Enhancing Legacy Java Desktop Applications: A Web Migration Approach with Collaborative Functionality

Antonio Labián
Computer Systems Department,
University of Castilla-La Mancha,
Albacete, Spain
antonio.labian@uclm.es

Jesús D. García-Consuegra
Computer Systems Department,
University of Castilla-La Mancha,
Albacete, Spain
jesus.gbleda@uclm.es

and

Manuel Ortega
Department of Information Technologies and Systems
University of Castilla-La Mancha,
Ciudad Real, Spain
manuel.ortega@uclm.es

Abstract

The need to migrate legacy applications to advanced technology platforms is intensifying due to technological evolution and socioeconomic changes, such as the COVID-19 pandemic. This process seeks to overcome the limitations of traditional desktop applications by integrating advanced capabilities for remote collaboration, highlighting the importance of adapting and preserving the knowledge and experience inherent to these applications. This work delves into the transition of Java Swing applications to web environments, with a focus on the incorporation of collaborative functionalities. Several migration strategies and solutions have been evaluated, highlighting the Vaadin platform for its effectiveness in minimizing the migration effort and facilitating the integration of new collaborative functionalities. Additionally, an innovative approach is introduced in the migration process through the application of generative artificial intelligence for the optimization of user interfaces, which represents significant savings in time and resources. This approach was applied to two case studies, which not only confirmed the feasibility of the process and the technology involved but also allowed the methodologies used to be improved for future migrations of more complex applications.

Keywords: CSCW, Migration, Collaborative web application, Vaadin, Generative artificial intelligence

1 Introduction

The transition from legacy applications to the latest technology platforms is a major focus in both academia and the business sector. This evolution is motivated by the goal of overcoming the restrictions imposed by traditional desktop applications and incorporating advanced remote access and collaboration capabilities[1]. These features are crucial not only for improving technology, but also for enabling organizations to take advantage of remote collaboration. The COVID-19 pandemic has intensified this need, pushing numerous organizations towards the adoption of remote work and online collaboration modalities [2], forcing them to

transform their information systems. There are various methodologies and techniques, such as conversion, reimplementation and wrapping, that propose different strategies to facilitate this migration, minimizing the effort required during the process.

This work focuses specifically on the migration of applications developed in Java Swing to web environments, with a special emphasis on those that integrate collaborative functionalities. Various solutions that adopt different migration strategies have been evaluated, including software reengineering and refactoring. These approaches make it possible to adapt existing applications to new technologies and software architectures without having to completely rewrite the source code. Some authors have also proposed specific methodologies for migrating legacy applications to web environments, such as ADM (Architecture-Driven Modernization) [3]. This evaluation led to a more detailed analysis of the Vaadin platform as an effective tool to facilitate this transition, minimizing the effort required and paying particular attention to the incorporation of new collaborative functionalities. As a result, the initially proposed migration process has been improved by including techniques to analyze and evaluate the usability of the legacy application, using user-centered design principles [4]. This allows not only a technological adaptation but also an improvement in the interaction and experience of the end user.

Furthermore, the exploration of generative artificial intelligence as a method to streamline the migration process has yielded encouraging results, particularly in the design of data entry forms. This technique has emerged as a powerful asset [5], significantly reducing the time required to develop structured user interfaces. By automating aspects of design and layout, generative AI enables a quicker adaptation of legacy applications to modern platforms, thus improving the overall efficiency of the migration process.

As part of this study, two use cases were developed and evaluated to validate the proposed migration procedure and the application of advanced technologies, such as generative artificial intelligence, in the creation of user interfaces adapted to collaborative web environments. The first use case, although based on a real situation, presented a reduced complexity and served primarily to validate the viability of the technology and the established migration process. This initial scenario allowed the applied methodologies to be adjusted and refined, preparing the ground for larger challenges. The second use case, characterized by its greater complexity, more closely replicates the complexity inherent in legacy applications in an organization. This allowed not only to test the robustness of the migration process in a more demanding context, but also to improve said process by facing and resolving challenges representative of future migrations. The successful implementation in this more complex use case demonstrates the ability of the proposed migration procedure to adapt and handle the varied and complex needs of legacy applications, thus ensuring an effective transition to advanced technology platforms.

2 Migration of Java applications to Web

When addressing the migration of a legacy application, there are three approaches applicable to the process: conversion, reimplementation, and wrapping [6].

- The conversion option is based on directly transforming the software from one language to another, manually or automatically. This category includes, for example, transpiler-based solutions.
- Reimplementation involves translating the code into an intermediate language and improving it before producing the new code. This category covers processes that involve software reengineering.
- Wrapping proposes to decompose the system into independent components to treat their migration independently and connect them through service interfaces, being able to do the process incrementally.

This section reviews different frameworks and tools available for migrating Java Swing applications to their web version based on their approach to migration. Table 1 presents a comparison of the frameworks based on criteria that can help to decide on which one to choose in a migration process. One of the aspects evaluated is the possible collaborative support of the frameworks to incorporate collaborative features. An aspect under evaluation is the potential of frameworks to provide collaborative support, enabling the integration of interactive features into migrated applications. Specifically, the emphasis lies on synchronous text editing and presence control mechanisms, which allow participants to identify who is present in the session and facilitate interaction amongst them.

Swing2Script [7] is a solution for migrating Java Swing applications to Ajax-enabled collaborative web applications. The migration process consists of three phases and is supported by programs that automate it. It allows applications to be migrated without having to rewrite the original source code, improving accessibility and usability. However, in addition to the potential complexity of the process, some advanced features may not be fully migrated due to differences between the Java Swing and Ajax platforms.

Table 1: Comparison between the analyzed frameworks.

	Swing2Script	FlexMigration	WebSwing	Vaadin
Scalability	High	High	Medium	High
Web technology	JavaScript	AJAX	JavaScript	WebComp.
Last update	2013	2016	Active	Active
Reusability	High	High	Medium	Medium
Collaborative Support	No	No	No	Yes
Performance	High	Medium	Low	High
Difficulty	Easy	Medium	Easy	Medium
Migration type	Wrapping	Reimplement.	Wrapping / Reimplement.	Reimplement.
Process type	Automatic	Semiautomat.	Manual	Manual
Complex applications	No	No	Yes	Yes
Execution	Client	Client	Client / Server	Server

FlexMigration [3] is presented as a solution to modernize desktop applications developed in Java Swing to web applications using an ADM approach for automatic reverse engineering of the user interface in web migration. The advantages of the ADM approach include the automation of the migration process [3] by obtaining a web interface similar to the original. However, there may be limitations in terms of the complexity and size of the applications to be migrated.

WebSwing [8] is a solution that allows running Java Swing applications on a remote server and stream the user interface over the web using a remote emulation technique. This technique is based on screen capture and event streaming, allowing users to interact with the application as if it were running locally. One of its advantages is that it does not require code rewriting, since the original application is completely maintained. However, remote emulation consumes large server resources and may introduce latency in the transmission of the user interface. Furthermore, the original application still requires maintenance, and the user interface does not improve, so new functionalities such as synchronous collaboration cannot be added.

Vaadin [9] is a platform for the development of web applications with two frameworks and different tools. One of these frameworks is Vaadin Flow, which allows the creation of Progressive Web Applications (PWA) based on web components through Java, following a programming model very similar to that used in Swing. For complex applications, it provides a tool for rendering new views within Swing applications allowing incremental migration. In addition, Vaadin supports real-time collaboration with full awareness management.

3 Framework Vaadin

In an initial phase, Vaadin has been preselected, for its capabilities in application migration and its support for collaborative features, to carry out a migration test of part of a live application developed in Java Swing. At the architecture level, Vaadin Flow runs on the server side, on the Java Virtual Machine (JVM), as part of a Java web application. The application runs on an application server or servlet container, such as Apache Tomcat, Jetty, WildFly, GlassFish, WebLogic or WebSphere, among others. When developing an application with Vaadin Flow, the application logic is written in Java, while the client side, which runs in the user's browser, uses WebComponents and JavaScript. Vaadin Flow takes care of the communication between the server and the browser, allowing to focus on the application logic and user interface without worrying about the details of the communication between the client and the server

3.1 Performance

Server-side execution model environments can be a bottleneck in CPU (Central Processing Unit) or memory consumption. In the case of Vaadin Flow the session size per application instance is quite small, with low CPU consumption. In [10], a study is presented evaluating the resource consumption of a test application and its latency for different volumes of virtual users. Thus, for two thousand virtual users, the CPU consumption was approximately 12.5% with a total memory consumption of 470MB. In this same study it is recommended to dedicate between 1 - 1.5GB of memory for every thousand concurrent users.

3.2 Collaborative features

Vaadin Collaborative Kit is the platform's solution for including collaborative aspects of data synchronization and awareness support in a Vaadin Flow application. The solution, embodied in a library, is composed of a collaborative engine and an extension on the Flow framework components. The extension allows implement real-time collaboration functionalities in Web-Components, abstracting from low-level details such as message distribution, event concurrency management or state synchronization between application instances.

At the view or section level, the awareness support [11] provided focuses on the dimensions of presence and identity of users connected to the same topic. At the component level, the dimensions of presence and identity are maintained, also adding that of location dimension. This support is provided by generating notifications of who is interacting in which component, as long as they work within the same topic.

3.3 Topic

A topic is a data structure that enables real-time collaboration between users in a web application. Essentially, a topic is a collection of items, called categories, that can be accessed, modified or deleted by different users at different times. In this way, users can submit updates to the topic data and receive updates from other collaborating users.

A view can use multiple topics to delimit different collaboration contexts. Users linked by a topic can then share the same awareness. Therefore, the topic is a mechanism for sharing data, as well as a context for delimiting collaboration.

Internally, topics are maps or lists that have a unique identifier within the context of the application [12]. These are created on demand and maintain the overall order of the changes produced. Following the Vaadin server execution model, physical data is shared within the JVM, avoiding data duplication between multiple connections to the same topic. Topic data is not persisted, and its state is lost after the server restarts. The framework provides an API (Application Programming Interface) to interact with the topics as a data producer and/or consumer. Upon establishing a connection in the capacity of a consumer, the individual is systematically notified whenever an update is disseminated by a producer within the pertinent topic.

3.4 CollaborationAvatarGroup

It is the component to display the avatars of the users who are collaborating in a topic. It is a User Interface (UI) component that can be added to a view and automatically updates to reflect the current presence of users in the topic. In addition to displaying avatars, it also provides several options to customize the display of avatars, such as size, layout, and presentation. It is also feasible to regulate the upper limit of avatars presented individually prior to their consolidation into an aggregated list.

3.5 Binder / CollaborationBinder

A binder is a programming component used to bind data between the user interface and a data model in an application. In the context of Vaadin, a binder is a tool that allows linking the fields of a form in the UI with a Java data model object. Thus, when the user interacts with the form fields, the binder automatically updates the corresponding properties of the data model object. Similarly, when data model properties are updated, the binder ensures that the UI fields are updated with the new values. This facilitates data validation, conversion and update between the UI and the data model.

The CollaborationBinder is the extension of the binders provided by Vaadin Collaboration Kit to allow real-time collaboration on forms. By including the CollaborationBinder in a form, it connects its components to a specific topic, allowing changes made to the form to be automatically propagated to other users working on the same topic. It also supports awareness of the linked components, showing both the presence and identity of the users interacting with each component.

This mechanism is the simplest to implement collaborative aspects since it allows visual components to be associated with their categories within a topic. The alternative would be to implement methods for linking the event manager of each component as producer and consumer of data in the topic category.

3.6 Migration process

A migration process from a Java Swing application to its web version with collaborative functionalities has been defined. When upgrading legacy applications, it may be necessary to redesign the user interface to improve usability or user experience issues. In such cases, design techniques or methodologies that evaluate and improve interfaces, such as User-Centered Design (UCD), can be applied.

1. Analysis of the original application to determine by studying the following points:

- Determine if it follows a client-server architecture. If so, identify the communication mechanism it uses.
- At the form level identify if the Model View Controller (MVC) pattern is applied.
- Determine if the application has usability problems. This can be done using common HCI techniques for detecting usability problems, such as user interviews or heuristic evaluation by interface design experts. For views with usability problems, the user-centred design process can be applied to achieve a redesign that satisfies user needs.
- Determine which views are susceptible to collaboration and whether the collaboration context will be general for the entire view or a division by area is required.
- Identify whether unit, integration or acceptance tests exist in the original application.

2. Server migration.
   - In applications with client-server architecture, it is necessary to determine whether to maintain this architecture or to integrate the two parts, taking advantage of the fact that in the new technological environment with Vaadin Flow everything is executed on the server and does not require the development of a communication layer.
   - In the absence of this architecture the most recommendable is not to implement a specific server and perform a single project following the Flow model.
   - If the server is maintained, the model layer will be maintained for database access. In the application, existing classes will also be reused, or entities can be extracted from the server to a new module for import into both the server and the client. However, if the refactoring process can be deepened, the most appropriate would be to integrate client and server following the Flow model since, even extracting the entities to a new module, for efficiency and security it is a good practice to apply the DTO (Data Transfer Object) pattern to use specific objects instead of entities, for data transfer in service calls.
3. Migration of the Swing client application to its web version. The Model View Presentation (MVP) pattern will be applied.
   - Implement the new views. If a direct migration of the original views is applied, add the original existing components. In case of redesigned views, implement them according to the new optimized design to improve usability.
   - If MVC was applied, the original controllers must be refactored, updating the calls to the view references with references to the new objects.
   - If the original application had some type of tests, include them in the new application project and update them with respect to the new development. If there were no previous tests, it is advisable to develop unit and integration tests to validate the integration of the controllers after refactoring.
4. Inclusion of collaborative functionalities.
   - If the application lacked user management, add at least basic management to identify the user.
   - A topic will be created for each view or area.
   - Replace the original binders with collaborative binders, associating them with their corresponding topics.
   - Update existing tests.

4 Case study 1. Registry of Third Parties of public Administration

The Registry of Third Parties of a public administration (Fig. 1) is presented as a case study. The objective of this registry is to unify the persistence of users who use its telematic services, gradually integrating their existing information into other systems and applications.

For its management, a Swing desktop application in Java is used, which is proposed to be improved with a web application that allows better access with collaborative functions, to improve the collaboration of users who use it from different locations. The study identified a user requirement for assistance during the form completion process. To address this, synchronous form editing was incorporated as a collaborative feature. For user coordination, an awareness presence component was introduced, facilitating the identification of

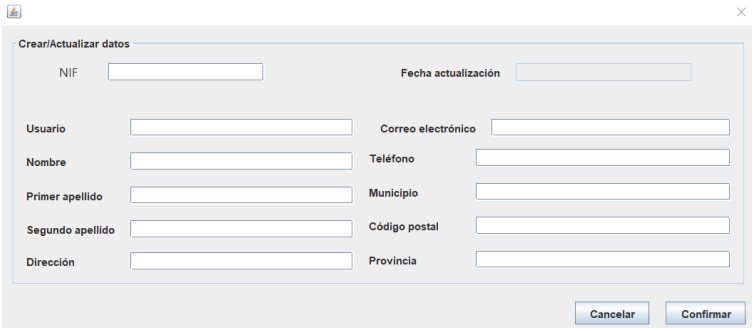

Figure 1: Original application developed with Java Swing.

active users on the same form. Finally, the integration of a chat function enabled communication among participants, enhancing their coordination and preventing potential parallel edits on the same record.

Applying the migration process, the original application was checked to have a client-server architecture communicated by REST (Representational State Transfer) services. It has only five forms, applies MVC, and has no test project.

It was decided to keep the client-server architecture over REST because the server endpoints used by the application to be migrated are used by other external systems. Consequently, the integration of the client and server parts was discarded. The original server entities are also reused, moving them to a new module to import them into the new client project.

Initially, the refactoring process of the original view was discarded when it was found that it was more efficient to generate a new view taking as reference the components of the original view. Other problems were also detected, such as the difference between desktop and web components. Desktop environments tend to use dialogs or small modal forms, which in web are applied more as new views or drop-down panels. This difference beyond aesthetics also implies differences in the reference of objects from the controller. Moving on to the migration of controllers, these are refactored, with changes being very limited to the parts that reference the new presentation. The change in the internal structure of a Vaadin presentation compared to a Swing form also adds more changes than expected.

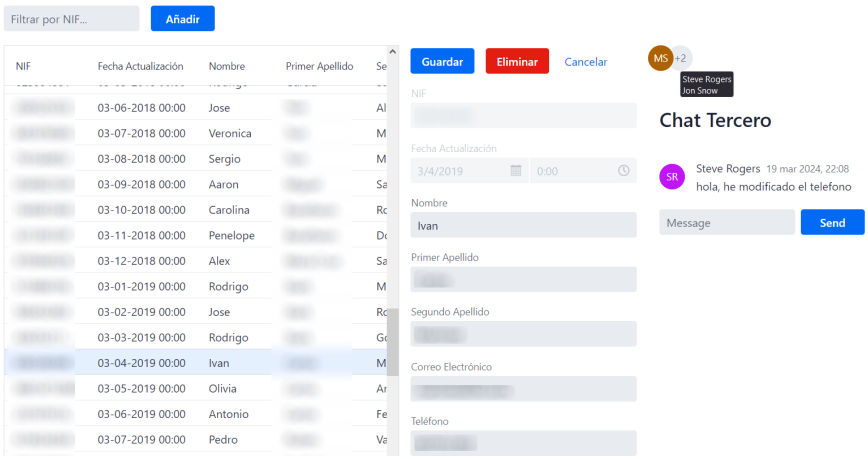

Figure 2: Web application with collaborative aspects.

Once the migration to a functional web version is done, collaborative aspects can be added to the editing form, to include collaborative editing and awareness support in the different fields of the view (Fig. 2).

4.1 Evaluation

The migration process, complexity, and degree of code reuse depend mostly on the patterns with which the original application was designed and the degree of coupling between the different layers and their components. It is also conditioned by the number of resources to be invested in the process since, although at some points of the process there are direct migration options, it is usually advisable to perform a major refactoring, in order to obtain a higher quality maintainable development.

It has been proven that the existence of the MVC pattern facilitates the reuse of existing code, with the advantages that this entails. Thus, the controller has a high degree of reusability. On the contrary, the view layer, despite having a similar programming model between Vaadin Flow and Swing, the differences of use between components and their event handlers makes it more advisable to apply a migration process by reengineering. Moreover, in this test, the MVC pattern allowed the technological incorporation of collaborative aspects to be done quickly. Once the application was migrated, the modifications were limited to creating the topic and changing the binder types of the components for their collaborative version associated with the created topic.

Another factor to be considered in the migration process and in the choice of framework is the conversion of possible technical staff responsible for maintaining legacy applications. The use of Vaadin allows the programming model to represent an evolution and not a total change, which will allow its adaptation to the new environment.

The migration from a Java Swing application to a collaborative web application represents a significant change from a Human-Computer Interaction (HCI) point of view. Changes in user interface, interactivity and collaboration lead to substantial improvements in user experience.

At the interaction level, one of the benefits of this migration is the ability to have real-time collaboration between users. The inclusion of collaborative editing in the data update views of the application, being able to see users online and their activity on the different components of the user interface, has enabled a level of collaboration that is difficult to achieve with Java Swing. Users can now work together on simultaneous tasks, with changes instantly reflected in user interfaces.

The new application follows the single page architectural approach known as SPA (Single Page Application), which provides a more fluid experience by eliminating the need to reload the page [13]. This improvement in the interactive capability of the application is especially relevant for collaborative functionalities.

In addition, the change to a PWA has opened the door to accessibility on a variety of devices. This allows users to use it whenever and wherever they want, freeing them from the limitation imposed by the need for prior installation, typical of Java Swing applications. This type of universal availability further improves usability and accessibility, key aspects of HCI [14]. PWAs can provide a user experience like that of native applications, even on mobile devices [15].

5 Case study 2. Application for Harvest and Wine Declaration

DECOVI (*Declaración de Cosecha y Vinos*) is the web-based computer application of the *Consejería de Agricultura, Agua y Desarrollo Rural* of the *Junta de Comunidades de Castilla-La Mancha* (JCCM) in Spain for the management of grape harvest declarations in the region [16]. Harvest declarations must be made annually by grape harvesters [17], and contain information on the variety and quantity of grapes harvested, the production area, and the destination to be given to the grapes in winemaking or alcohol distillation.

The current DECOVI application (Fig. 3) is developed with JavaServer Pages (JSP) [18] using the Spring framework [19] and Apache Struts [20].

5.1 Migration requirements

The main objective is to migrate to a technology that improves the accessibility and usability of the application under study. JSP-based applications are typically less dynamic and interactive than modern web technologies because of their server-centric approach, where the majority of data processing and application logic occurs on the server rather than in the client browser. JSP generates static HTML content on the server side before sending it to the client browser. This limits the ability to create rich, reactive user interfaces that respond in real time to user interactions. On the contrary, modern web technologies, such as HTML5, CSS3, and JavaScript or frameworks such as Vaadin Flow, allow greater interactivity and dynamism.

An important aspect in the migration of the DECOVI application is to facilitate its maintenance. JSP-based applications face significant challenges in this area, primarily due to the tendency to merge Java and HTML code. This practice results in a code structure that lacks clarity and becomes convoluted, making it difficult to manage effectively. The confluence of business logic with the presentation layer hinders both the tracking and resolution of errors, particularly those that affect the user interface executed in the client's browser. Additionally, the dependence on specific server configurations for JSP is a limitation in terms of flexibility and can complicate processes such as updates and scalability. This scenario increases the complexity and effort required for efficient and continuous maintenance.

As part of the effort to bring the public administration closer to citizens, possible improvements in citizen support will be studied through a more interactive assistance system between users and specialized

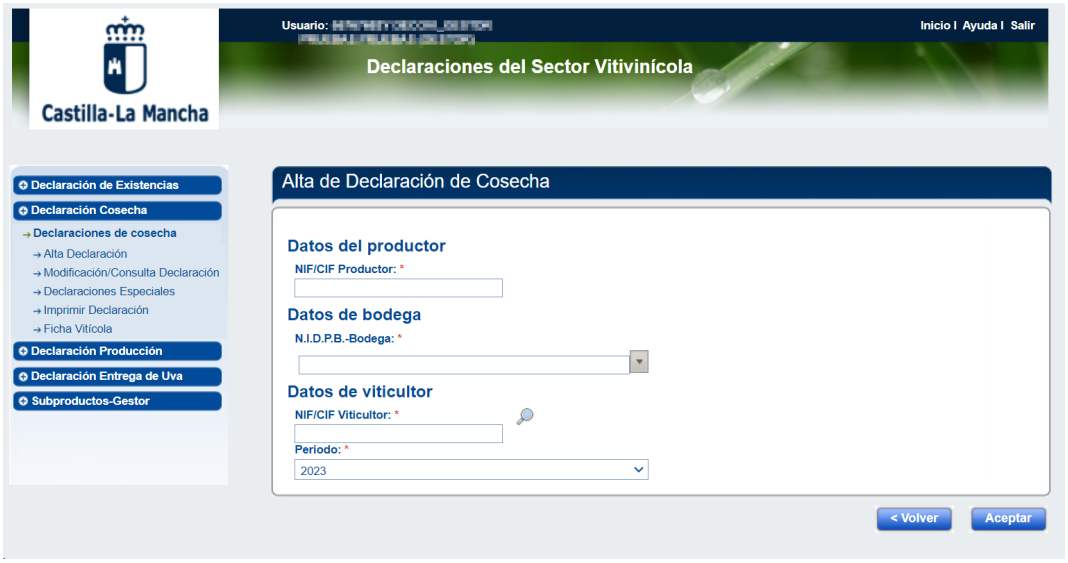

Figure 3: Original DECOVI application in JSP.

administration personnel. This improvement implies the need for advanced technology capable of efficiently managing such interactions and collaboration. The integration of this interactive capability would not only facilitate effective communication between stakeholders, but would also contribute to greater user satisfaction and optimization of administrative resources.

Finally, a critical aspect in IT (Information Technology) system migrations lies in the mastery of knowledge about the problem and the solution accumulated by the human team that has been responsible for its maintenance over the years. To prevent migration from leading to the disconnection of this valuable team, it is essential that the new technology selected maintains a programming paradigm and model that is coherent and comparable to those previously used with JSP and Spring. This continuity in the programming paradigm not only facilitates the technology transition but also ensures the retention of essential technical knowledge and encourages more effective and efficient adaptation by the development team.

Considering these requirements, Vaadin Flow has been selected as the candidate technology for this migration. Table 2 shows a comparison between JSP and Vaadin Flow for the choice of technology in this process. An important aspect is that both technologies, based on Java, adopt an approach oriented towards server-side processing, which facilitates the transition of existing applications in a non-disruptive way, taking advantage of the benefits and advantages of the new technology. In this aspect, the migration to Vaadin will allow the creation of more dynamic and interactive user interfaces thanks to the use of WebComponents, which combined with a more efficient communication scheme allows a better synchronization of the interface with the server. As a result, the user experience is significantly enhanced, offering web applications that are not only responsive but also accessible.

5.2 Collaborative functionalities in the application

In the context of a current application, support for questions and issues is handled through an asynchronous process. This begins by filling out a support request web form. After receiving it, the administration manager contacts the citizen via email or telephone. The introduction of Vaadin Flow opens up new possibilities for improving the assistance to users. This technology allows the implementation of a synchronous help system, in which the user, while working in the application, can request assistance in real time. This collaborative functionality of Vaadin Flow is a significant improvement over the previous method, providing more immediate and direct support.

To implement this requirement for synchronous assistance, Vaadin Flow relies on various dimensions of awareness. These dimensions facilitate the creation of a collaborative environment in which administration personnel can have a more immediate and accurate understanding of the user's situation, enabling more effective and personalized assistance. The specific dimensions of awareness that Vaadin supports for this purpose include:

- Presence awareness. Show users if a manager is available in real time, and know and identify the manager when remotely accessing the user's work view.
- Context awareness. Managers and users can see the same state of data and documents.

Table 2: Comparison between JSP and Vaadin Flow.

	JavaServer Pages (JSP)	**Vaadin Flow**
Programming paradigm	Event and page oriented	Oriented to components (WebComponents) and the construction of user interfaces on the server side
Client-Server Abstraction	Less abstraction, requires handling communication between client and server manually	Greater abstraction, Vaadin handles client-server communication transparently using AJAX
User interface	Mainly HTML based with embedded Java code snippets	Building the user interface on the server side using Java components
Reactivity and real-time updates	May require additional implementations to achieve real-time updates	Provides built-in reactivity, automatic updates thanks to its component-based approach
Maintenance	May require additional efforts to maintain a clear and modular structure	Promotes a more modular and organized structure, with server logic focused on components
Browser support	It may require adaptations for different browsers.	Better support for modern browsers, including mobile technology.
State management	States are generally handled manually using sessions and request parameters.	Vaadin automatically handles the state of the components and offers an event-driven programming model
Flexibility and control	Offers greater flexibility by directly integrating server logic into web pages	Provides a more controlled, component-centric approach, simplifying the development of complex user interfaces
Compatibility	Compatible with Java EE technologies and allows integration with various frameworks	Is compatible with Java EE, Spring and other Java frameworks.
Popularity and community	Widely used, with a long history and an established community	Increasingly popular, with a growing community

- Intention awareness. Send and receive notifications when a participant creates or edits an item. Enable real-time communication, through a chat, to share information about the question or issue and coordinate

5.3 Migration process

In the proposed migration process, we started with a detailed analysis of the legacy application. It was identified that the application does not adopt a client-server architecture. All processing, including generating pages for users, is performed on the server. Furthermore, it was observed that the legacy application implements the Model View Controller (MVC) pattern, using the Apache Struts Framework. This framework contributes significantly to the architecture of the application, facilitating the effective separation between the different layers: the presentation layers, the controllers and the data model. Apache Struts allows actions implemented in controllers to handle the logic associated with user interfaces, while the data model remains accessible and independent of these actions and the rest of the application. The usability of the 21 views of the original application was also analyzed through a heuristic evaluation; only usability problems were detected in the parcel edition view (Fig. 4). This view is the most complex since it combines the editing of four tables. The main problem detected was the table with components. To solve it, it was decided to replace this table by a new view to add the grape varieties and their associated data one by one, instead of having all the fields visible initially to include information for up to three grape varieties.

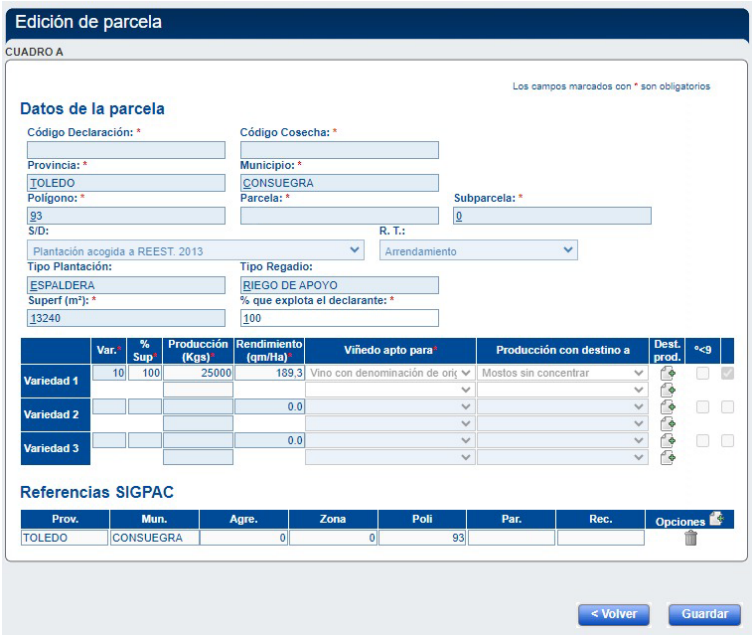

Figure 4: View for parcel edition.

In the second stage of the process, the application was migrated, keeping the architecture and part of the original structure, developing a single project with Vaadin. This application does not provide service to other applications. Therefore, it is not necessary to maintain or develop communication mechanisms with other external components. The order followed in the migration strategy was model, view and controller.

Although in the migration process, the possibility of directly transferring the existing model was initially considered, it was decided to refactor it. This allowed the elimination of unnecessary obsolete dependencies in the new implementation and the optimization of the mappings of some entities. This step is crucial to ensure that the model is more consistent and efficient within the context of the updated application.

Subsequently, the focus moved to migrating JSP views. Given the differences between the technologies involved, it was determined that direct reuse of these views was not feasible. This involves developing new views from scratch. It was identified that the layout and arrangement of the components in the interface represent the most demanding aspects in terms of time in the development of a view. In response to this, three different strategies were explored to optimize this process, seeking to maximize efficiency without compromising the quality and functionality of the user interfaces.

The first evaluated strategy consisted of creating views directly through the source code, taking the original views of the application as a visual reference (Fig. 3). This approach involved an iterative process of code modifications to achieve the desired visual design, characterized by its gradual and adjusted nature. The second strategy focused on the use of Vaadin Designer [21] for the design and construction of visual interfaces. This tool accelerates development, compared to direct programming, since it eliminates the need to know in detail the properties of each interface component. However, the code automatically generated by this tool is longer, reaching triple the number of lines compared to the code written manually.

The third strategy explored was the use of generative artificial intelligence for the development of views. In this case, ChatGPT [22] with image recognition capabilities was used. The procedure consisted of providing screenshots of the original views of the application as input and specifying a prompt for the generation, such as: “Using the provided image, generate in Vaadin Flow a view with the same elements and distribution” (Fig. 5). This technique resulted in obtaining approximate views, particularly useful for data capture forms or tables. On the other hand, in more complex views (Fig. 4) with tables that integrate components or with large density of components, the generated code does not fulfill the expected result.

Once the views were developed using Vaadin, the controllers were migrated. These controllers, which in the context of Apache Struts are called ”actions”, were originally associated with JSP views. It is important to note that the degree of reusability of these controllers varies significantly depending on the visual components of each view and their functionalities. The main reason for this variability lies in the need to modify the references to HTML elements in the controllers to adapt them to the WebComponents used in Vaadin. This adaptation is a critical aspect of the migration process, as it involves a substantial change in the way controllers interact with the user interface. Conversion HTML references to WebComponents not only requires technical adjustment, but also a detailed understanding of how these new references will affect

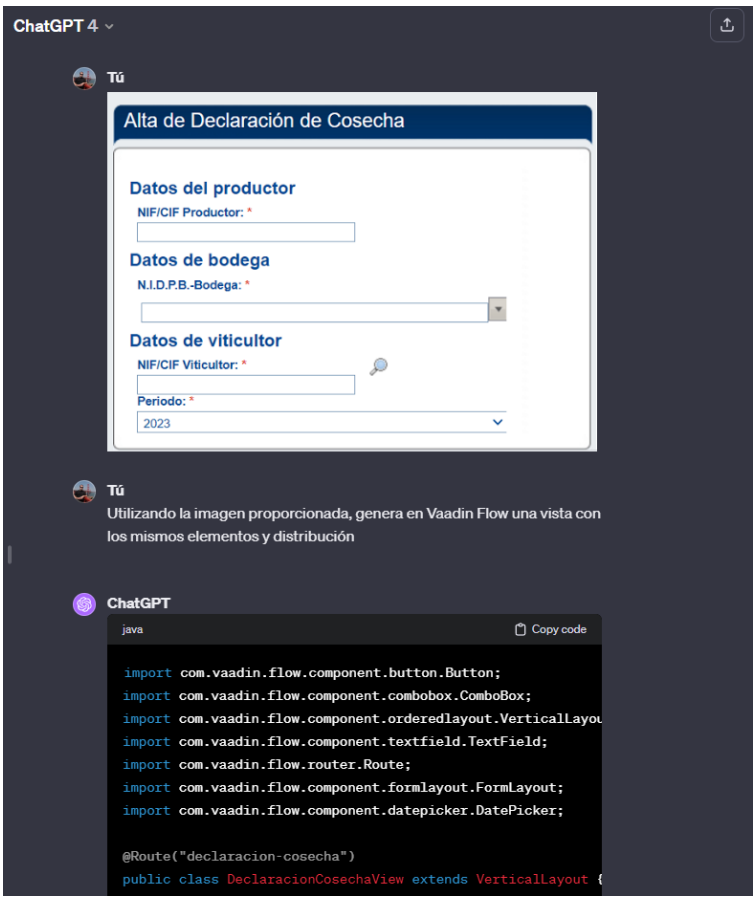

Figure 5: View migration using generative AI.

the logic of the controllers and, therefore, the overall functioning of the application.

5.4 Evaluation

At the development level, the controller migration is the part that requires the most effort in the process. Although parts of Struts actions are reused, the change between JSP and Vaadin components requires analyzing the differences in interface and use of each element, as well as refactoring or changing each associated method. In the actions that have been migrated to controllers with less impact, 34% of lines have been changed in the code, including the method headers, while in the actions with greater need for changes, up to 89% of lines have been changed with some modification. In these latter cases, although an almost complete reimplementation is achieved, having the base code is an aid for the analysis of the scenarios and cases that would otherwise have to be completely analyzed and defined.

Another point that requires effort in the migration is the development of the views. For this task, as mentioned in the migration process, three strategies were evaluated. The use of generative artificial intelligence was of great value, since it allows to reuse the views of the original application and perform a migration for the most common views such as data forms. As an estimate of the performance improvement using generative AI for the migration of interfaces, it has been found that form-type views, implemented directly with source code, with an average development time of two hours, have been generated by approximately 25% over time, with very similar final results.

Due to the reduction of files required in the project and the simplicity of the new code, the maintenance of the new application is improved compared to the original. The change from Struts and JSP represents an approximate 30% reduction in the configuration files and Java interfaces used in Apache Struts to relate views and actions. In addition, compared to JSP, Vaadin Flow's programming model is more maintainable, as it does not combine java and HTML code on the same pages and orients new development to components, reducing the complexity of the source code.

In usability and accessibility, the Vaadin UI components used in the new application (Fig. 6) are accessible and responsive by design. This ensures that the migrated application is more accessible to users with different skills and on different devices. This is complemented by support for adaptive and responsive design, which

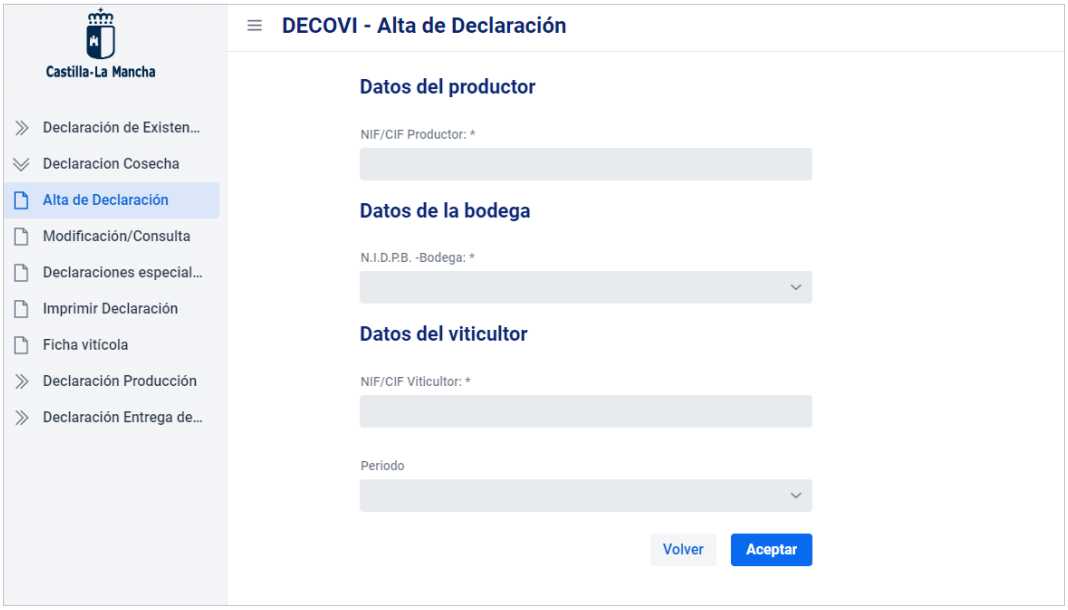

Figure 6: Migrated DECOVI application in Vaadin.

ensures that the new application displays optimally on a wide variety of screen sizes. Presentation, as a Single Page Application (SPA), provides a smoother and faster user experience compared to traditional web applications. The automatic client-server synchronization offered by Vaadin Flow reduces errors and improves the user experience. Finally, the new application allows customization through themes that could be used to adjust the look and feel and user experience to meet specific accessibility and usability needs.

6 Conclusion and future work

In this work, two legacy Java Swing applications were upgraded and migrated to collaborative web applications using the Vaadin framework. The first allowed the technology and process to be validated. The second was used to migrate several common technology challenges in the organization's legacy desktop applications and to improve the proposed migration process to include the analysis and, if necessary, redesign of views with usability problems following the user-centered design methodology. In addition, the second application has been used to test the migration of visual interfaces using generative artificial intelligence. It is considered a very interesting option to continue analyzing in future works.

The primary factor for selecting Vaadin was its ease in creating fast, collaborative applications and its straightforward architecture that facilitates efficient migration. The migration process to a web application was simple following the methodology proposed by Vaadin with more than 20 years of experience in this process. However, advantages were appreciated in the reengineering for the GUI (Graphical User Interface). Several limitations were identified in the Vaadin collaborative solution, the most notable being the lack of persistence of the awareness history and, by design [12], the event is not associated with the producing user. In addition, data management is limited because it is independent of the application logic. Finally, management of topics has weaknesses with complex data structures and nested data collections.

Acknowledgment

The practical cases presented have been developed within the framework of an assignment to *Tragsatec* itself by the *Consejería de Agricultura, Ganadería y Desarrollo Rural*, within the program "*Expansion of electronic administration in the field of agricultural production and the agri-food sector with the purpose of promoting the social and economic development of rural areas*". This program is co-financed by the European Agricultural Fund for Rural Development (EAFRD), under the Rural Development Program of Castilla-La Mancha 2014-2020, within action 7.3 - Technical Assistance.

The authors would like to express their gratitude to the *Consejería de Agricultura, Ganadería y Desarrollo Rural* and *Tragsatec* for their collaboration in the development of the case presented.

References

[1] S. Murugesan, "Understanding web 2.0," *IT Professional*, vol. 9, 2007. doi: 10.1109/MITP.2007.7810.1109/MITP.2007.78

[2] S. Lund, W.-L. Cheng, D.-S. Aaron Dua, Andre, O. Robinson, and S. Sanghvi, "What 800 executives envision for the postpandemic workforce," *Mckinsey Global Institute*, 2020.

[3] S. Mbarki, N. Laaz, S. Gotti, and Z. Gotti, "Adm-based migration from java swing to ria applications," *International Journal of Information Systems in the Service Sector*, vol. 8, 2016. doi: 10.4018/IJISSS.2016040108

[4] D. A. Norman and S. W. Draper, *User centered system design; new perspectives on human-computer interaction*. L. Erlbaum Associates Inc., 1986.

[5] D. Baulé, C. Gresse von Wangenheim, A. Von Wangenheim, J. Hauck, and E. Júnior, "Automatic code generation from sketches of mobile applications in end-user development using deep learning," 2021. doi: 10.48550/arXiv.2103.05704

[6] H. M. Sneed and C. Verhoef, "Cost-driven software migration: An experience report," *Journal of Software: Evolution and Process*, vol. 32, 2020. doi: 10.1002/smr.2236

[7] H. Samir, E. Stroulia, and A. Kamel, "Swing2script: Migration of java-swing applications to ajax web applications," 2007. doi: 10.1109/WCRE.2007.48. ISSN 10951350

[8] Webswing, "Brings java, javafx, netbeans, applet to browser — webswing," 2024. [Online]. Available: https://www.webswing.org/en

[9] Vaadin, "Vaadin — discover the web app framework built for java," 2024. [Online]. Available: https://vaadin.com/

[10] ——, "Vaadin scalability," 2024. [Online]. Available: https://vaadin.com/scalability

[11] C. A. Collazos, F. L. Gutiérrez, J. Gallardo, M. Ortega, H. M. Fardoun, and A. I. Molina, "Descriptive theory of awareness for groupware development," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, 2019. doi: 10.1007/s12652-018-1165-9

[12] L. Åstrand, "Collaboration engine architecture - summer 2021 edition · vaadin/collaboration-engine · discussion 49 · github," 2021. [Online]. Available: https://github.com/vaadin/collaboration-engine/discussions/49

[13] M. Mikowski and J. Powell, *Single Page Web Applications: JavaScript End-to-End*, 1st ed. Manning Publications Co., 2013. ISBN 1617290750

[14] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. Pearson, 2016. ISBN 013438038X

[15] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, *Progressive Web Apps: The Possible Web-native Unifier for Mobile Development*, 1 2017.

[16] JCCM, "Decovi," 2024. [Online]. Available: https://decovi.castillalamancha.es/decovi

[17] ——, "Orden 110/2020, de 3 de agosto, de la consejería de agricultura, agua y desarrollo rural, por la que se modifica la orden de 01/09/2015, de la consejería de agricultura, medio ambiente y desarrollo rural, por la que se regula la presentación de solicitudes y declaraciones obligatorias en el sector vitivinícola de castilla-la mancha," 2020. [Online]. Available: http://docm.jccm.es/docm/eli/es-cm/o/2020/08/03/110

[18] E. Foundation, "Jakarta server pages," 2024. [Online]. Available: https://projects.eclipse.org/projects/ee4j.jsp

[19] Broadcom, "Spring framework," 2024. [Online]. Available: https://spring.io/projects/spring-framework

[20] A. S. Foundation, "Apache struts project," 2024. [Online]. Available: https://struts.apache.org

[21] Vaadin, "Java ui builder design tool," 2024. [Online]. Available: https://vaadin.com/designer

[22] OpenAI, "Chatgpt," 2024. [Online]. Available: https://chat.openai.com/