Cloud-based Endpoint Management Solutions: A Software Engineering Internship Reflection

CS4991 Capstone Report, 2023

Yesenia Andrade
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
yya2jq@virginia.edu

ABSTRACT

Two challenges faced administrators accessing Microsoft's managed devices: 1) the code to display the site needed to be updated; and 2) some of the information displayed was retrieved from different data sources that needed to be consolidated using Microsoft's other product called Kusto. To solve these issues, I migrated the existing Managed App tab to use a new web framework called React. As a result of my internship, the product grabs data much more quickly and is more user-friendly. User-friendliness was assessed using Microsoft's internal tool that gives a user interface (UI) score and was verified by a product designer on the team. This migration made grabbing data 3-4 times faster to render and load the data on a web page, based on query metrics and Kusto time. As for future work, several new features to be explored include design refinements, ongoing user feedback, and more optimization as data is consolidated. These efforts are essential as to ensure the product remains competitive and adaptable and meets user needs.

1. INTRODUCTION

Microsoft is a big tech company that owns a cloud-based endpoint management solution that helps manage user access and simplifies app and device management across company's many devices, including mobile, desktop computers, and virtual endpoints (Microsoft, 2023). My project consisted of updating one of the many features of this product. Before my work, this product had a managed app tab for companies to view the apps that the user's devices contain. This tab was written in a language that was no longer used across multiple companies due to its loss of popularity over the years. The user's app data would take a while for the website to render because it grabbed from multiple data sources.

The team had been working on migrating other tabs into a more widely used language React. UI's features seemed to give a cleaner look, making all these tabs look revamped. Because of the product being written in multiple languages, my job was to continue the integration to React to move the team one step closer to a full migration.

2. RELATED WORKS

Handshake, a platform used for career services and university recruitment, has also moved to migrating their front end from server-rendered views with jQuery and Knockout to TypeScript and React (Botkin, 2021). The motivation for migrating to a modern user interface (UI) library, React, stemmed from the increasing complexity of Handshake's UX requirements, which made maintaining and extending the current UI challenging. There were also issues resulting from mixing Knockout and jQuery, which was addressed with the more modern framework, React. The goal of this transition was to a component-based UI with enhanced features, resolve deficiencies in Knockout's component system, and improve the front-end testing. Additionally, the evolving landscape of front-end technologies and limitations of their current existing stack prompted Handshake to explore new options to stay updated with industry advancements.

The migration for Handshake utilized the same steps I used in my project. The migration process to React was approached incrementally, beginning with setting up the necessary tooling to integrate React components alongside existing previously used Knockout code, such as packages. This initial phase included introducing a basic component to familiarize Handshake's team with the new technology. As with my project, their work could only be seen internally before full migration.

The next step was to create an actual feature in React, such as a table component. During my project a separate team oversaw the work to make sure all business needs were being met and so the minimum viable product wasn't getting lost in the migration.

In the third phase, we worked on making sure that all the user data previously managed by Knockout could now be managed by React and the new Java framework.

The fourth phase focused on ensuring the team could use the new features. The process ended with the team learning more about React, allowing for an incremental migration of existing functionality while mitigating risks by rolling out updates alongside the existing UI and gathering feedback for iterative improvements.

React is maintained by Meta (formerly known as Facebook). Products such as Messenger and Facebook Marketplace use React today (Vanderhoof, 2023). However, these products did not always use React. Before 2014, Facebook was using XHP, a server-side framework, to create their UI features. A lot of well-known companies are migrating to use React now because of all its advantages.

3. PROJECT DESIGN

In this section, we will delve into the project design, which encompasses two distinct phases, each presenting its own set of challenges and unique requirements.

3.1 UX/UI Methods

Part of the migration consisted of revamping two UI features. The first was the drop-down box that displayed the list of users logged into a particular device. The updated drop-down box received improvements on its appearance and functionality.

The second was the table that displayed all the managed apps under a certain user's device. My team had created a new UI component that this feature heavily depended on. This component assisted in the usability and the aesthetic of the product. The table looked more like the other tabs tables and fit the look of all other sites across the team.

3.2 Database Migration

Apart from UI, we need Kusto to consolidate data to load the table and to select from users. Prior to this integration, data fetching involved retrieving information from two other data sources and databases, which not only made the process less efficient but also caused challenges for migration (Figure 1). The transition to Kusto brought several significant advantages.

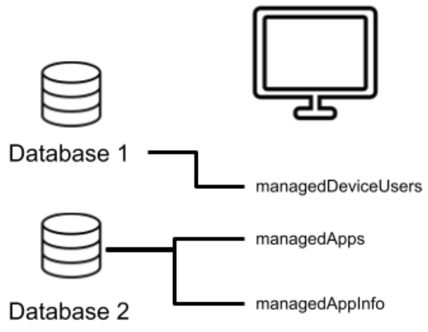

Figure 1: Flow of Data Before Migration

Consolidating data within Kusto simplified the data architecture. Instead of fetching data from other databases, all relevant information could now be stored and managed within Kusto. This consolidation eliminated the need to navigate between two entirely different databases, which reduced the complexities associated with data migration (Figure 2).

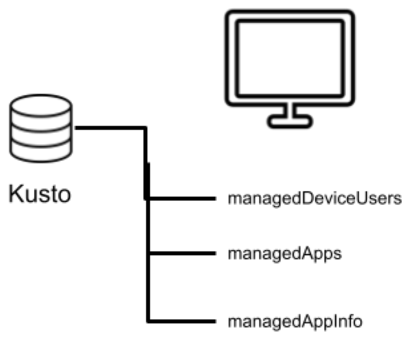

Figure 2: Flow of Data After Migration

Furthermore, the integration of Kusto made data fetching significantly quicker. By centralizing the data in a single repository, queries and retrievals became more efficient, as Kusto's architecture is designed for high performance analysis. This resulted in faster access to the required data, enhancing the overall website's responsiveness and user experience. In summary, the adoption of Kusto not only simplified the data management process but also made data fetching quicker and more efficient.

3.3. **Challenges**

Some challenges that became apparent while working on my project included a learning curve, missing information, and data interpretation. The learning curve associated with adopting Kusto and transitioning from the previous method of fetching data from multiple sources to a centralized database was difficult as Kusto's architecture and query language required that I take some time to learn how it worked. A lot of the existing Kusto clusters of data had missing attributes that were necessary for an operating UI. Overcoming this hurdle meant putting efforts to identify, retrieve, or fill in the gaps in the data, requiring close collaboration with my mentor and team. These challenges emphasized the significance of planning, ongoing learning, and collaborative data management practices.

4. **RESULTS**

As a result of implementing my project, the team was able to get one step closer to migrating the product's entire UI to use React. Data retrieval was also sped up by 0.91 seconds. After running the UI through Microsoft's internal UI tester, the new React code received a 99.8% Lighthouse score, which means that it is in great condition to ship to production. The new UI made it easier for administrators using this tab to clearly find and identify all features of the product which included sorting, filtering, and selecting.

5. CONCLUSION

My project addressed challenges faced by administrators accessing Microsoft's managed devices. I successfully migrated the existing Managed App tab to utilize the modern web framework, React, which not only improved the appearance and usability of the user interface but also significantly accelerated data retrieval times, enhancing the overall user experience. Through UI testing, we achieved a 99.8% Lighthouse score, indicating that the UI is in excellent condition for production use.

This project marked a significant milestone, bringing my team closer to a comprehensive migration to React and streamlining data management through Kusto. The results included a faster data retrieval and a more user-friendly interface which contributes to the ongoing success of the product.

6. FUTURE WORK

Looking ahead, I will continue refining the design, gathering user feedback, and optimizing data consolidation to ensure our product remains competitive, adaptable, and aligned with the evolving needs of our users. Despite the learning curve and challenges faced, this project represents a valuable step forward in enhancing our product's performance and user satisfaction. It is also in our best interest to continue migrating the rest of this product's websites to use React as that will allow for better performance across the entire product.

REFERENCES

Dougeby. (February 2023). *What is Microsoft Intune*. Learn.microsoft.com. https://learn.microsoft.com/en-us/mem/intune/fundamentals/what-is-intune

Vanderhoof, B. *Meet the developers—React @ meta edition*. (April 2023). https://developers.facebook.com/blog/post/2023/04/12/meet-the-dev-blair-vanderhoof/

Botkin, B. (February 2021). *Our experience migrating to react*. Handshake. https://joinhandshake.com/blog/our-team/migrating-to-react