International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

A Software Quality-Based Comparative Guide for Integrating React, Angular, and Vue.js with Django

Axel Omar Salazar Guarneros[1], Juan Ramos Ramos[2], José Juan Hernández Mora[3], María Guadalupe Medina Barrera[4], Yesenia Nohemí González Meneses[5]

[1]Tecnológico Nacional de México: Campus Apizaco
Email: *m23370049[at]apizaco.tecnm.mx*

[2]Tecnológico Nacional de México: Campus Apizaco
Corresponding Author Email: *juan.rr[at]apizaco.tecnm.mx*

[3]Tecnológico Nacional de México: Campus Apizaco
Email: *juan.hm[at]apizaco.tecnm.mx*

[4]Tecnológico Nacional de México: Campus Apizaco
Email: *guadalupe.mb[at]apizaco.tecnm.mx*

[5]Tecnológico Nacional de México: Campus Apizaco
Email: *yesenia.gm[at]apizaco.tecnm.mx*

Abstract: ***This article presents a comparative analysis of the front-end frameworks React, Angular, and Vue.js, evaluating usability and maintainability based on ISO/IEC 25010:2011 and ISO 9241-11:2018 standards. A standardized backend using Django REST Framework was developed alongside three functionally identical front-end clients, each implemented in a different framework. The study utilized a structured technological research methodology to assess developer experience and architectural maintainability. Results indicate Vue.js offers concise code and a gentle learning curve, Angular ensures structural consistency suitable for large-scale applications and React delivers high flexibility and developer satisfaction. A practical guide is proposed to support evidence-based framework selection, contributing to improved software project outcomes.***

Keywords: React, Angular, Vue.js, Django REST Framework, Software Quality

1. Introduction

In the current context, where software development is key to innovation, the high failure rate of projects—with figures such as 70% reported by BCG and 66% according to the CHAOS Report 2020[1]—reveals critical shortcomings in the initial phase, particularly in technology selection. This work addresses that issue by focusing on the appropriate choice of front-end tools, a key determinant of the final product's usability and maintainability. Through a comparative analysis of the most relevant frameworks and the creation of a taxonomy of needs, it aims to develop a practical and comprehensive guide to facilitate their efficient implementation. The research questions guiding this study focus on identifying the most suitable framework for each system and defining the essential elements that such a guide should include to optimize front-end development, reduce risks, and ensure successful, high-quality software products.

2. Methodology

For this technological development research, the technological research methodology proposed by Celso de la Cruz Casaño (2016) [3] was chosen, as it is specifically oriented toward projects that not only aim to generate theoretical knowledge but also to propose, design, develop, implement, and evaluate applied technological solutions.
This methodology is particularly useful when the objective is not merely to describe or explain a phenomenon, as in traditional scientific research, but rather to transform a situation through the design and development of an innovative solution or technological improvement.

2.1 Technological research methodology

Figure 1: Methodology proposed by Celso de la Cruz Casaño [3]

2.2 Proposed Development Methodology

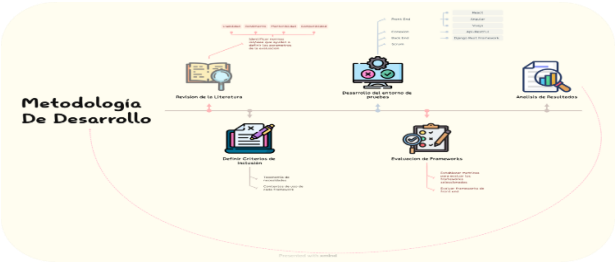

Figure 2: Proposed development methodology

3. Analyzing usage contexts

Figure 3: Context of use ReactJs, Angular, VueJs

The graph indicates React's dominance in terms of global search interest over the past five years. With more than half of the "pie," React stands as the undisputed leader, surpassing its closest competitors, Angular and Vue.js, by a significant margin [4]-[5].

React (Light Blue – 57%)
- Holds the largest portion of the chart, representing 57% of total interest.
- This figure suggests that React is not only the most popular framework but also generates more searches than Angular and Vue.js combined (28.1% + 14% = 42.1%).
- This may translate into a larger community, more available learning resources (tutorials, articles, courses), and likely greater demand in the job market.

Angular (Teal – 28.1%)
- Ranks as the second most popular framework, with 28.1% of interest.
- Although this is a considerable share and highlights its relevance—especially in corporate environments and large-scale projects—it still lags significantly behind React.

Vue.js (Orange – 14%)
- Takes third place with 14%.
- Despite being third, Vue.js maintains a solid user base and is known for its friendly learning curve and flexibility. Its 14% represents an active community and a growing ecosystem.

4. Test environment setup

To conduct an objective evaluation of the React, Angular, and Vue.js frameworks, a controlled testing environment was designed with the primary goal of isolating variables. The strategy consisted of developing a single, consistent backend so that the front-end framework would be the only variable element. This approach allows any differences in usability and maintainability metrics to be attributed directly to the characteristics of each technology.

The process involved two main phases: the creation of the server and the implementation of the clients.

4.1 Phase 1: Creation of a Standardized Backend with Django REST Framework

A robust and uniform RESTful API was built for a "Task Manager" application, serving as the foundation for the three clients. The main steps were as follows:
- Environment and Structure: A Python virtual environment was set up to install key dependencies: Django, Django REST Framework (DRF), and a library for managing CORS. The project was organized modularly to facilitate maintenance.
- Model and Serialization: A data model named Task was defined, including fields for title, description, and status (completed or not). To send and receive this data over the web in JSON format, a ModelSerializer was implemented, acting as a translator between database objects and the API format.
- Logic and Endpoints: Instead of manually writing logic for each operation (create, read, update, delete), a ModelViewSet from DRF was used. This high-level tool automatically generates all necessary endpoints. Finally, a DefaultRouter was configured to expose the API under a standardized URL (/api/v1/tasks/), ensuring that all three clients connected to the same endpoint. The API was verified using Postman before proceeding.

4.2 Phase 2: Implementation of Three Identical Front-end Clients

Three front-end applications with identical functionality were developed, each using a different framework. All applications fully implemented the functionality of a "Task Manager" by consuming the DRF API.
- **React Client:** The setup was done using Vite.js. It used functional components and Hooks (useState, useEffect) to manage state and logic. Communication with the API was handled through the axios library, and navigation between views was managed with react-router-dom. The project was organized into folders for components, pages, and API services, promoting clear and reusable code.
- **Angular Client:** The project was generated using Angular CLI. Following its more structured architecture, a specific module (TasksModule) was created, along with components for the task list and form, and an injectable service (TasksService) that centralized HTTP calls using Angular's HttpClient. Reactive forms were used for data handling, and a TypeScript interface ensured consistency in the Task data model.
- **Vue.js Client:** Vite.js and the official Vue template were used. The Composition API and Single-File Components (SFCs) structure were adopted, encapsulating template, logic, and styles within a single file. Reactive state was managed with ref, API requests were centralized in a service using axios, and routing was handled with vue-router.

Upon completion, three fully operational and functionally equivalent client applications were obtained, ready for comparative analysis in terms of usability and maintainability.

5. Results and discussion

5.1 Usability Analysis

Usability was evaluated from the developer's perspective (Developer Experience), focusing on efficiency, learning curve, and satisfaction.

- **Code Efficiency:** Differences were observed in the amount of code required. Vue.js proved to be the most concise, thanks to its Single-File Components (SFCs). React ranked in the middle, requiring explicit state management. Angular was the most verbose due to its module- and service-based architecture, which generates more configuration code. This finding is consistent with Moraguez (2025), who highlights Vue's simple syntax [8].
- **Learning Curve:** Vue.js presented the smoothest learning curve, making it ideal for becoming productive quickly. React held an intermediate position, with an easy-to-learn core but a complex ecosystem. Angular showed the steepest curve due to being a full-featured framework and its use of advanced concepts like TypeScript and RxJS [5]-[7].
- **Performance and Satisfaction:** Although not the main focus, the observed performance aligns with industry benchmarks (Interactive Results, n.d.), where Vue and React excel in DOM manipulation. In terms of satisfaction, data from the State of JavaScript (2022) survey are clear: React leads (82.95%), followed closely by Vue (77.32%), while Angular (42.62%) shows lower preference—likely due to its complexity [7].

5.2 Maintainability Analysis

Maintainability was analyzed based on architecture, ease of modification, and scalability.

- **Architectural Structure:** Each framework enforces a distinct philosophy, as summarized by Tupeli Pauli (2020) [10]. Angular enforces a rigid structure that ensures consistency, ideal for large teams. React, being a library, offers total flexibility, but consistency depends on team discipline. Vue.js strikes a balance by providing a clear yet less rigid structure compared to Angular.
- **Ease of Change and Testing:** Angular stands out thanks to its mandatory use of TypeScript and dependency injection system, which make refactoring and testing easier. While React and Vue support TypeScript, its use remains optional.
- Vue's Composition API has significantly improved code organization. As Pauli (2020) notes, Angular's testing tools are more integrated, whereas React and Vue rely on externally configured ecosystems [10].

5.3 Comparative Results Matrix

To consolidate the analysis, the following table summarizes the evidence obtained from both the practical case study and the external sources consulted. Each framework is rated on a scale of (+++) High, (++) Medium, and (+) Low for each evaluated metric.

Table 1: Comparative Results Matrix

Evaluation Criteria	React.js	Angular	Vue.js
Code Efficiency	++	+	+++
Learning Curve	++	+	+++
Performance (DOM)	+++	+	+++
Developer Satisfaction	+++	+	++
Structure and consistency	++	+++	++
Ease of Maintenance / Refactoring	++	+++	++
Architectural flexibility	+++	+	++
Community & Ecosystem	+++	++	++

5.4 Practical Guide: A Methodological Framework for Decision-Making

The previous analysis and comparative matrix demonstrate that there is no universally superior framework; the right choice depends on context. To translate these findings into a practical and operational tool, a methodological guide has been structured to help teams navigate this decision-making process. This guide provides a roadmap to systematize evaluation and facilitate the adoption of the most suitable technology. The proposed structure is organized into the following logical phases:

5.4.1 Planning and Definition Phase

- **Project Definition:** Begin by establishing the project foundation—its purpose, scope, objectives, and requirements (functional, non-functional, hardware). This ensures that the framework selection aligns with the business's real needs.
- **Selection of Standards and Metrics:** Define evaluation criteria (Evaluation Parameters and Metrics) based on international standards such as ISO 9241-11 (Usability) and ISO/IEC 25010 (Maintainability). This phase guarantees an objective and rigorous comparison [11]-[12].

5.4.2 Practical Implementation Phase

- **Backend Configuration:** Set up a common foundation (in this case, Django REST Framework) to ensure that the front-end comparison is performed under equal conditions.
- **Implementation per Framework (Angular, React, Vue):** Detail a standardized process for each technology, covering initial setup, component creation, services, routing, and styling. This practical section is crucial for developers to directly experience each framework's Developer Experience.

5.4.3 Evaluation and Conclusion Phase

- **Evaluation with Defined Metrics:** Apply the criteria established in the first phase to analyze the results from the practical implementation.
- **Results Analysis and Conclusions:** Summarize findings in a comparative matrix, highlighting the strengths, weaknesses, and specific usage recommendations for each framework.

This structured methodology enables teams to make informed, evidence-based decisions, reducing subjective bias and aligning technology selection with project goals and organizational strategy.

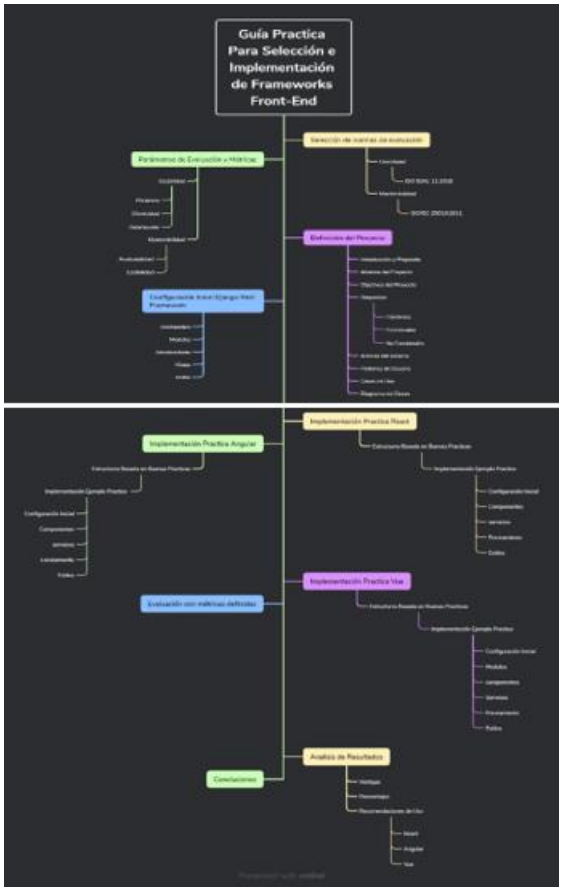

Figure 4: Structure Practical Guide

6. Conclusions

The research conducted demonstrates conclusively that there is no universally superior front-end framework; the optimal choice is inherently dependent on the project's context, business objectives, and the development team's capabilities. In response to the first research question regarding which framework is most suitable, this study concludes that suitability is defined by a balance between maintainability and usability according to specific priorities:

1) React.js stands out as the preferred option for projects that require high flexibility, a robust ecosystem, and a dynamic development experience. Its component-based architecture and high satisfaction rate make it ideal for interactive, scalable applications where the freedom to choose tools is an advantage.
2) Angular positions itself as the most robust solution for large-scale enterprise applications, where structural consistency, long-term maintainability, and code predictability are critical. Its opinionated nature and integration with TypeScript, though imposing a steeper learning curve, ensure quality and stability in large teams.
3) Vue.js emerges as the optimal solution for projects prioritizing development speed, a low entry barrier, and concise code. It is the most accessible tool for startups, SMEs, and teams needing to deliver Minimum Viable Products (MVPs) quickly without compromising competitive performance.

Regarding the second research question—about the content of a comprehensive practical guide—it was determined that its effectiveness lies in combining three key elements: (1) a comparative analysis based on international standards (ISO/IEC), (2) a practical implementation of a standardized use case ("Task Manager"), and (3) a set of contextualized recommendations that map findings to real-world project scenarios. The guide developed in this study facilitates technological decision-making, reduces uncertainty, and improves development efficiency.

This study directly addresses the initial problem of the high software project failure rate by replacing technology selection based on popularity or subjective preference with a methodological, evidence-based process. The main contribution of this work is, therefore, a structured decision-making framework that provides software architects and technical leaders with the tools to align technological choices with the product's strategic objectives, thereby improving the likelihood of success.

Finally, it is important to acknowledge the study's limitations. The evaluation focused on a low-complexity application. Projects with higher performance demands or more complex architectures could yield different nuances. Future research could expand this analysis to include emerging frameworks (such as Svelte or Qwik), assess the impact on larger-scale applications (e.g., e-commerce platforms), and conduct usability testing with end users to complement the developer's perspective.

References

[1] Standish Group. (2021, enero). *CHAOS 2020: Beyond Infinity Overview*. Standish Group International.
[2] Forth, P., Reichert, T., de Laubier, R., & Chakraborty, S. (2020, octubre 29). *Flipping the odds of digital transformation success*. Boston Consulting Group. https://www.bcg.com/publications/2020/increasing-odds-of-success-in-digital-transformation
[3] De la Cruz Casaño, C. (2016). *Metodología de la investigación tecnológica en ingeniería*. Universidad Continental. https://www.academia.edu/94930372/Metodología_de_la_investigación_tecnológica_en_ingeniería
[4] Google Trends. (n.d.). *React, Angular, Vue.js*. Recuperado el 25 de agosto de 2025, de https://trends.google.es/trends/explore?date=today%205-y&q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d&hl=es
[5] Tkrotoff. (2025). *Front-end frameworks popularity (React, Vue, Angular and Svelte)*. GitHub Gist. https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190
[6] Interactive results. (s. f.). *JS Framework Benchmark*. https://krausest.github.io/js-framework-benchmark/current.html
[7] State of JavaScript. (2022). *Front-end frameworks*. https://2022.stateofjs.com/en-US/libraries/front-end-frameworks
[8] Moraguez, E. R. (2025). *Comparativa de Frameworks Front-end: React vs. Vue vs. Angular*. LovTechnology. https://lovtechnology.com/comparativa-de-frameworks-front-end-react-vs-vue-vs-angular/

[9] Pano, A., Graziotin, D., & Abrahamsson, P. (2018). Factors and actors leading to the adoption of a JavaScript framework. *Empirical Software Engineering, 23*(6), 3503–3534. https://doi.org/10.1007/s10664-018-9613-x

[10] Tupeli, P. (2020, octubre). *Ensuring maintainability of JavaScript web applications* [PDF]. https://tupelipauli.pdf

[11] International Organization for Standardization. (2018). *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts* (ISO 9241-11:2018).

[12] International Organization for Standardization & International Electrotechnical Commission. (2011). *Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — System and software quality models* (ISO/IEC 25010:2011).

[13] Sanjay, K. C. (2024). *Comparative study of front-end frameworks: React and Angular* [Tesis de licenciatura, Theseus]. Theseus. https://urn.fi/URN:NBN:fi:amk-2024080724089

[14] Angular. (n.d.). *Application framework for creating single-page applications*. https://angular.io/

[15] React. (n.d.). *A JavaScript library for building user interfaces*. https://reactjs.org/

[16] Vue.js. (n.d.). *The progressive JavaScript framework*. https://vuejs.org/