de LoginScreen

LoginScreen	
Descripción	
Es la encargada de las acciones y la renderización de la pantalla de inicio de sesión.	
Atributos propuestos	
-	
Métodos propuestos	
signInWithGoogle	Hace una llamada al objeto Fire para el inicio de sesión con Firebase authentication mediante una cuenta de Google.
render	

Tabla 9.2: Descripción de diseño de HomeScreen

HomeScreen	
Descripción	
Es la encargada de las acciones y la renderización de la pantalla de emergencia.	
Atributos propuestos	
-	
Métodos propuestos	
componentWillMount	
emergencyCalling	Es el método encargado de redirigir la aplicación hacia el marcador con el 112 marcado.
warnProtectors	[Falta implementar] Es el encargado de generar un mensaje de aviso a los protectores creando notificaciones push.
render	

9.3. CSI 3: EJECUCIÓN DE LAS PRUEBAS UNITARIAS

9.4. CSI 4: EJECUCIÓN DE LAS PRUEBAS DE INTEGRACIÓN

9.5. CSI 5: EJECUCIÓN DE LAS PRUEBAS DEL SISTEMA

N/A

9.5.1. Prueba de Usabilidad

N/A

9.5.2. Pruebas de Accesibilidad

N/A

9.5.2.1. Revisión Preliminar

N/A

9.5.2.2. Evaluación de Conformidad

N/A

9.5.2.3. Checklist del WCAG 2.1

N/A

9.5.2.4. Accesibilidad con Dispositivos Móviles

N/A

9.6. CSI 6: ELABORACIÓN DE LOS MANUALES DE USUARIO

9.6.1. Manual de Instalación

9.6.2. Manual de Ejecución

9.6.3. Manual de Usuario

9.6.4. Manual del Programador

9.7. CSI 8: CONSTRUCCIÓN DE LOS COMPONENTES Y PROCEDIMIENTOS DE MIGRACIÓN Y CARGA INICIAL DE DATOS

Capítulo 10

IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA

FASE DE DESARROLLO

IAS

N/A

10.1. IAS 1: ESTABLECIMIENTO DEL PLAN DE IMPLANTACIÓN

10.2. IAS 4: CARGA DE DATOS AL ENTORNO DE OPERACIÓN

10.3. IAS 5: PRUEBAS DE IMPLANTACIÓN DEL SISTEMA

10.4. IAS 7: PREPARACIÓN DEL MANTENIMIENTO DEL SISTEMA

10.5. IAS 8: ESTABLECIMIENTO DEL ACUERDO DE NIVEL DE SERVICIO

10.6. IAS 9–10: PRESENTACIÓN Y APROBACIÓN DEL SISTEMA Y PASO A PRODUCCIÓN

Capítulo 11

CONCLUSIONES Y AMPLIACIONES

11.1. CONCLUSIONES

11.2. AMPLIACIONES

ANEXOS

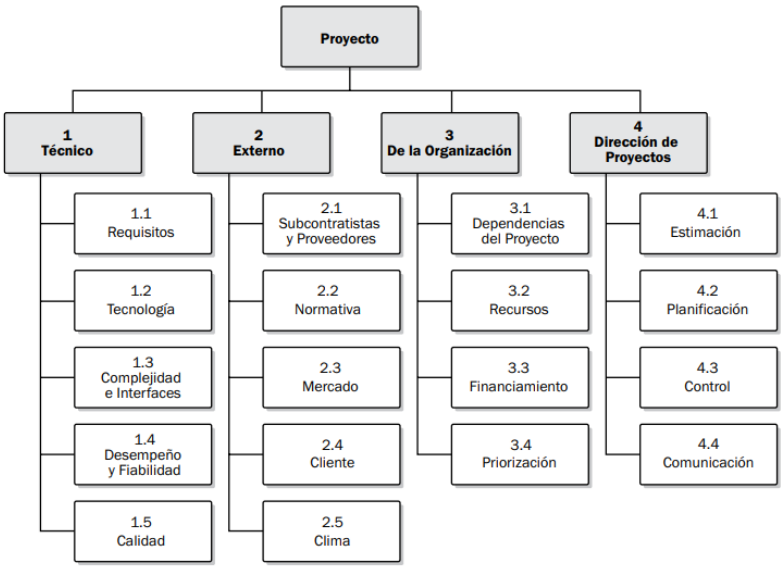


Figura 11.1: Clasificación de riesgos

PLAN DE GESTIÓN DE RIESGOS

Todo proyecto tiene riesgos asociados. El plan de gestión se 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[12], en la cuál se encuentran las siguientes fases en la gestión de riesgos:

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 porciento, 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 11.1.

La gran mayoría de riesgos se identifican al inicio del proyecto, pero también es cierto que algunos otros han aparecido en el trascurso 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:

Matriz de Probalididad 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 11.2: Matriz de probabilidad e impacto

- **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.

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 11.2.

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.

– 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, según la naturaleza y teniendo en cuenta la rentabilidad de este.

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.

CONTENIDO ENTREGADO EN LOS ANEXOS

Contenidos

Ejemplo de como especificar los contenidos entregados

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-GuardMe_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_paquetes.png*
  - *Diagrama_firestore.png*
  - *Diagrama_navegabilidad.png*
  - *Diagrama_secuencia_enviar.png*
  - *Diagrama_secuencia_visualizar.png*
  - *Diagrama_UML-Diseño.png*
  - *Diagrama_UML-Analisis.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 la cual esta dividida a su vez en dos carpetas:

AuthServerGuardMe

Contiene el código que se aloja en *Heroku* para darle funcionalidad al servidor. La clase principal es la llamada `mainAuthServer.js`.

GuardMe

Contiene el código fuente de la aplicación y se compone de las siguientes carpetas:

- **assets** -¿Carpeta que contiene los elementos gráficos usados en la aplicación. Se subdivide en una carpeta llamada *images* que contiene todas las imagenes utilizadas para la construcción de la aplicación.
- **components** -¿Carpeta que contiene el código para todos los componentes creados.
- **constants** -¿Carpeta que contiene el código
- **docs** -¿Carpeta que contiene los archivos html generados por JSDoc.
- **files** -¿Carpeta en la que se encuentras los futuros archivos de Términos y Condiciones y Política de Privacidad entre otros.
- **modules_LICENSES** -¿Carpeta que contiene una por una todas las licencias de las librerías utilizadas en el desarrollo.

- **navigation** -¿Carpeta que contiene las clases relativas a la navegación de la aplicación.
- **objects** -¿Carpeta que contiene los objetos utilizados en el desarrollo que en este caso ha sido solo Fire.js.
- **screens** -¿Carpeta que contiene todas las pantallas, agrupadas a su vez en subcarpetas que identifican la pantalla sobre la que están relacionadas.
- **styles** -¿Carpeta que contiene todos los estilos de las pantallas, agrupadas a su vez en subcarpetas que siguen la misma estructura que *screens*.
- **App.js** -¿Clase principal y encargada de que comience la aplicación entera.
- **LICENSE** -¿Licencia sobre el código fuente.
- **README.md** -¿Archivo con la descripción del proyecto para la documentación y el repositorio de GitHub.
- **package.json** -¿Archivo que contiene las librerías utilizadas en el proyecto.
- **app.json** -¿Archivo que contiene la configuración de la aplicación.
- **configJSDoc.json** -¿Archivo de configuración para la creación de documentación por parte de JSDoc.
- **Otros archivos** -¿Los demás archivos no son relevantes ya que muchos se generan por defecto y los demás son configuraciones propias de expo.

Bibliografía

[1] 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.

[2] 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.

[3] 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.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, ch. 2. Cambridge, UK: Cambridge University Press, 7 ed., 2010.

[5] S. Pattanayak, *Quantum Machine Learning with Python*. Bangalore, India: Apress Media LLC, 1 ed., 2021.

[6] V. Rodríguez Bouza *et al.*, "Sobre los cuaterniones, álgebras de lie, y matrices de pauli. teoría básica y aplicaciones físicas," 2012.

[7] Louis de Broglie, "The wave nature of the electron." https://www.nobelprize.org/uploads/2016/04/broglie-lecture.pdf, 1929. Online; accessed 20 Mar 2022.

[8] A. Einstein, B. Podolsky and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?," *Physical Review*, vol. 47, no. 777, p. 3, 1935.

[9] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers. Machine Learning*. Springer, 2018.

[10] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly, 2 ed., 2019.

[11] Qiskit, "Documentación simuladores qiskit." https://qiskit.org/documentation/locale/es_UN/tutorials/simulators/1_aer_provider.html, 2022. Online; accessed 17 Jun 2022.

[12] PMI, *Guía de los fundamentos para la dirección de proyectos (guía del PMBOK®)*. Newtown Square, Pensilvania: Project Management Institute, 5 ed.

[13] 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.

[14] C. Bernhardt, *Quantum Computing for everyone*. Cambridge, MA: The MIT Press, 1 ed., 2019.

[15] Louis de Broglie, "The wave nature of the electron." `https://www.nobelprize.org/uploads/2016/04/broglie-lecture.pdf`, 1929. Online; accessed 20 Mar 2022.