CIRI2 Tech-stack Document

Jordy Walraven

Version history:

Version	Author(s)	Description	Date
1	Jordy Walraven	Setup Initial Tech stack	27-3-2024

Distribution history:

Version	Distributed to	Comments	Date

Table of contents

Context

Ciri2 is a platform designed for PC gaming enthusiasts. At its core, Ciri2 serves as a hub where users can assess their system's capability to run specific games and benchmark their performance against others in the community. Here's an overview of its key functionalities:

Account Creation:

Users can easily create personalized accounts, unlocking access to a host of features tailored to their gaming needs.

PC Rig Configuration:

Within the application, users have the flexibility to construct and customize their PC rigs using intuitive part pickers for various components. This empowers them to tailor their systems to meet the demands of their favorite games.

Game Overview:

Through Ciri2, users gain access to a comprehensive overview of games, including key details such as system requirements, gameplay mechanics, and community-generated performance data. This empowers gamers to make informed decisions about which titles to explore based on their hardware capabilities.

FPS Submission:

Gamers can submit their frames per second (FPS) benchmarks for specific games and graphics presets, allowing them to share and compare their performance metrics with fellow users. This feature fosters a sense of community and healthy competition among players.

Non-functional requirement

My choices for tech stack are based on my non functional requirements.

Architecture

To see which architecture choices I made look at the research document, you can find the latest info about the architecture in the architecture document.

Frontend Framework

For my frontend application I will be making a single page application. Here I looked at multiple frameworks to create my application. Some of the most professional and most used frameworks are Angular, Reactjs, Vue and Svelte, although svelte is relatively new.

I will make my decision of which framework to use based on my requirements:

Requirement	Explanation	Weight (0-1)
Documentation quality	Does the framework have documentation that is understandable and can be used to fix issues.	0.8
Performance	How fast does the framework load, and how quick does it process events	0.8
Development speed	How fast is it to add a new feature to the application	1
Active community	How active is the community, and can it be used to fix issues	0.8
Ease of use	How easy is the framework to learn and add new features	0.8
Maintainability	How well can you maintain the framework	1
Maturity	How much has the framework matured	0.8

Angular

Angular is a very matures framework with a lot of users, a couple of years back angular was known to be quite have and pretty slow, this has luckily been fixed, and angular now is a pretty quick framework. Because Angular preferences a default project structure it means there is a lot of documentation available on how to structure you project effectively. This also means that the angular structure of different project is very much like each other. Angular has also improved its documentation with a new website with a lot of information and handy docs.

Requirement	Explanation	Score
Documentation quality	Angulars documentation has been renewed and looks and works very good	9
Performance	Angular is equally as fast as it's competitors	7
Development speed	The new angular uses standalone components making development speed way quicker, and because of the default structure and lots of boilerplate code the development speed is high	8
Active community	Angular has an active community with a lot of users	8
Ease of use	Angular might be quite daunting for new users, as there is a lot of functionality and is more complex than its competitors	7
Maintainability	Because of the default project structure angular is very maintanable	10
Maturity	Angular has proven it's success and is very matured	10

React

React is a highly matured library with a vast user base. In the past, React was criticized for its performance, but significant improvements have been made, rendering it now a swift framework. React's strength lies in its flexibility, allowing developers to structure projects according to their preferences. However, this flexibility can sometimes lead to inconsistent project structures across different teams. Despite this, React boasts extensive documentation, which has been continuously refined to provide comprehensive guidance. Its modular approach, particularly with the use of standalone components, enhances development speed. React's active community ensures ample support and resources for users. Nonetheless, newcomers may find React initially challenging due to its complex nature compared to other frameworks. Nevertheless, once mastered, React's maintainability is high, thanks to its clear project structure and modular design principles.

Requirement	Explanation	Score
Documentation quality	React's documentation is extensive and regularly updated, providing thorough guidance for developers	9
Performance	React's performance has significantly improved over time, making it on par with its competitors.	7
Development speed	Development speed React's modular design and component-based architecture contribute to faster development speeds. However, project structures may vary, impacting consistency across different teams.	8
Active community	React boasts a large and active community, ensuring ample support and resources for developers.	9
Ease of use	React's flexibility allows for powerful customization, it may present a steeper learning curve for newcomers compared to other frameworks.	8
Maintainability	Because of the default project structure angular is very maintanable	7
Maturity	Reactjes is a matured framework and has been around for long	10

Vue.js

Vue.js is a well-established framework with a growing community of users. Previously, Vue.js was lauded for its simplicity and ease of use, but it was perceived to lack the maturity and scalability of its counterparts. However, significant advancements have been made, addressing these concerns and positioning Vue.js as a competitive option in the front-end development landscape. Vue.js emphasizes a progressive approach to building user interfaces, allowing developers to integrate it seamlessly into existing projects. This flexibility, combined with its clear and concise documentation, facilitates a smooth learning curve for developers of all skill levels. Moreover, Vue.js prioritizes performance, with optimizations continually implemented to enhance speed and efficiency. The framework's active community ensures ongoing support and the availability of valuable resources for users. Vue.js encourages a modular and component-based development approach, promoting code reusability and maintainability. Overall, Vue.js offers a balanced combination of simplicity, performance, and scalability, making it an attractive choice for front-end development projects.

Requirement	Explanation	Score
Documentation quality	Vue.js boasts clear and concise documentation, making it easy for developers to learn and use the framework effectively.	9
Performance	Vue.js prioritizes performance, with optimizations aimed at enhancing speed and efficiency.	8
Development speed	Vue.js facilitates rapid development through its progressive approach and support for modular, component-based architecture.	8
Active community	Vue.js has a growing and active community, providing ongoing support and valuable resources for users.	8
Ease of use	Vue.js is renowned for its simplicity and ease of use, making it accessible to developers of all skill levels	8
Maintainability	Vue.js promotes code reusability and maintainability through its modular and component-based development approach.	8
Maturity	Vue is a matured framework	8

Svelte

Svelte, a relatively newer entrant into the front-end development landscape, has been gaining momentum rapidly with its innovative approach to building user interfaces. Initially praised for its simplicity and unique compiler-based architecture, Svelte has evolved to address concerns about maturity and scalability, establishing itself as a compelling choice for developers.

Requirement	Explanation	Score
Documentation quality	Svelte offers comprehensive documentation that guides developers through its unique concepts and workflow. But it doesn't contain that much information	8
Performance	Svelte prioritizes performance by shifting much of the heavy lifting to compile time, resulting in highly optimized and efficient output code.	9
Development speed	Its component-based architecture fosters modularity and reusability, further streamlining development workflows.	8
Active community	Svelte boasts a vibrant and rapidly growing community of developers, educators, and enthusiasts.	8
Ease of use	Svelte is renowned for its simplicity and developer-friendly approach. With its minimalistic syntax and intuitive concepts	8
Maintainability	Svelte promotes maintainability through its reactive and component-based architecture.	8
Maturity	Svelte is less matured than other frameworks	7

Totals

Framework	Documentation quality (0.8)	Performance (0.8)	Development speed (1)	Active community (0.8)	Ease of use (0.8)	Maintainability (1)	Maturity (0.8)
Angular	7.2	5.6	8	6.4	5.6	10	8
React	7.2	5.6	8	7.2	6.4	7	8
Vuejs	7.2	6.4	8	6.4	7.2	8	6.4
Svelte	6.4	7.2	8	6.4	6.4	8	7

Angular	React	Vuejs	Svelte
50.8	49.4	49.6	49.4

We can see that angular barely scores higher than the other frameworks , and this is mainly because of the maturity of the framework. We can see that all scores are very close to each other, and this is because all the frameworks are quite like each other.

Databases

We will make a decision on what database to used based on my non-functional requirements and the data I will be storing, the first question that pops up is:

Relational or non-relational database?

Before we can make a decision between a non-relational database and relational database we need to take a look at the advantages of both types:

Relational database

Relational databases offer a structured approach to data storage, with tables, rows, and columns facilitating efficient querying and data retrieval. Their predefined schemas ensure data integrity and consistency, making them ideal for applications requiring complex transactions and strict adherence to ACID (Atomicity, Consistency, Isolation, Durability) properties. Additionally, relational databases support powerful SQL querying, enabling seamless integration with existing systems and robust reporting capabilities. (Relational vs. Non-Relational Databases, 2022)

Non-relations database

Non-relational databases excel in handling unstructured or semi-structured data, offering flexibility and scalability to manage vast volumes of information across distributed environments. With schema-less designs and various models like document, key-value, column-family, and graph databases, non-relational databases can adapt easily to evolving data needs, making them suitable for agile development, rapid prototyping, and applications requiring high availability and horizontal scalability. (Relational vs. Non-Relational Databases, 2022)

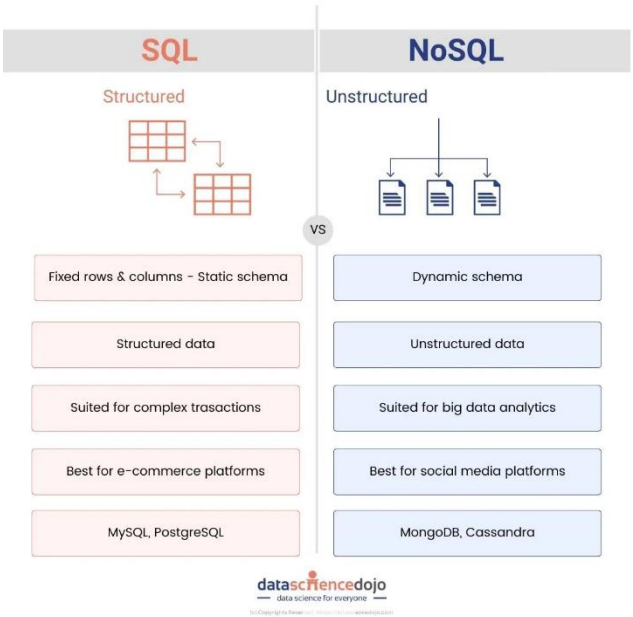

Figure 1 https://datasciencedojo.com/

Comparison

In conclusion, the decision between a relational and non-relational database hinges on the specific requirements of Ciri2. Since some of the models prioritize flexibility over rigid relationships and consistency, opting for a non-relational database is the most fitting choice. Its ability to handle unstructured or semi-structured data, coupled with its scalability and adaptability, aligns well with the needs for agile development and accommodating varying data structures. Therefore, embracing a non-relational database would best serve the requirements by providing the necessary flexibility and scalability to effectively manage your data. Horizontal databases can also scale horizontally while this is not possible in relational databases. This is also a big positive if we look at the predicted amount of traffic for the system.