QCRAFT Quantum Developer Interface: A Tool for Continuous Deployment of Quantum Circuits

Javier Romero-Alvarez[a], Jaime Alvarado-Valiente[b], Enrique Moguel[c], Jose Garcia-Alonso[d] and Juan M. Murillo[e]

Quercus Software Engineering Group, Universidad de Extremadura, Av. de la Universidad s/n, 10003, Cáceres, Spain

Keywords: Quantum Computing, Quantum Software Engineering, Quantum Circuits, Quantum Services, Quantum Web Interface.

Abstract: Quantum computing is evolving quickly, with increasing demand for accessible tools to design, execute, and analyze quantum circuits. However, the lack of standardization and interoperability between different quantum platforms, such as the IBM Quantum Platform and Amazon Braket, presents challenges including a dependence on the platforms. This work introduces the QCRAFT Quantum Developer Interface, a web interface for the Continuous Deployment and execution of quantum circuits in the form of services. By integrating tools like Quirk for circuit design and Docker containers for environment consistency, this tool enables developers to design quantum circuits, store them in databases, and execute them on various quantum platforms with minimal adjustments. This process simplifies and automates quantum deployment workflows, offering an accessible and modular solution for quantum researchers and developers.

1 INTRODUCTION

Quantum computing, an emerging field based on the principles of quantum mechanics, is set to revolutionize the way we tackle complex computing problems. Theoretically, it can perform computations that would otherwise take classical computers an impractical time (Zhao, 2020).

Therefore, leading technology companies such as IBM, Google, Microsoft, and Amazon have recognized the transformative potential of quantum computing and are making substantial investments in both hardware and cloud-based quantum computing services (AbuGhanem, 2025; Arute et al., 2019). IBM, for instance, has developed an ambitious roadmap to increase the number of qubits available in its quantum processors over the coming years, with the aim of making quantum computing widely accessible and practical for a wider user base[1]. These companies have also created cloud platforms, such as IBM Quantum Platform[2] and Amazon Braket[3], that allow researchers and developers worldwide to experiment with quantum computing without needing their own quantum hardware. However, the diversity of programming languages and frameworks used on these platforms results in a fragmented environment, which complicates the development and deployment of quantum applications (Alvarado-Valiente et al., 2023b).

At the core of quantum computing is the concept of quantum circuits. Unlike classical circuits, they exploit superposition and entanglement to process multiple solutions simultaneously, which makes them especially powerful for Bounded-error Quantum Polynomial time (BQP) class problems, including cryptography (Shor, 1994) and optimization (Chicano et al., 2025).

However, the design and deployment of quantum circuits remain complex and challenging. Quantum computing platforms are highly specialized and often rely on unique programming languages and tools. For example, IBM's Qiskit SDK and Amazon's Braket

[a] https://orcid.org/0000-0002-3162-1446
[b] https://orcid.org/0000-0003-0140-7788
[c] https://orcid.org/0000-0002-4096-1282
[d] https://orcid.org/0000-0002-6819-0299
[e] https://orcid.org/0000-0003-4961-4030

[1] https://research.ibm.com/blog/ibm-quantum-roadmap

[2] https://quantum.ibm.com

[3] https://aws.amazon.com/braket

Romero-Alvarez, J., Alvarado-Valiente, J., Moguel, E., Garcia-Alonso, J. and Murillo, J. M.
QCRAFT Quantum Developer Interface: A Tool for Continuous Deployment of Quantum Circuits.
DOI: 10.5220/0013533200004525
In *Proceedings of the 1st International Conference on Quantum Software (IQSOFT 2025)*, pages 89-96
ISBN: 978-989-758-761-0

SDK each use proprietary approaches to quantum circuit design and execution (Abraham et al., 2019). In this context, deployment refers to the process of defining quantum circuits as services and making them available for execution on quantum hardware. Consequently, developers need to adapt their quantum circuits to work with different platform-specific tools, which limits interoperability and creates inefficiencies. Furthermore, the field lacks standardized development environments for circuit design, debugging, and deployment, making it difficult to reuse quantum circuits across platforms or share them within the research community (Moguel et al., 2022). This fragmentation poses a significant barrier to entry and hinders collaborative progress in quantum computing.

To address these challenges, this work aims to answer the following research questions:

1. How can we design a unified framework that facilitates the seamless development, deployment, and management of quantum circuits across different quantum platforms?
2. How can modularity, cross-platform compatibility, and environment consistency be integrated effectively to improve the accessibility and abstraction of quantum software development workflows?

The need for standardized, cross-platform solutions has become apparent as more researchers and developers seek efficient ways to leverage quantum computing resources from various platforms. An ideal solution would simplify the quantum development workflow by enabling quantum circuits to be designed, stored, executed, and managed within a single, cohesive environment, thus minimizing compatibility issues and streamlining the deployment process (Murillo et al., 2025).

To do this, we propose the QCRAFT (Quantum Circuit Research and Framework Toolbox) Quantum Developer Interface, a tool that enables developers to design, store, deploy, and execute quantum circuits seamlessly across different quantum computing platforms, such as Amazon Web Services (AWS) and IBM, in the form of services.

With features such as cross-platform compatibility, automated deployment through Docker[4], secure credential management, and visual circuit design integrating Quirk[5], this tool offers a practical solution for Continuous Deployment (CD) and execution of quantum circuits as services.

[4]https://www.docker.com

[5]https://algassert.com/quirk

2 BACKGROUND

2.1 Programming Tools for Quantum Circuits

With the growth of quantum computing, several programming tools and platforms have emerged to facilitate the development, testing, and deployment of quantum circuits.

These tools often cater to the unique requirements of quantum algorithms and hardware constraints, offering specialized functionalities to support quantum software development. Various research efforts have proposed architectures specifically tailored for programming quantum systems, recognizing the need for structured development environments similar to those available in classical computing.

For example, a systematic review by (Khan et al., 2023) explores software architectures in quantum computing, providing a framework to design and implement quantum systems with a focus on modularity, reusability, and adaptability. Their research underscores the necessity of dedicated quantum programming environments that address the challenges associated with hardware-specific requirements and computational constraints, which are characteristic of quantum circuits. This type of architecture is essential for managing complex quantum workflows and enabling collaboration across development teams.

In terms of programming methods, some research has been directed toward designing quantum circuits under specific hardware constraints. (Hirata et al., 2011) proposed a method to transform quantum circuits into a linear nearest-neighbor architecture, optimizing them for physical implementations that impose spatial constraints on qubits. This approach is especially useful in quantum devices where interactions between qubits are restricted by proximity, such as certain superconducting qubit systems. Although limited to specific types of quantum hardware, this method represents an important step toward more efficient and adaptable quantum circuit design.

In addition to contributing to the field, (Hevia et al., 2022) introduced QuantumPath, a quantum software development platform that aims to streamline the creation, management, and deployment of quantum applications across different quantum hardware back-ends. QuantumPath supports a range of quantum programming languages and provides an Integrated Development Environment (IDE) with built-in tools for circuit design, testing, and deployment. Moreover, it addresses the complexities of managing multi-language quantum software projects, supporting developers in creating reusable and modular code

components.

Another relevant approach to the development of quantum programming tools is Classiq (Minerbi, 2022), a platform that focuses on the automatic generation of quantum circuits using high-level synthesis techniques. Classiq highlights by integrating different quantum platforms and providing advanced tools, such as visualizations at multiple levels of abstraction.

Moreover, (Stirbu et al., 2024) present a framework that uses Kubernetes to manage hybrid applications that combine quantum and classical components.

In comparison to these works, QCRAFT Quantum Developer Interface differentiates itself by offering a unique Docker integration, encapsulating each quantum circuit as a separate service within a Docker container, unlike other work such as the Kubernetes work that encapsulates entire resources. This ensures that specific dependencies and configurations remain constant across different environments. Moreover, the proposal focuses on the implementation of Continuous Deployment workflows to quantum services, which allows automating the lifecycle of the development and deployment of quantum circuits as services.

Among the many tools available, the quantum circuit simulator selected in this work has been Quirk, as it is an open-source tool. Quirk provides a drag-and-drop interface that simplifies the process of creating and manipulating quantum circuits. This ease of use makes it accessible to both novice and experienced developers, facilitating experimentation with quantum gates and measurements. The accessibility of Quirk has established as a widely-used tool for prototyping and education in the quantum computing community (Serrano et al., 2022). Its flexibility and visual representation of quantum operations make it ideal for developers who are exploring quantum algorithms without needing extensive knowledge of the underlying hardware. For this reason, in this work Quirk has been adapted to enable developers to make a visual design of quantum circuits.

2.2 CI/CD in Quantum Computing

As quantum computing matures, the demand for efficient deployment and management solutions in this domain has grown, making the adoption of DevOps methodologies highly relevant (Alvarado-Valiente et al., 2023a).

DevOps, an approach that combines "development" and "operations", is geared toward automating and streamlining the software lifecycle, enabling Continuous Integration (CI), Continuous Deployment (CD), and rapid feedback loops. In conventional software engineering, these practices improve collaboration, accelerate release cycles, and ensure consistent deployment across environments. Applying these practices to quantum computing could provide similar benefits, addressing the unique challenges associated with managing and deploying quantum circuits in diverse hardware architectures (Murillo et al., 2025).

By implementing DevOps practices, developers can manage the deployment lifecycle of quantum circuits more efficiently, ensuring consistency across platforms (Gheorghe-Pop et al., 2020). Tools such as Docker for environment consistency, and CI/CD platforms like Jenkins or GitHub Actions can be instrumental in achieving this goal (Romero-Álvarez et al., 2023).

In a quantum context, CI/CD principles such as automated testing, version control, and environment standardization are essential for creating a stable and scalable development pipeline (Díaz et al., 2025). Quantum computing requires careful handling of deployment environments, as different platforms impose specific requirements regarding qubit configurations, gate sets, and performance constraints.

In this context, research in quantum computing has increasingly focused on adapting classical DevOps practices, such as CI/CD and environment consistency, to meet the unique requirements of quantum applications (Romero-Alvarez et al., 2023). For example, the work by (Kourtis et al., 2024) proposes a comprehensive framework for quantum DevOps, which extends traditional DevOps practices to accommodate the unique requirements of quantum computing. Their framework incorporates the use of Qrisp[6], a high-level programming language that allows developers to create quantum circuits without dealing with low-level details, such as manipulating qubits and quantum gates. This simplifies the coding process and improves accessibility for programmers.

Another study by (Nguyen et al., 2024), proposes the Quantum Function-as-a-Service (QFaaS) framework, which presents a quantum software lifecycle that includes seven stages for quantum function development, facilitating the integration of DevOps practices into the quantum programming workflow. In addition, it implements a strategy for automatically selecting the most suitable back-end for executing quantum computing parts, thus optimizing performance and efficiency.

In contrast to these approaches, which focus on a serverless model for deploying quantum functions, the QCRAFT Quantum Developer Interface advances quantum DevOps by providing a fully integrated platform specifically designed to support the CD and

[6]https://qrisp.eu

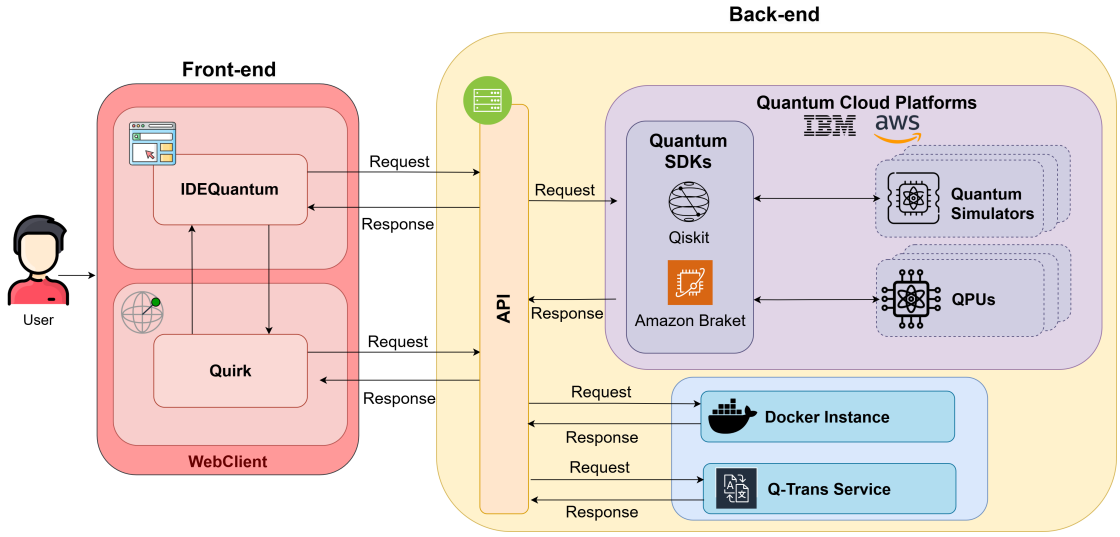

Figure 1: CD Architecture of QCRAFT Quantum Developer Interface.

cross-platform execution of quantum circuits. It centers on cross-compatibility and modularity, enabling developers to seamlessly switch between hardware providers such as IBM Quantum Platform and Amazon Braket without modifying their circuits. Furthermore, the tool uses Docker for containerization to ensure environment consistency across platforms.

3 QCRAFT QUANTUM DEVELOPER INTERFACE

QCRAFT Quantum Developer Interface provides an intuitive interface in which developers can design quantum circuits visually through Quirk, a drag-and-drop circuit builder. Quirk allows developers to experiment with quantum gates and qubits, making it accessible to developers of varying expertise levels. The tool also integrates with cloud-based quantum platforms, specifically, AWS and IBM, supporting seamless transitions between hardware environments. This integration eliminates the need for manual adjustments or recoding circuits, ensuring interoperability and efficient cross-platform compatibility.

To improve replicability, all code and data used in this work are available in a public Zenodo repository[7].

The CD architecture of the tool can be seen in Figure 1, where two components can be distinguished: front-end (red) and back-end (yellow).

Starting with the front-end, this is the part of the tool with which users will directly interact, accessing it through any web browser. In this case, it is mainly composed of two parts: *IDEQuantum* and *Quirk*. *IDEQuantum* comprises all the functionalities available to the user, except the design of the quantum circuit, which is exclusively managed by the *Quirk* component. The functionalities available to the user in *IDEQuantum* include the complete management of circuits, their execution, the visualization of obtained results, the management of credentials, and the deployment of circuits as services, as can be seen in Figure 1. These functionalities will be explained in more detail in the following sections.

The back-end is responsible for processing all requests based on the operations performed by the users on the front-end. This back-end includes an *API* as its main component, containing all the methods required to ensure the proper functioning of the different features. Moreover, it integrates connections with the specific SDKs of AWS and IBM, allowing the execution of designed circuits.

The back-end also provides the essential services needed to complete the functionalities of the tool. Specifically, it offers a real-time translation service of circuits generated in Quirk to code compatible with the different AWS and IBM frameworks, such as Qiskit, QASM or Amazon Braket code, called the Q-Trans Service (Alvarado-Valiente et al., 2024). Therefore, it acts as a translator, adapting the quantum circuits to the specific set of gates and the topology of each hardware platform. This enables users to view or copy the code implementation of the circuits they have designed, and, within the tool, to execute those circuits on each platform or deploy them as services. Furthermore, it supplies the necessary infrastructure to deploy circuits as services, encapsulating them in Docker containers to ensure isolation and control.

[7]https://doi.org/10.5281/zenodo.14745283

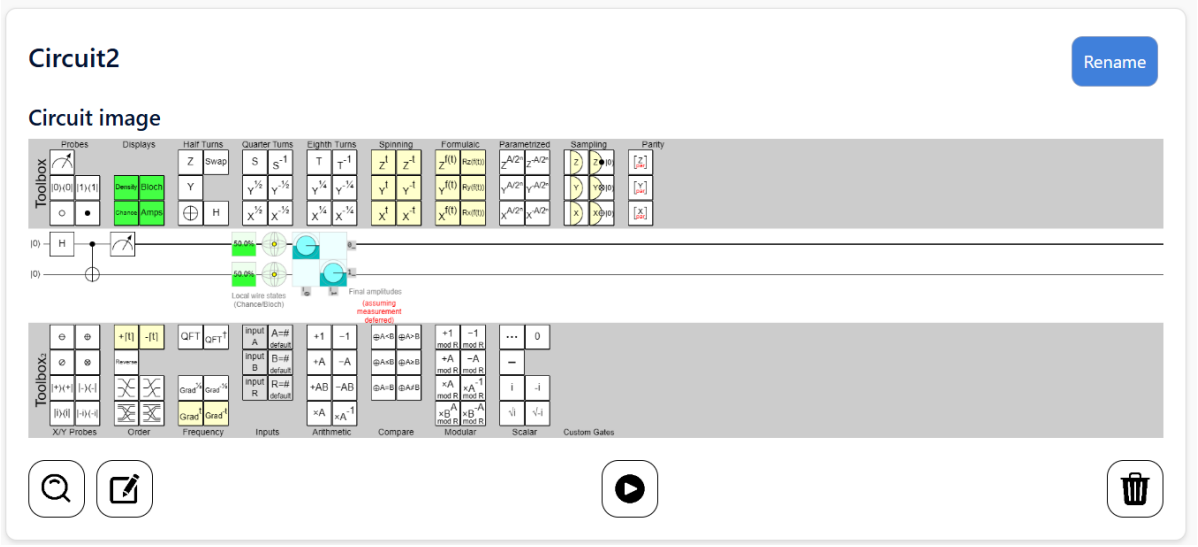

Figure 2: Screenshot of QCRAFT Quantum Developer Interface. Circuit design view.

3.1 Integration and Adaptation of Quirk in the Web Application

The Quirk tool has some limitations such as the lack of translation into executable code, user management, or a suitable way of storing the designed circuits, being the URL of the circuit or a JSON file the only way to do it.

To this end, the first adaptation implemented in the tool was to add a new button, called the *Translate circuit*, which redirects the user to the interface of the *IDEQuantum* component. This component, once it has received the circuit, connects to the translation service of the external server and displays the code of the circuit to the user for each platform.

This allows the *IDEQuantum* component to process the circuit and store it in the user profile, which led to the second adaptation that has been made to the Quirk tool, to allow the editing of a previously designed and stored circuit. To do this, the tool was adapted to allow the opposite flow, receiving a circuit from *IDEQuantum*, viewing it, editing it, and returning it for updating.

To this end, a new button, called Edit Circuit, has been added, which allows that once the circuit has been received and edited by Quirk, it can be returned to the *IDEQuantum*. These modifications, which allow the creation and edition of circuits, are completed by *IDEQuantum*, where from its interface it is possible to see the visualization of the circuits in Quirk, and also to launch the edition, execution, or elimination of each one of them, as can be seen in Figure 2.

3.2 Security and Credential Management

The QCRAFT Quantum Developer Interface also addresses security through a credential management system that securely handles the credentials of AWS and IBM, as shown in Figure 3.

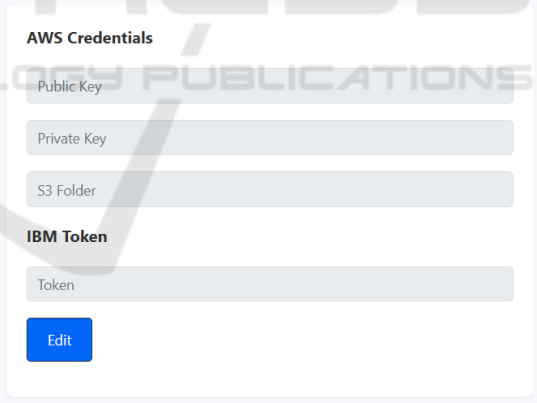

Figure 3: Security credentials example.

Security protocols are integrated into the deployment pipeline to ensure that user credentials are stored, accessed, and managed safely. This is essential for CD, where deployment workflows must maintain strict security standards, particularly when interfacing with external platforms.

These credentials can be specified in the profile menu of the user and can be used to launch the different executions in the simulators or Quantum Processing Units (QPUs) available on each platform.

3.3 Execution and Results

From the tool, once the user has designed the circuits and configured his credentials, the execution of the circuits on the available simulators and QPUs can be performed.

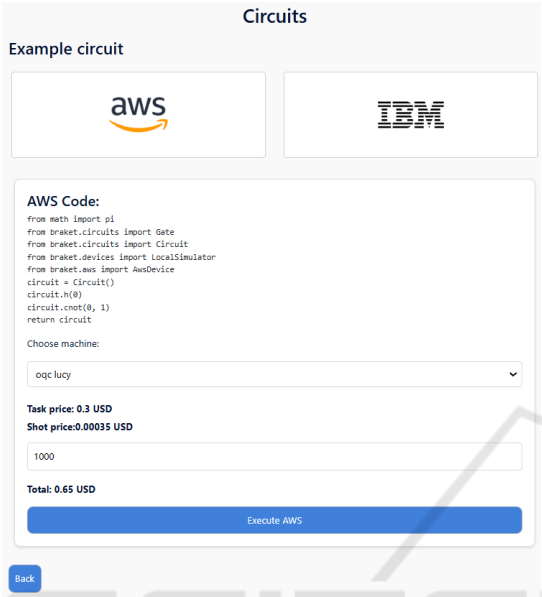

Figure 4: Interface for visualization of the circuit code at the time of its execution.

To do it, as can be seen in Figure 4, the user has the option to select a platform, the device on which he wants to launch the execution, and the number of shots.

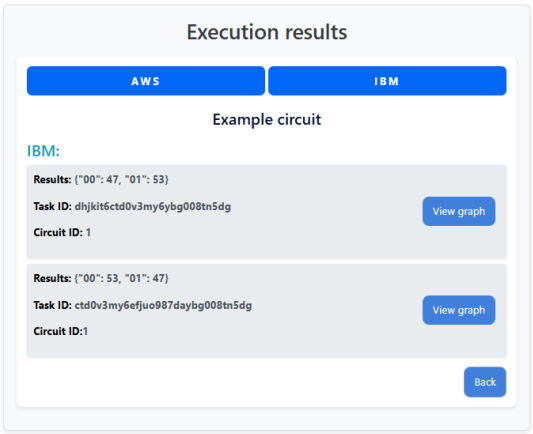

Figure 5: List of the results of a circuit after execution.

Additionally, for each platform, there is a real-time cost estimate of the execution price that takes into account the number of shots to be executed or the estimated execution time.

When the user has executed a circuit, the option is enabled to view the results of the different executions that have been executed. To do so, the user accesses a window that splits the executions by provider, as can be seen in Figure 5. In this window, once one of the available providers is selected, a list of launched executions appears, allowing the user to visualize the results graphically. In case it is still in queue or has failed, it offers this information at the moment.

The graphical visualization (Figure 6) of the results obtained in the execution shows the distribution of the results according to the number of shots.

Figure 6: Graphic view of a circuit already executed.

3.4 Continuous Deployment of Quantum Services

The CD architecture allows developers to automate the design-to-deployment pipeline for quantum circuits. Once a circuit is designed, it can be automatically pushed through a CI/CD pipeline. This process minimizes manual intervention, allowing for faster, more reliable deployment and execution of quantum circuits across supported platforms.

Using Docker containers, the tool provides a modular infrastructure for deploying these quantum circuits in the form of services. The Docker encapsulates the environment of each circuit, managing dependencies and configurations to prevent compatibility issues when transitioning from one quantum platform to another. This modularity improves the consistency of the environment, enabling easier debugging and facilitating the automated deployment of quantum services on quantum platform. To launch the CD of a circuit as a quantum service, the user has available a button, called Deploy service, in the same window of normal execution. Once deployed, the endpoint where the service is available is displayed on the screen, ready to be consumed or integrated by the user.

3.5 Use Case and Connection with AWS and IBM

As a use case for the tool, connections have been made to the two aforementioned cloud platforms, AWS and IBM. The connection to AWS is made through the use of its SDK, which allows the use of on-demand simulators and QPUs provided by Amazon Braket. AWS offers several simulators, including the free local Amazon Braket SDK simulator and three on-demand simulators: State Vector 1 (SV1), Density Matrix 1 (DM1), and Tensor Network 1 (TN1). Each of these simulators has specific features that allow for the execution and testing of quantum circuits with different capabilities and performance levels. The cost of using these on-demand simulators is based on the duration of each simulation task, billed per minute in one-millisecond increments, with a minimum billing duration of 3 seconds per simulation. In addition, AWS also offers hybrid jobs that allow simulators to be integrated into the same container as the application code, which can affect the duration of the use of the hybrid job instance.

To run circuits on the real QPUs that are offered by Amazon Braket, it is necessary to provide the appropriate credentials, as explained above, and the cost is calculated based on the number of shots and the selected QPU. On the one hand, for simulators, costs are based on the duration of each simulation task, billed per minute in one-millisecond increments, with a minimum billing duration of 3 seconds. On the other hand, using an on-demand QPU implies per-shot and per-task rates. The cost per shot depends on the QPU selected, and the use of error mitigation may require a minimum of 2500 shots per task.

Finally, the connection to IBM is done in the same way as with AWS, by implementing their connection using their Qiskit SDK. With this, developers can choose to run their circuits on local simulators or real QPUs, and the system will automatically manage the credentials and tokens needed to authenticate and perform the executions. Finally, the cost is calculated based on the computational resources used and the hardware access time, that is, the actual QPU usage time, measured in seconds of "execution time". For both cases, the estimated cost to execute the circuit is presented. This allows users to make informed decisions on the use of resources and cost management.

4 CONCLUSION

The QCRAFT Quantum Developer Interface represents a significant advancement in CD practices for quantum computing, providing a comprehensive solution for quantum circuit deployment. By integrating tools, such us Quirk and Docker, this tool addresses the complexities of designing, storing, and executing quantum circuits across different hardware platforms (AWS and IBM) making quantum technology more accessible and efficient for developers.

One of the most outstanding conclusions is the demonstration that the integration of visual tools, such as Quirk, with quantum execution platforms can significantly simplify the development process in computing. It has been observed that the ability to design quantum circuits in an intuitive and visual way, and then execute them in real hardware in the form of services facilitates a better understanding and at the same time a great organization since both circuits and results can be stored.

Furthermore, the research questions posed at the beginning have been successfully addressed. QCRAFT demonstrates how a unified framework can streamline the development and deployment of quantum circuits across diverse platforms, reduces fragmentation through cross-platform compatibility and environment consistency, and provides a modular and scalable solution to improve accessibility in quantum software workflows. Moreover, thanks to its modular structure, it is easy to integrate with other quantum platforms, highlighting its interoperability.

Future improvements to this tool could include optimizing execution times and defining credential management protocols, which will further improve deployment efficiency. Enhanced integration of DevOps with tools like Jenkins could also be explored to provide more extensive automation options.

A significant future enhancement would be the facilitation of collaboration between different users. Implementing a system that allows multiple users to work together on the design of quantum circuits in real time would be highly beneficial. This could include features such as shared circuit editing, integrated communication, and collaborative project management, which promote a more dynamic working environment.

ACKNOWLEDGEMENTS

This work has been partially funded by the European Union "Next GenerationEU /PRTR", by the Ministry of Science, Innovation and Universities (TED2021-130913B-I00, and PDC2022-133465-I00). By QSERV project (PID2021-1240454OB-C31) funded by the Spanish Ministry of Science and Innovation and ERDF; by the Regional Ministry of Economy,

Science and Digital Agenda of the Regional Government of Extremadura (GR21133); and by European Union under the Agreement - 101083667 of the Project "TECH4E -Tech4effiency EDIH" regarding the Call: DIGITAL-2021-EDIH-01. Also, supported by grant PRE2022-102070 financed by MCIN/AEI/10.13039/501100011033, "FEDER/EU", and FSE+.

REFERENCES

Abraham, H., Akhalwaya, I. Y., Aleksandrowicz, G., et al. (2019). Qiskit: An Open-source Framework for Quantum Computing. Available: https://doi.org/10.5281/zenodo.2562110.

AbuGhanem, M. (2025). Ibm quantum computers: Evolution, performance, and future directions. *The Journal of Supercomputing*, 81(5):687.

Alvarado-Valiente, J., Romero-Álvarez, J., Moguel, E., and García-Alonso, J. (2023a). Quantum web services orchestration and management using devops techniques. In *International Conference on Web Engineering*, pages 389–394. Springer.

Alvarado-Valiente, J., Romero-Álvarez, J., Moguel, E., García-Alonso, J., and Murillo, J. M. (2023b). Technological diversity of quantum computing providers: a comparative study and a proposal for api gateway integration. *Software Quality Journal*, pages 1–21.

Alvarado-Valiente, J., Romero-Álvarez, J., Moguel, E., García-Alonso, J., and Murillo, J. M. (2024). Orchestration for quantum services: The power of load balancing across multiple service providers. *Science of Computer Programming*, 237:103139.

Arute, F., Arya, K., Babbush, R., et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.

Chicano, F., Luque, G., Dahi, Z. A., and Gil-Merino, R. (2025). Combinatorial optimization with quantum computers. *Engineering Optimization*, pages 1–26.

Díaz, A., Alvarado-Valiente, J., Romero-Álvarez, J., Moguel, E., Garcia-Alonso, J., Rodríguez, M., García-Rodríguez, I., and Murillo, J. M. (2025). Service engineering for quantum computing: Ensuring high-quality quantum services. *Information and Software Technology*, 179:107643.

Gheorghe-Pop, I.-D., Tcholtchev, N., Ritter, T., and Hauswirth, M. (2020). Quantum devops: Towards reliable and applicable nisq quantum computing. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE.

Hevia, J. L., Peterssen, G., and Piattini, M. (2022). Quantumpath: A quantum software development platform. *Software: Practice and Experience*, 52(6):1517–1530.

Hirata, Y., Nakanishi, M., Yamashita, S., and Nakashima, Y. (2011). An efficient conversion of quantum circuits to a linear nearest neighbor architecture. *Quantum Information & Computation*, 11(1):142–166.

Khan, A. A., Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., and Mikkonen, T. (2023). Software architecture for quantum computing systems—a systematic review. *Journal of Systems and Software*, 201:111682.

Kourtis, M. A., Tcholtchev, N., Gheorghe-Pop, I.-D., Becker, C. K.-U., Xylouris, G., Markakis, E., Petric, M., Seidel, R., and Bock, S. (2024). Towards continuous development for quantum programming in decentralized iot environments. *Procedia Computer Science*, 238:7–14.

Minerbi, N. (2022). Quantum software development with classiq. In *Quantum Software Engineering*, pages 269–280. Springer.

Moguel, E., Rojo, J., Valencia, D., Berrocal, J., Garcia-Alonso, J., and Murillo, J. M. (2022). Quantum service-oriented computing: current landscape and challenges. *Software Quality Journal*, 30(4):983–1002.

Murillo, J. M., Garcia-Alonso, J., Moguel, E., Barzen, J., Leymann, F., Ali, S., Yue, T., Arcaini, P., Pérez-Castillo, R., García Rodríguez de Guzmán, I., et al. (2025). Quantum software engineering: Roadmap and challenges ahead. *ACM Transactions on Software Engineering and Methodology*.

Nguyen, H. T., Usman, M., and Buyya, R. (2024). Qfaas: A serverless function-as-a-service framework for quantum computing. *Future Generation Computer Systems*, 154:281–300.

Romero-Álvarez, J., Alvarado-Valiente, J., Moguel, E., Garcia-Alonso, J., and Murillo, J. M. (2023). Enabling continuous deployment techniques for quantum services. *Software: Practice and Experience*.

Romero-Alvarez, J., Alvarado-Valiente, J., Moguel, E., Garcia-Alonso, J., and Murillo, J. M. (2023). A workflow for the continuous deployment of quantum services. In *2023 IEEE International Conference on Software Services Engineering (SSE)*, pages 1–8. IEEE.

Serrano, M. A., Cruz-Lemus, J. A., Perez-Castillo, R., and Piattini, M. (2022). Quantum software components and platforms: Overview and quality assessment. *ACM Computing Surveys*, 55(8):1–31.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE.

Stirbu, V., Kinanen, O., Haghparast, M., and Mikkonen, T. (2024). Qubernetes: Towards a unified cloud-native execution platform for hybrid classic-quantum computing. *Information and Software Technology*, 175:107529.

Zhao, J. (2020). Quantum software engineering: Landscapes and horizons. *arXiv preprint arXiv:2007.07047*.