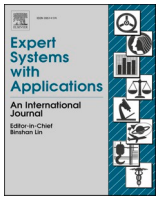
Expert Systems With Applications 255 (2024) 124491

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Enhancing data tables user experience via automated adaptations

Mario Díaz Ceñera [a,*], Julián Grigera [b,c,d], Jordán Pascual Espada [a]

[a] Department of Computer Science, University of Oviedo, C. Federico García Lorca 18, 33007 Oviedo, Asturias, Spain
[b] LIFIA, Facultad de Informática, Universidad Nacional de La Plata, Argentina
[c] CONICET, Argentina
[d] CICPBA, Argentina

ARTICLE INFO

Keywords:
User experience
Tables
Expert system
HTML

ABSTRACT

User experience is a crucial aspect of software quality, and while various guidelines and heuristics have been proposed to enhance it, tables have not received as much attention as other elements. Additionally, many developers are unaware of the guidelines they should follow in this regard. The objective of this article is to enhance the user experience on websites by streamlining the development process of usable tables. To achieve this, we propose a set of heuristics integrated into a tool called BetterTable that enables automatic application of usability improvements while parsing the HTML document. The complexity involved in developing a table with applied usability guidelines is evaluated using different implementations: the proposed approach, DataTables, and AntDesign. Furthermore, tests are conducted to validate the effectiveness of the proposed heuristics through user interactions with 13 table usage scenarios, measuring time, mouse movement, and clicks. Each user is randomly presented with 13 scenarios that either meet or do not meet a specific guideline. The results indicate that our proposal reduces implementation complexity by 43.83% and 24.69% compared to DataTables and AntDesign, respectively. Moreover, user testing reveals that 6 out of the 13 guidelines show improvements in at least 2 of the 3 calculated metrics. Guidelines fulfilled show 40.37% less average completion time and 20.39% less mouse movement than the baseline. Based on the conducted studies, we observed that compliant tables, on average, exhibit reduced user time and mouse movements compared to non-compliant tables. However, no significant differences are found in terms of clicks.

1. Introduction

Since the creation of the internet, web interfaces have experienced an evolution driven by the need to improve the user experience, usability, and to make the web easier for any type of user (Hassini, 2023). Usability is an important feature of software quality (ISO 9126, 1991), which can be defined as *"the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"* (ISO 9241-11:2018, 2018). Several heuristics, rules or guidelines (Altin Gumussoy, 2016; Borges et al., 1996) have been proposed to improve usability. Despite this, some research works suggest that the level of knowledge about usability guidelines by web developers can be quite limited (Alonso-Virgós et al., 2019, 2020). Since usability improvement can be costly for the developers and designers of websites (Juristo et al., 2007), several CSS libraries have been introduced over the years to help develop web interfaces (W3Techs, 2023).

One of the most important aspects to consider when developing a web interface is the way the information is presented to the user, like paragraphs, list of elements, images, links, or tables. This is affected by the specific circumstances of the website (Spyridakis, 2000). Therefore, another relevant issue arises: the understanding of the data.

In the last few years, new proposals have appeared to simplify the work of developers in different areas related to visualisation, like data analytics (Dibia & Demiralp, 2019; Golfarelli & Rizzi, 2020) or HTML Interfaces (Aşiroğlu et al., 2019; Halbe & Joshi, 2015; Mgheder & Ridley, 2008; Okamoto et al., 2006; Pawade et al., 2018). By applying these proposals to improve usability, it would be possible to generate usable interfaces in a way that is transparent to the developer, avoiding the problem of unfamiliarity described above. This would involve analysing the information on the website (Akbar et al., 2016) and modifying its interface.

One of the key elements on a website are tables. They are usually used to display large amounts of data. Consequently, it is relevant to

* Corresponding author.
E-mail addresses: uo257354@uniovi.es (M.D. Ceñera), julian.grigera@lifia.info.unlp.edu.ar (J. Grigera), pascualjordan@uniovi.es (J.P. Espada).

https://doi.org/10.1016/j.eswa.2024.124491
Received 26 June 2023; Received in revised form 28 May 2024; Accepted 10 June 2024
Available online 22 June 2024

consider how to display this information. Some designers and usability experts have proposed more than 30 usability guidelines oriented to tables (Coyle, 2017; Hellmuth, 2019; Huijing, 2019; Rafalowicz & Nguyen, 2017; Redfield, 2021; Shepherd, 2023; Shestopalov, 2019). Some proposed improvements are the inclusion of graphics, allowing the user to perform actions on the table or improving styles. Nevertheless, despite the existence of these guidelines, many developers still struggle to generate tables with a good user experience because they are not aware of them (Alonso-Virgós et al., 2019, 2020).

Hence, the main objective of this article is to enhance the user experience of using tables on websites while streamlining the development process for developers by providing transparent usability improvements. To accomplish this, a proposed library is introduced that analyses and identifies usability issues within tables, enabling the automatic application of usability enhancements. This proposal will be referred to as BetterTable throughout the article.

The rest of the article is divided as follows: Section 2 shows the related work, Section 3 describes the proposal, BetterTable, Section 4 describes the methodology and the evaluation of the proposal. Finally, Section 5 presents conclusions and future work.

2. Related work

Many software products suffer from usability issues, which can significantly impact their usage. It is beneficial to incorporate usability techniques from the outset of a software project rather than applying them later (Juristo et al., 2007). While some companies can afford the assessment of usability experts, not all organizations have access to them, so developers often share the responsibility (Barnum et al., 2005). Despite the significance of usability, many web developers are unfamiliar with usability guidelines and fail to implement them consistently in their work (Alonso-Virgós et al., 2019, 2020). It is common for production websites to contain usability errors that could be rectified by following usability guidelines (Alarcon et al., 2014; Vargas et al., 2011). Furthermore, certain errors tend to occur frequently (Cappel & Huang, 2007). Several reasons contribute to this situation. Sometimes web developers lack adequate knowledge of usability guidelines and other techniques to enhance the user experience. Alternatively, developers may recognize the poor user experience but prioritize other aspects of application development, such as functional features, due to cost constraints (Etemadi et al., 2018).

Numerous research papers provide mechanisms for enhancing usability, with guidelines and heuristics being two prominent approaches (Altin Gumussoy, 2016; Bader et al., 2017; Borges et al., 1996; Hvannberg et al., 2007; Nielsen, 1994; Spyridakis, 2000). Guidelines and heuristics provide developers with concrete instructions to follow. Many of these guidelines can be objectively validated to determine whether a website adheres to them or not (Alonso-Virgós et al., 2020). While implementing all the guidelines does not guarantee excellent usability, they serve as valuable tools for many web developers. In addition to general usability guidelines compilations (Mariage et al., 2006), there are also specific guidelines available for domain-specific applications (Altin Gumussoy, 2016). Unfortunately, only a few of them include estimates of the potential improvement in user experience (Borges et al., 1996). Finding studies that support the efficacy of implementing all usability guidelines is a challenging task. While there are specific studies available for certain patterns, most patterns lack empirical evidence. This can be attributed to the difficulty in quantifying the concrete benefits of a pattern, as interface performance can be influenced by various factors and combinations of patterns. While these guidelines propose logical ideas, quantifying the expected gains from their implementation remains a challenge.

Tables are a key part of web interface development. They are present in many relevant web applications (Cafarella et al., 2008). On many occasions, they are the only way to show complex data. Also, sometimes the use of tables can improve usability, in contrast with other representations. Many experts and web designers proposed several guidelines to design more usable tables (Coyle, 2017; Hellmuth, 2019; Huijing, 2019; Laubheimer, 2022; Rafalowicz & Nguyen, 2017; Redfield, 2021; Shepherd, 2023; Shestopalov, 2019).

Several proposals aim to address the challenges associated with maintaining websites in terms of usability and design guidelines. Researchers have suggested methodologies to develop user-centered web applications that enhance usability (Sukamto et al., 2020). Similarly, frameworks have been introduced to reduce usability issues (Geng et al., 2014). Additionally, there are tools available to assist developers by automating the detection of certain usability problems (Ruiz-Rodríguez, 2006). Although these automated tools only address specific aspects and a small portion of website usability, some proposals go beyond problem detection and offer potential solutions (Grigera et al., 2017). Unfortunately, the focus of these proposals is limited to a few security aspects, and no research has been found that addresses usability problems related to tables. This research focuses on analysing the website itself and combining it with user interaction data to propose adaptations, without providing automatic improvements for table data.

There are some proposals that offer systems for analysing HTML table data, for example analysing table data to generate automatic table joins (Akbar et al., 2016), extracting specific information for HTML tables (Embley et al., 2005) or identify important fields. Very few of the proposals that analysed table data did it to get features related to usability or user experience improvement (Chen et al., 2000). The proposal (Marriott et al., 2013) provides an algorithm to automatically adapt tables in a compact way, no proposal waves on more advanced usability aspects. Most automated solutions for tables tend to focus on how the table is displayed (Tajima & Ohnishi, 2008), but do not cover aspects such as the nature of the data being displayed or what data iteration mechanisms to use, such as paging, filtering, etc.

One possible of solution that could overcome developers' lack of usability knowledge or lack of time are proposals focused on the automatic generation of usable interfaces. Data2Vis (Dibia & Demiralp, 2019) generates automatic data visualizations from datasets, the proposal is based on a declarative language. The resulting interfaces are mostly graphical data, not usable tables. SkyViz (Golfarelli & Rizzi, 2020) is another similar alternative, describing a domain-specific language used to translate data sets into the most appropriate visualizations. These visualizations are primarily focused on graphical components, not tables. Aşiroğlu et al. (2019) and Halbe and Joshi (2015) are automatic tools for creating HTML websites from a GUI design paper or mock-up, respectively. This could be useful for saving developers time but not include usability considerations. Proposal copies the user's design, so it does not add any usability improvements.

WIDE (Okamoto et al., 2006) is a proposal which tries to assist developers minimizing their efforts in implementing. They could create conceptual models and the proposal automatically generates web application, it does not include usability considerations or work on tables. Mgheder and Ridley (2008) analyses data of a database and automatically generate web user interfaces. For creating appropriate user interfaces, it explores the possibility of adding metadata in the database tables. Although it appears that the system could include tables within its web interfaces, they have no special usability considerations in the representation of those tables, they look purely like the data in the database.

Numerous research studies have proposed guidelines to enhance user interfaces, particularly focusing on tables as they are a crucial element in many web interfaces. While there are several proposals and tools available to simplify the development of web interfaces, there is a lack of solutions specifically aimed at improving table usability. Some proposals have analysed web table information for various objectives like data extraction or table merging. However, in terms of usability, these proposals primarily focus on adapting tables to different devices, neglecting considerations such as data representation and the inclusion of interaction tools within the proposal itself.

3. BetterTable

We propose a solution based on an automatic library which developers can include in their website. This solution analyses the data of HTML tables using a set of heuristics and selects the most appropriate usability guidelines to improve the user experience. Then the library automatically applies that adaptation to the HTML code.

The proposal allows to automatically improve usability in HTML tables without the developer having to spend time in implementing the usability guidelines. The solution is also for developers who are not aware of usability guidelines in tables.

3.1. Detection of UX guidelines needs and application process

Tables are complex UI elements, since they show structured data, sometimes with many columns. This can be sometimes confusing for users, which struggle with issues like finding information, making comparisons between row values, or making sense of the different data formats.

The research work starts from a list of **26 UX guidelines** for HTML tables. These usability guidelines were obtained from various usability experts, style guides and research papers (Coyle, 2017; Hellmuth, 2019; Huijing, 2019; Laubheimer, 2022; Rafalowicz & Nguyen, 2017; Redfield, 2021; Shepherd, 2023; Shestopalov, 2019). Unfortunately, the improvement on user experience that comes from the application of most of the guidelines has never been analysed through rigorous research work, the reality is that we don't know how effective each guideline could be for improving user experience. The complete list of analysed guidelines is the following:

1. Ability to sort elements. The ability to sort various columns streamlines data searches and analysis.
2. Filter by columns values. Allowing the filtering of values in multiple columns independently or simultaneously is a powerful tool for conducting searches.
3. Logical sorting by default of columns. The statistically most consulted columns should be placed first.
4. Logical sorting by default of rows. The most relevant and statistically consulted rows should be placed first.
5. Zebra striping for improved readability. Zebra striping aids in keeping the focus on the same row during reading.
6. Correct alignment of textual and numerical values. For ease of reading and comparison, texts are left-aligned, and numbers are right-aligned, consistently using the same decimal format.
7. Fixed headers. Fixing the table headers ensures the column names are always visible, which is particularly useful in tables with many rows.
8. Fixed columns. The use of a fixed main column, such as the person's name, helps maintain reference in tables with numerous columns.
9. Appropriate spacing between rows and columns. Very small spacing can decrease reading speed. The spacing between rows and columns should be appropriate or have options to allow for adjustment.
10. Icons and mini graphs. Numeric or percentage values can be recognized and compared more quickly and visually by including mini graphics.
11. Cell colouring: In some cases, applying cell colouring can expedite the recognition of potential values, such as positive or negative numbers, values in different ranges, or states that are part of a list.
12. Highlighting the focused row: The row that is currently selected by the mouse cursor should be highlighted, allowing the reader to identify it more quickly.
13. Choice of columns to display. When a table has numerous columns, the visualization often demands extensive scrolling, and in some cases, irrelevant columns are displayed. Allowing configuration for the visible columns is a good option to enhance the user experience and tailor the information to the user's needs.
14. Choose column as main column. Often, rows have an identifying value, somewhat like the 'primary key' in a database table. This text should be emphasized over other fields using bold formatting, providing a visual reference for the most distinctive value of the row.
15. Correct vertical alignment data in the top. The text should start at the top of the cell, not in the middle, especially when some fields have many lines and others have few. This helps to facilitate a more efficient reading patter.
16. Including a row with summaries. Including summaries in the first row: When the table contains numeric and/or comparable data, it is useful to include a summary row beneath the header row. This summary can incorporate percentages, variations, etc. A notable example of this practice is observed in Google Analytics tables.
17. Pagination. Instead of displaying all results on a single page, they are divided into several pages. This approach is employed to avoid excessive scrolling.
18. Column resizing: When tables contain columns that are either too large or too small, providing a resizing mechanism is appropriate, allowing users to adapt the table to their needs. In some cases, resizing may also include column reordering.
19. Actions on table elements: If there is a need to allow actions on an element in a table, include actions directly on the row itself, such as edit or delete. This can be quicker than requiring the user to enter details. If there are many actions, consider progressive disclosure and make them appear only when the row is in focus.
20. Simultaneous actions on multiple elements. In some applications, users routinely perform actions on multiple elements within a table. In these cases, including a mechanism for multiple selection can simplify the interaction. This functionality is particularly useful in tables that allow multiple actions between elements. It becomes even more powerful when combined with filters and a "select all elements" option.
21. Allow direct editing of the field. In some applications, editing information in tables is quite common, especially for certain fields. Instead of including an edit view, enabling direct editing of the fields can simplify user interaction.
22. Expanded view of rows. In some cases, rows contain a substantial amount of information that cannot be displayed without scrolling. If the row includes less important columns, you can collapse that information, allowing users to expand the row to view the complete information. This approach is often more efficient than using a separate detailed view.
23. Avoid long text or descriptions in columns. Excessively long text can lead to inappropriate column sizes. Consider using abbreviations or shortened strings, while also providing the option to access the complete text or more detailed information.
24. Last Update Date. When displaying time-dependent data or information acquired at regular intervals, include the timestamp in each column along with the value of the table's latest update.
25. Column Grouping. In the case of having many columns, try to group them logically or conceptually. If possible, this allows users to start exploring concepts from more abstract to more concrete ones.
26. Adaptation in case of not fitting in container: Avoid double scrolling. If the container is too small, the table should not have both horizontal and vertical scrolling. It is highly recommended to implement automatic adaptation for mobile devices instead.

Some of these guidelines must be applied on very specific data types or very specific conditions, while other guidelines are recommended to be applied on all tables.

We combined the previous information with the analysis of **50**

popular websites to detect the usability guides followed in their tables. These 50 popular sites were obtained from a list of pages with high traffic. We analysed these pages using a bot that searched for tables, as many pages barely used tables. Once the tables from these pages were obtained, they were analysed with the guidelines proposed by various authors. Based on this information, we could get an idea of how the table design guidelines were being applied on important/popular websites.

In the end, we compiled a selection of 11 guidelines for this research work. The decision to select these guidelines was based on various factors. First, an effort was made to choose guidelines that seemed important, meaning they were included in several important websites. Second, to exclude guidelines that apply only in very specific conditions and rarely appeared on important websites. Third, to eliminate guidelines that were commonly included in other UX libraries, such as "Zebra striping for improved readability." Fourth, to eliminate guidelines that require excessively complex data analysis and could be the subject of a second, more in-depth investigation focused on data analysis and understanding. For example, "Logical sorting by default of columns" requires understanding the context of the entire information to determine which columns users are likely to consult more frequently to achieve the best order.

It was decided to select a set of guidelines that could ensure both the detection and application of the guideline could be carried out through heuristic rules. The remaining guidelines will be the focus of a second research that will be more based on the analysis of table data. Finally, the set of selected guidelines is as follows:

1. Ability to sort elements.
2. Logical sorting by default.
3. Correct alignment.
4. Fixed headers.
5. Icons and mini graphs.
6. Cell colouring.
7. Choice of columns to display.
8. Choose column as main column.
9. Pagination.
10. Abbreviation for long descriptions.
11. Adaptation in case of not fitting in container.

These guidelines do not cover 100 % of the usability guidelines on tables, but they represent a large set of them that can mark a good user experience when interacting with a table. With those patterns, we defined a **set of heuristics to determine when the usability pattern should be added to the table**, based on the idea that (1) perhaps not all guidelines should apply to all data sets, but that (2) some guidelines may only affect large data sets or (3) specific data types like numbers, etc. These heuristics are listed in Table 1.

Table 1
Heuristics designed from usability patterns.

Design pattern	Heuristic	Actions
Ability to sort elements	If the number of rows is greater than 5	Add ascending/descending sorting arrows
Logical sorting by default	If the column is numeric or textual	In case of numerical columns, sort from most to least, in case of textual columns, sort alphabetically.
Correct alignment	If the column is numeric	Align the numbers to the right. The decimals of all cells in the column are equalised.
Fixed headers	Always applies	Add fixed headers
Icons and mini graphs	If a column has textual values, the number of unique values is between 2 and 10 and does not exceed 30 % of the total number of values.	Label each field with a colour.
	If within a column the header cell includes the percentage symbol (%) and all the values in the column are between 0 and 100	Add graphical percentage bar
	If within a cell there is a link	Add favicon. Try to replace the URL with the page title to make it more descriptive.
Cell colouring	If a column is numeric and there are positive and negative values	Colour positive numbers green and negative numbers red. Also, colour with different intensities depending on whether it is higher or lower.
Choice of columns to display	If a table has more than 8 columns	Display a drop-down to select the columns you want to select.
Choose column as main column	If in the first column all the values are different	Highlight it in bold
Pagination	If the number of elements is greater than 10	Enable pagination to display fewer items per screen and browse through them per page
Abbreviation for long descriptions	If the cell contains more than 75 characters	Limit the characters to 40, displaying the remaining text in a tooltip.
Adaptation in case of not fitting in container	If the table does not fit on the screen	Limit the visible columns, adding a button that allows you to expand the information.

3.2. Solution details

We propose a client-side JS library, which developers can include in the view/HTML of their application (Fig. 1). The library works as follows: (1) First, it can be configured in different ways; in the default configuration it tries to apply as many improvements as possible to the table. (2) In the analysis a set of heuristics is applied to determine if it is relevant or not to apply each guideline. (3) Once it is determined which guidelines are going to be applied, an adaptation for applying the guideline to the content of the table is generated, (4) this application results in modifications to the original table. Finally, the modifications are applied to the HTML (5), so that the table is modified to improve the user experience in a transparent way for the developer.

Web developers will include HTML tables within their web application, on the web page they will refer to the proposed solution, which is a JavaScript library. The table will be labelled in a specific way indicating that they want to use the proposal. The Code 1 snippet shows a module type script tag, which is created containing the call to the library.

Code 1. *Script module tag to execute the library.*

```
<script type="module">
    import { execute } from "../js/Executor.js";
    document.addEventListener("DOMContentLoaded",
        function () {
            execute("table_id");
        });
</script>
```

The previous snippet of code is a default example which will use all analysis and implemented without any specific configuration. The user can optionally specify the adaptations to be made, and the location, by adding configuration parameters. The Code 2 snippet of HTML shows a specific configuration.

The connection between the library and HTML table code is established by the execute() function. This function creates a UsableTable object, which is the main part of the library, and calls its execute() method. It can be seen in Code 3.

Code 2. *Configuration parameters of the execute call.*

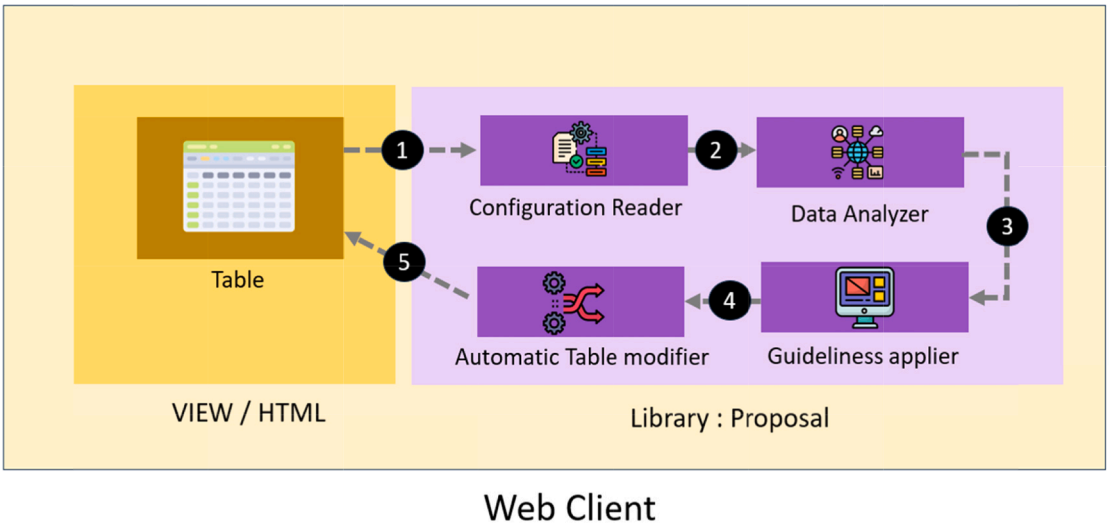


Fig. 1. Solution architecture.

```
<script type="module">
    import {execute} from "./js/Executor.js";
    document.addEventListener("DOMContentLoaded",
function () {
        execute("table_id", {
            orderElementsPosibility: true,
            defaultOrder: true,
            correctAlign: true,
            fixedHeader: true,
            iconsAndGraphics: false,
            cellColor: false,
            columnSelector: false,
            principalColumn: false,
            pagination: true,
            responsive: false,
            largeDescriptions: false,
            locale: "es",
        });
    });
</script>
```

Code 3. *Connection module between the HTML and the library.*

```
import { UsableTable } from "./UsableTable.js";

function execute(tableId, configuration = {}) {
    new UsableTable(configuration, tableId)
        .execute();
}
export { execute };
```

3.2.1. Configuration reader

Based on the configuration indicated by the user in the call to *execute ()*, the adaptations can be deactivated or activated, indicating for each property true if it is desired to be activated and false otherwise. The properties that can be specified are:

- orderElementsPossibility
- defaultOrder
- correctAlign
- fixedHeader
- iconsAndGraphics
- cellColor
- columnSelector
- principalColumn
- pagination
- responsive
- largeDescriptions.

In addition, the locale property can be set. This allows the library to be able to read numbers in different languages. Supported values are texts indicating a language code, such as “es” (Spanish) or “en” (English).

After receiving the user’s configuration, the table is read by rows and columns, to store this data internally and make it easier to exploit.

3.2.2. Data analyzer & guidelines applier

After the initial loading of the configuration and table data, the analysis is performed. Each guideline has been encapsulated in a command. Algorithms have been defined for each case and are listed below. Those that will always be applied are not listed: zebra colouring, fixed header and marking the selected row.

1. Ability to sort elements. **Analyse** if the number of rows in the table is greater than 5. That **analysis returns** a boolean. If the analysis is positive, the **action** will be modifying the table to allow ascending/descending sorting arrows in all columns.
2. Logical sorting by default. Loop through the columns and **analyse** if the column is numeric. That **analysis returns** a true or false for every column of the table. In case of true, columns sort in decreasing order, otherwise column is textual, sort alphabetically. The **action** involves going to the HTML table, include sort in every column and applying an ascending/descending order.
3. Correct alignment. The pseudocode of the algorithm can be seen in Algorithm 1. Loop through the columns and **analyse** whether it is numerical. This **analysis returns** a boolean. If true, it checks the number that has the highest number of decimal places. Then, the values in the column are looped through and, if the current value has fewer decimal places than the maximum, it is stored in a pre-initialised array with [originalNumber, newValue]. There are **two actions**:
   a. Access the table, scroll through the array of filled decimals, and update the affected numbers.
   b. Align the numerical columns to the right updating table styles.

Algorithm 1 Correct alignment

```
1: procedure execute(enabled, tableId, elements, options, locale)
2: Initial state:
3:    options − Array of options
4: Input:
5:    enabled − Boolean value to indicate if the adaptation is enabled
6:    elements − Array with all values read from the table
7:    tableId − Id of the table element tag
8:    locale − Locale
9: Output:
10: if enabled then
11:    if size(elements) > 0 then
12:       columns ← getColumns(elements)
13:       for all column ∈ columns do
14:          if isColumnNumeric(column, locale) then
15:             maxDecimals ← getMaxDecimals(column, locale)
16:             decimals ← [ ]
17:             for all value ∈ column.values do
18:                if getDecimals(value, locale) < maxDecimals then
19:                   decimals.add(adjustDecimals(maxDecimals, value))
20:                end if
21:             end for
22:             applyChanges(options, tableId, decimals, locale)
23:          end if
24:       end for
25:    end if
26: end if
27: end procedure
```

4. Icons and mini graphs. Three adaptations apply to this case. The pseudocode of the complete algorithm can be seen in Algorithm 2.
   a. Label unique values with colours. The columns are looped through and **analysed** for being textual. This analysis **returns** a boolean. If true, an array is obtained with the unique values among all the values of the column. In case they are not more than 30 % of the total values and they are between 2 and 10, the adaptation is applied. The **action** involves creating labels with the colour assigned to each value.
   b. Percentage bar. Columns with a percent (%) symbol in the header are searched for. In these columns, it is **analysed** if all values are numeric and between 0 and 100. If true, the adaptation is applied. The **action** consists of creating a bar chart with the percentage.
   c. All the links in the table are **searched** for and their favicon and link description are obtained. The **action** includes updating the links to add the favicon and replace the text of the link with its description.
5. Cell colouring. The pseudocode of the algorithm can be seen in Algorithm 3. All columns are looped through and **analysed** if they are numeric. This analysis **returns** a boolean. If true, the column is checked if there are both positive and negative values. If so, the adaptation can be applied. The **action** consists of changing the CSS of the elements affected. For both negative and positive values, three cut-off points are obtained to increase the colour intensity. After that, the values are looped through, and their styles are updated.

Algorithm 3 Cell colouring

```
1: procedure execute(enabled, tableId, elements, locale)
2: Input:
3:    enabled − Boolean value to indicate if the adaptation is enabled
4:    tableId − Id of the table element tag
5:    elements − Array with all values read from the table
6:    locale − Locale
7: Output:
8: if enabled AND size(elements) > 0 then
9:    columns ← getColumns(elements)
10:   for all column ∈ columns do
11:      if isColumnNumeric(column) then
```

(*continued on next column*)

(*continued*)

Algorithm 3 Cell colouring

```
12:         if havePositiveAndNegativeValues(column) then
13:            positiveSplits ← getPositiveSplits(column.values)
14:            negativeSplits ← getPositiveSplits(column.values)
15:            for all value ∈ column.values do
16:               if value > 0 then
17:                  applyPositiveColour(value, positiveSplits)
18:               else if value < 0 then
19:                  applyNegativeColour(value, negativeSplits)
20:               end if
21:            end for
22:         end if
23:      end if
24:   end for
25: end if
26: end procedure
```

6. Choice of columns to display. The **analysis** checks the number of columns and return how many columns has the table. If it is more than eight the solution applies. The **action** consists of modifying the original HTML code of the table for including a menu in the top of the table, which is linked to the visibility of all table columns.
7. Choose column as main column. **Analyse** all the columns of the table checking if there are columns in which all values are unique. This **return** true if all values of column are unique. If so, the adaptations can be applied. The **action** consists of bolding the entire column using a CSS styles.

Algorithm 2 Icons and mini graphs

```
1: procedure execute(enabled, tableId, elements)
2: Input:
3:    enabled − Boolean value to indicate if the adaptation is enabled
4:    tableId − Id of the table element tag
5:    elements − Array with all values read from the table
6: Output:
7:    if enabled AND size(elements) > 0 then
8:       columns ← getColumns(elements)
9:                              ▷ Label unique values with colours
10:      for all column ∈ columns do
11:         if isColumnTextual(column) then
12:            uniqueValues ← getUniqueColumnValues(column)
13:            if size(uniqueValues) <= size(elements) * 0.3 then
14:               if size(uniqueValues) ∈ [2, 10] then
15:                  labelColours ← assignColours(column, uniqueValues)
16:               end if
17:            end if
18:         end if
19:      end for
20:      applyLabelColours(tableId, labelColors)
21:                              ▷ Percentage bar
22:      columnsWithPercentage ← getColumnsWithPercentage(columns)
23:      for all column ∈ columnsWithPercentage do
24:         if isColumnNumeric(column) then
25:            if checkAllValuesBetween0And100(column) then
26:               applyPercentageBar(tableId, column)
27:            end if
28:         end if
29:      end for
30:                              ▷ Links
31:      links ← getAllLinks(tableId)
32:      for all link ∈ links do
33:         favicon ← getFavicon(link)
34:         description ← getDescription(link)
35:         applyLink(link, favicon, description)
36:      end for
37:   end if
38: end procedure
```

8. Pagination. **Analyse** if the number of rows of the table is bigger than 10. This analysis **returns** a boolean. If true, the adaptation applies. The **action** consists of modifying the code of the table to divide it into pages, including a menu in the down part of the table to allow movement among pages.
9. Abbreviation for long descriptions. Scroll through all the cells in the table. **Analyse** if the number of characters in the cell is greater than 75. This test **returns** a boolean. If true, the adaptation is applied. The **action** involves modifying the HTML to reduce the text length to 75, adding ellipses (…) at the end, and include a tooltip containing the full text.
10. Adaptation in case of not fitting in container. **Analyse** whether the table fits on the screen. This analysis **returns** a boolean. If true, the adaptation is applied. The **action** consists of removing columns that do not fit in the table and include an extra button which allow the user to expand the information by clicking on a (+) button that will show the other columns from top to bottom.

3.2.3. Table modifier

Most of the actions result in a modification of the data table, data structure or style. Modifications are made in different ways depending on the guide. Modifications can be divided into four categories:

- Modifications that do not affect the table data, but properties. These are, for example, pagination or sorting. These modifications are applied through an array of options. This array will be one of the parameters used to initialise the table of the DataTables library on which we rely.
- Modifications that affect the content. For example, when a number that appears in the table as "2″ is modified by "2.00″ to adjust the decimals; or when we reduce the number of characters. In these cases, the elements are in the DOM and the modification to the content is made.
- Modifications that require the generation of new elements in the DOM. This is the case of progress bars or favicons for links. The steps to make them are to locate the cell where the change is to be made, create a new HTML element of the corresponding type, and modify the HTML DOM where the cell is located by adding the new element.
- Modifications that affect styles. This is the case of cell colouring, setting positive values to green and negative values to red, in different shades. These modifications are made by locating the elements in the DOM and adding a class containing the respective CSS.

3.3. Use case

In this section we show an example to depict how the automated usability improvements can help users interact or search content in HTML data tables. Consider a car magazine wants to make a list of sales of different vehicle brands. For each car, information is displayed such as:

- Name
- Brand
- Link
- Price
- Percentage of sales (%)
- Price change

An example with data is shown in Fig. 2.

Developers of the company do not have enough time or expertise to create a table that uses good UX guidelines, so they decide to use the proposed solution.

Using the library, cases where usability improvements can be applied would be detected:

- The column with the name of the cars would be bolded.
- Car brands would be marked with a distinctive colour each.
- Links would have an icon on their left.
- The numbers would be aligned to the right and the decimals would be the same for all.
- A percentage bar would be displayed next to the percentage number.
- In the case of price change, positive values would be green (the higher, the darker), while negative values would be red (the lower, the darker).
- Pagination and column selector would be added.

After activating the library improvements, the table shows may differences, as can be seen in Fig. 3.

Search:

Name	Brand	Link	Price (€)	Sales percentage (%)	Price change
SEAT Arona	Seat	https://www.seat.es	19.920,50 €	16%	-360,5 €
SEAT Ibiza	Seat	https://www.seat.es	17.760 €	20%	-180 €
SEAT León	Seat	https://www.seat.es	23.230,65 €	11%	+230 €
SEAT Tarraco	Seat	https://www.seat.es	36.590 €	7%	-1.080 €
SEAT Ateca	Seat	https://www.seat.es	27.160 €	10%	-60 €
Volkswagen Golf	Volkswagen	https://www.volkswagen.es/es.html	25.645 €	13%	+380 €
Volkswagen Polo	Volkswagen	https://www.volkswagen.es/es.html	16.945,66 €	17%	-340 €
Volkswagen T-Cross	Volkswagen	https://www.volkswagen.es/es.html	21.580 €	10%	+170 €
Volkswagen Tiguan	Volkswagen	https://www.volkswagen.es/es.html	33.150 €	8%	-720 €
Volkswagen Touareg	Volkswagen	https://www.volkswagen.es/es.html	68.695 €	4%	-900 €
Toyota Corolla	Toyota	https://www.toyota.es/	22.990 €	8%	-210 €
Toyota C-HR	Toyota	https://www.toyota.es/	25.500 €	7%	+100 €
Toyota RAV4	Toyota	https://www.toyota.es/	36.650 €	4%	+230 €
Toyota Highlander	Toyota	https://www.toyota.es/	63.300 €	2%	-1.420 €
Toyota Supra	Toyota	https://www.toyota.es/	61.750 €	1%	-160 €
Ford Focus	Ford	https://www.ford.es/	24.850 €	5%	-200 €
Ford Fiesta	Ford	https://www.ford.es/	17.400 €	8%	+300 €
Ford Puma	Ford	https://www.ford.es/	22.600 €	3%	-100 €
Ford Kuga	Ford	https://www.ford.es/	30.950 €	2%	-400 €
Ford Mustang Mach-E	Ford	https://www.ford.es/	49.500 €	1%	+500 €
Audi Q3	Audi	https://www.audi.es/	37.030 €	67%	+200 €

Fig.2. Original HTML table without automatic adaptations.

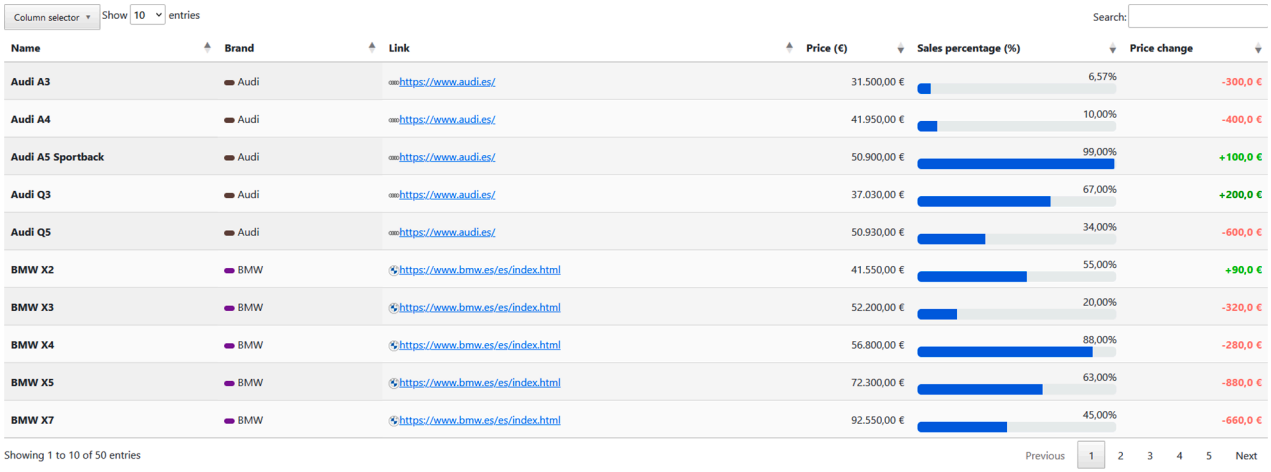


Fig.3. Table with automatic adaptations.

4. Evaluation

To validate the objective of this research, an evaluation against the prototype was carried out. This section will describe the tests that we ran to validate the degree to which the automated usability improvements of our library benefits both developers (Section 4.1) and end users (Section 4.2).

4.1. Gain for the developers

4.1.1. Methodology

The first test verifies the gain that is produced by applying the library's automatic adaptations, compared to what it would have taken the programmer to perform the application of these UX guidelines manually. In addition, it must be considered that the programmer may not know all the guidelines. Based on the methodology used in (Espada et al., 2015), the following measurements were taken:

- L: number of lines to achieve the implementation.
- T: number of characters, including spaces, to achieve the implementation.
- E: number of specific elements (JS functions, styles, HTML tags, react components or variables) to achieve the implementation.
- F: number of external libraries and external styles to achieve the implementation.

Code automatically generated such as the skeleton of a React project was not considered.

The table used was the one mentioned in Section 3.3. All implementations had the same characteristics. However, small variations in the styles may occur because completely different libraries were used. Measurements were taken for the following implementations:

- BetterTable.
- DataTables: one of the reference libraries on the web for representing tables, includes a simple way to implement typical table functionalities such as pagination, formatting, sorting. It can be used as a standalone library, but it is also integrated in many UI frameworks, like Material-UI, PrimeReact, Vuetify.
- AntDesign: this implementation was based on the React and AntDesign libraries. It is the second most popular React UI framework, used in some well-known websites like Alibaba, Baidu and Tencent. Ant design includes very powerful rich components for rendering tables with great care in usability.

Once the data has been obtained, the factor resulting from applying a weight to each metric was calculated. These weights were L = 2, T = 1, E = 3, F = 4. With this factor a comparison of the implementations was made globally. This weighting assignment is based on ordering each of the actions performed from least to most time-consuming and requiring many keystrokes. The chosen weights could lead to extensive discussion. That's why excessive weightings were not established, and consecutive values 1 to 4 were simply assigned instead. The aim of this methodology is not to obtain a "perfect complexity score" it is to establish comparisons of implementation complexity that are replicable and that other authors can challenge or improve current proposal.

4.1.2. Data analysis

The Table 2 shows the results obtained by each implementation for each of the metrics.

After applying the corresponding weights to each metric, we obtained the results for all 3 approaches, shown in Fig. 4.

In the evaluated scenario, the use of BetterTable reduced the code implementation factor by 43.83 % compared to DataTables, and 24.69 % compared to AntDesign. The proposal's improvement over AntDesign is less significant compared to DataTables, likely because AntDesign is more specialized in UX and inherently applies some of the UX guidelines included in this research. Some of the style definitions that require multiple lines of code in DataTables are already included by default in Ant Design or can be implemented more easily by simply adding an attribute. In both cases, the advantage of the proposed solution is achieved due to the automation that significantly reduces the lines of code needed to define a table which includes the UX guidelines. However, in practice, it is not just about including the code to implement these UX guidelines; developers must also possess expert knowledge to understand the specific UX requirements for the specific data they want to display.

4.2. Usefulness of the guides to improve the user experience

4.2.1. Methodology

As mentioned above, there are many compilations of guidelines, but

Table 2
Number of lines, characters, specific elements and external libraries and styles for each implementation.

	L	T	E	F
BetterTable	545	15,663	2	1
DataTables	922	27,930	20	3
AntDesign	896	20,332	43	2

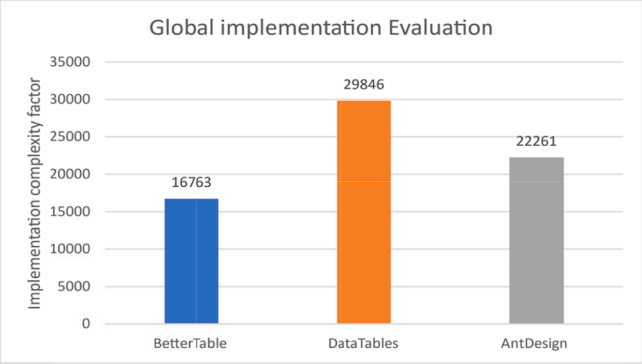


Fig. 4. Global implementation's complexity factor of the three implementations applying the described weights to each measurement.

very little evidence to prove the effectiveness of applying a particular guideline. The second part of the evaluation tries to estimate the benefit that the application of each guideline included in the solution can bring, applying an evaluation like the one conducted in (Borges et al., 1996). The procedure was the following:

1. For each guideline, a test scenario was created, with a table in a different domain. Each scenario had two variants that were randomly assigned to the user: one complied with the usability guide and the other did not.
2. The user was asked to search for a text on the table, according to the given instructions.
3. The time, clicks and mouse movement needed to complete the task were measured.

The scenarios created for the evaluation are described in the Table 3.

The evaluation was conducted by 42 users. All of them were regular users of web applications. For the tests, they use their own devices, without any assistance. Prior to initiating the actual tests, an example was provided to familiarize the users with the required tasks. It should be noted that some users may encounter difficulties and be unable to complete the tests. To mitigate this issue, a "Skip" button was implemented, allowing users to bypass a particular task if needed. This information is also recorded for reference.

After gathering the data, outliers were identified and removed by calculating the interquartile range (IQR) and obtaining a factor of 1.5 times the IQR. This factor was then added to the third quartile (Q3) and subtracted from the first quartile (Q1).

The resulting values represent the upper and lower limits, respectively, for accepting data points.

The analysis began by examining the time spent, clicks performed, and movement required by each user to complete the test for each guideline. The data was segregated based on whether the UX guideline was met or not.

Next, an investigation was conducted to determine whether there was any correlation between guideline compliance and the variables of time, clicks, and user movement. To assess this relationship, tests of independence were performed.

The initial step involves determining whether the sample adheres to a normal distribution. Two tests can be employed for this purpose: the Kolmogorov-Smirnov test when the sample size exceeds 50, and the Shapiro-Wilk test otherwise. The following assumptions are established.

Once normality has been verified, tests were performed to assess the independence of the samples. Two scenarios are considered based on whether the distribution is normal or not. If sample is normal, then independent samples *t*-test were used. Otherwise, Mann-Whitney *U* test. The effect size in both scenarios were calculated using Cohen's d.

Table 3
Scenarios for evaluation with end users.

Guideline	Domain	Instructions
Ability to sort elements.	Audience of TV channels	Select the name of the channel with the highest number of viewers.
Logical sorting by default	Clothing sizes	Select the smallest clothing size number.
Correct alignment	Time spent executing algorithms in different programming languages	Select the name of the experiment with the highest time in milliseconds.
Fixed headers	Price of different fuels at different service stations	From stations 41 and 42, select the name of the station with the highest AdBlue price.
Icons and mini graphs (percentages bar)	Car prices and valuations	Select the name of the car with the highest approval rating.
Icons and mini graphs (links)	Sportswear	Click on the Adidas brand link.
Icons and mini graphs (labels)	Bills issued to multiple users and their status.	Each row shows a receipt issued to a user and its status. Select the name of the user with the most receipts in PAID status (in any of the rows where it appears).
Cell colouring	Results of a bet	Given the score obtained in a bet, select the name of the player who has lost the most points.
Choice of columns to display	Formula 1 GP results	Given a table with the random number of places achieved in an F1 Grand Prix by each driver, click on the name of the driver who placed 10th in the Qatar Grand Prix.
Choose column as main column	Works of art	Select the name of the museum that has the painting "The Creation of Adam".
Pagination	Countries information	Select the currency of Somalia.
Adaptation in case of not fitting in container	Films, critics and awards	Select the review note of the film with most awards
Abbreviation for long descriptions	Videogames	Select the name of the video game that has been released before.

4.2.2. Data analysis

At the end of the period given to users to complete the test, a total of 473 tests were done, by 42 different users. Of all the tests carried out, 256 valid tests have been obtained that comply with the proposed UX guidelines and 213 valid tests that do not (469 valid tests for different guidelines, four tests were discarded due to apparent corruption of information due to a technical failure).

On each session we captured the participants' completion time, clicks and mouse movement. The Table 4 shows those metrics average with or without applying the proposed guideline. In the case of those that are met, the percentage of variation with respect to those that are not met is also shown. Negative values, indicated in green, represent patterns that result in a gain compared to those that are not met. In red, the opposite case. Those that have been skipped are not considered for these measurements. 1.5IQR has been applied. Table 4. Also shows the number of users who pressed the skip test button and the number of users who closed the evaluation program.

In 8 out of 13 patterns, time shows improvement. Mouse clicks and mouse movements exhibit varying results, with 6 and 4 improvements respectively. Fig. 5, Fig. 6 and Fig. 7 show the average time, movement and clicks, respectively, required by users to complete each test, depending on whether the guideline is fulfilled or not.

Following the proposed methodology, the sample was tested for normality. The factors are "Complies" and "Does not comply" and the variables are time, clicks and mouse movement. A confidence interval of

Table 4
Time, clicks and mouse movement for each guideline, depending on whether the pattern is met or not.

Guideline	Not met			Met			Abandonment		
	Time	Clicks	Mouse movement	Time	Clicks	Mouse movement	Users start Test	Users press inSkip Test	Users quit the evaluation
Ability to sort elements	17668.94	2.12	204.47	11370.00	3.15	286.05	42	2	1
				−35.65 %	48.75 %	39.90 %		4,76 %	2,38 %
Logical sorting by default	16000.07	1.80	220.47	8353.14	1.90	222.33	41	4	0
				−47.79 %	5.56 %	0.84 %		9,76 %	0 %
Correct alignment	20525.50	2.60	202.44	18102.18	2.21	376.85	41	6	0
				−11.81 %	−14.98 %	86.16 %		14,63 %	0 %
Fixed headers	12871.91	1.73	213.62	15848.61	1.88	279.25	41	4	1
				23.13 %	8.98 %	30.73 %		9,76 %	2,44 %
Icons and mini graphs (percentages)	23126.25	1.83	213.62	15990.29	1.76	198.24	40	3	1
				−30.86 %	−3.90 %	−7.20 %		7,50 %	2,50 %
Icons and mini graphs (links)	5917.14	1.00	213.62	5323.21	1.21	142.53	39	1	0
				−10.04 %	21.05 %	−33.28 %		2,56 %	0 %
Icons and mini graphs (label)	21385.50	2.80	219.57	18207.82	1.88	330.81	39	2	1
				−14.86 %	−33.04 %	50.66 %		5,13 %	2,56 %
Cell colouring	11729.75	2.07	219.57	10738.47	1.44	203.82	38	2	1
				−8.45 %	−30.27 %	−7.17 %		5,26	2,63 %
Choice of columns to display	14978.13	1.40	213.62	17706.00	1.07	316.21	37	1	1
				18.21 %	−23.47 %	48.03 %		2,70 %	2,70 %
Choose column as main column	4539.30	1.30	213.62	6848.81	1.52	142.90	36	1	0
				50.88 %	17.22 %	−33.10 %		2,78	0 %
Pagination	13320.65	1.56	245.79	14709.73	3.64	266.07	36	1	1
				10.43 %	132.73 %	8.25 %		2,78 %	2,78 %
Adaptation in case of not fitting in container	9807.59	1.71	245.79	11275.77	1.73	291.69	35	0	0
				14.97 %	1.25 %	18.68 %		0 %	0 %
Abbreviation for long descriptions	14975.00	2.31	248.60	12565.35	1.75	288.62	35	0	0
				−16.09 %	−24.32 %	16.10 %		0 %	0 %

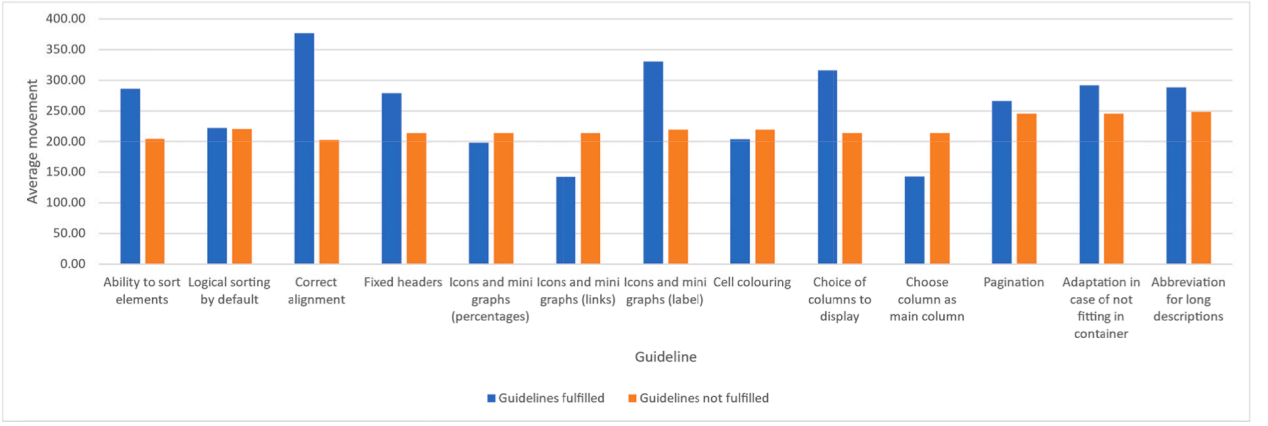


Fig. 5. Average time to pass each test when the established guideline is fulfilled or not fulfilled.

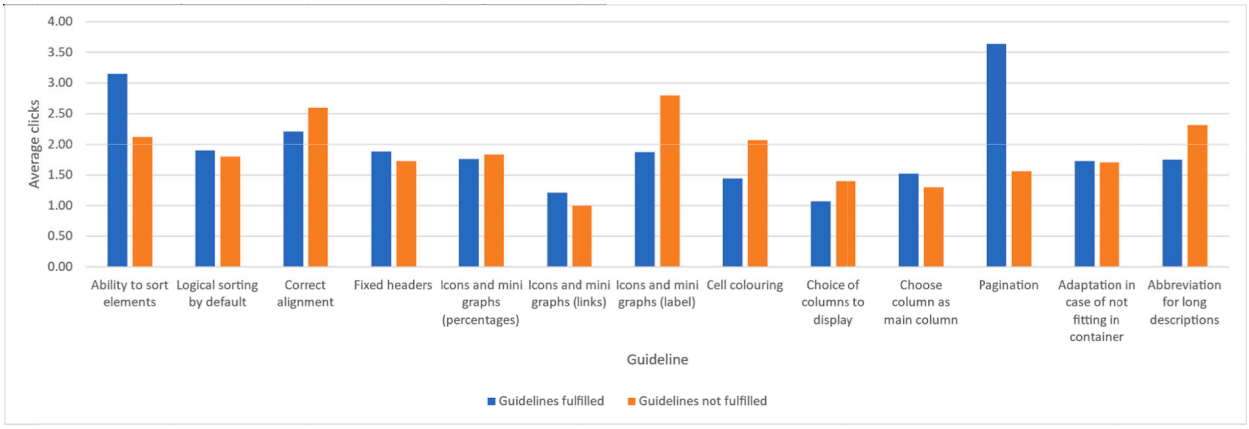


Fig. 6. Average clicks to pass each test when the established guideline is fulfilled or not fulfilled.

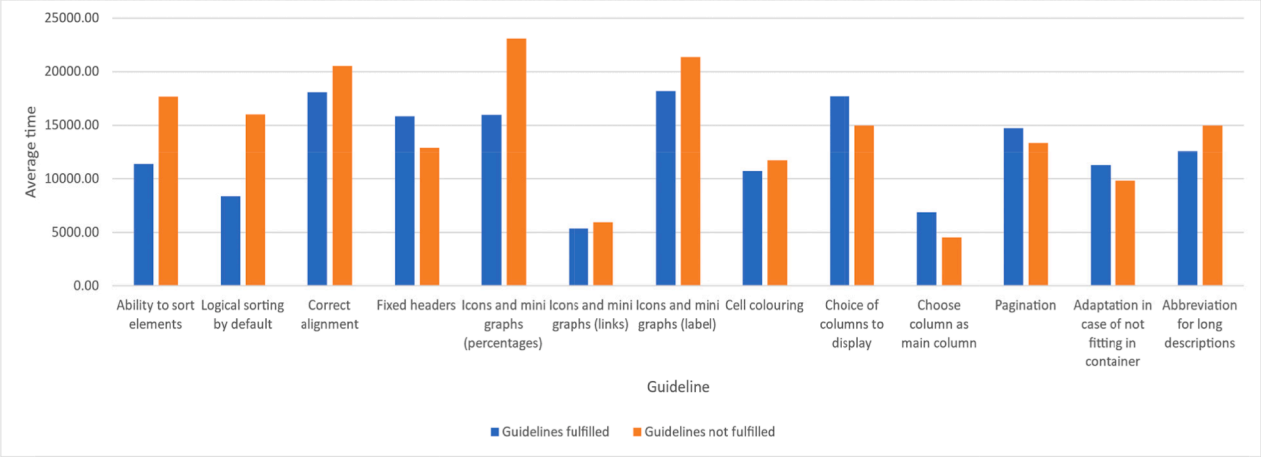


Fig. 7. Average movements to pass each test when the established guideline is fulfilled or not fulfilled.

95 % was established. Also, as the sample size is larger than 50, the Kolmogorov-Smirnov test was run.

Table 5 shows the results obtained after running the tests. In view of the results, H0 is rejected (p-value < 0.05), so the distribution is not normal. Thus, the Mann-Whitney test would be used to test for independence. The test results, displayed in Table 6, include the p-value and Cohen's d. Regarding the time and mouse movement variables, the analysis indicates statistically significant differences in completion time and mouse movements based on the fulfilment of the UX guideline, with a medium effect observed (p-value < 0.05; U(time) $= 0.40$; U(movement) $= 0.22$). Users who encountered compliant patterns exhibited 40.37 % less average completion time and 20.39 % less movements compared to users who encountered non-compliant patterns. Conversely, for the clicks variable, no statistically significant differences were found, indicating a very low effect (p-value > 0.05; U $= 0.02$).

Even if most guidelines showed improvements across the observed metrics (6/13 were better in at least 2 out of 3 metrics), some other were indecisive, and others showed worse results than in the "guideline not met" version. Those that performed worse, i.e. the guidelines that did not show efficiency improvements were: Fixed Headers (−23,13 %), Choice of Columns to Display (−18.21 %), Choose Column as Main Column (−50,88 %) and Pagination (−10,43 %). Some of these could be due to the complexity that's inherent in the operation of the widgets, such as "Choice of Columns to Display", "Choose Column as Main Colum" and even "Pagination"; in these cases, given the simplicity of the task to perform, it could be more difficult to operate the controls for these features than to simply search by hand. Some other cases, like "Fixed Headers" went unnoticed by some participants, as confirmed by them in some informal surveys. More experimentation should be carried out to really test these features.

Considering the metrics separately, time, which is the most relevant metric in our opinion, was only noticeable higher in "Choice of columns…" and "Choose column as main…". Another interesting result is that the click count in "Pagination" was much higher than in the unapplied version, which could be linked to the operation of the paginator widget. Mouse movement was noticeable higher in "Ability to Sort Elements", "Correct Alignment" and "Icons and Mini Graphs", but we found this metric to be more difficult to interpret in the sessions we ran.

Regarding the abandonment rate of the evaluation by users. There were some long tests in which users had to read and analyse much information, this contributes to diminishing the motivation of some users. The highest abandonment rate occurred in the early parts of the evaluation, it also the longest tests. According to the data, less motivated users abandoned the test in the early stages, first by skipping some questions and eventually closing the evaluation program. There is also a very interesting result, over the total of 27 skips, 19 (70,37 %) were in tests that did not fulfil the Guideline and 8 (29,63 %) in tests that fulfilled the guideline.

Table 5
Results of Kolmogorov-Smirnov normality tests for the three variables.

Variable	Factor (Complies/Does not comply)	p-value
time	complies	<0.001
	does not comply	<0.001
clicks	complies	<0.001
	does not comply	<0.001
mouse movement	complies	<0.001
	does not comply	<0.001

Table 6
Results of Mann-Shitney tests (p-value and d cohen) of each variable.

Variable	p-value	d cohen
time	0.002	0.40
clicks	0.246	0.02
mouse movement	0.013	0.22

5. Conclusions and future work

The main objective of this research is to enhance user experience reducing the developer's efforts when creating tables on web. Implementing usability guidelines requires spending time in development, plus there seem to be a lot of developers who don't know the guidelines. BetterTable is a JavaScript library that can be included in web applications to analyse the information in the table and modify it automatically. The modifications made to the table are based on the usability guidelines that are appropriate to include in each specific table, based on their properties and data. This solution requires practically no developer interaction or knowledge of usability, it is based on analysis and automatic modifications.

In the evaluated scenario, the creation of a table that followed usability guidelines was much less complex, using the proposed solution with 43.83 % less complexity compared to the DataTables library, one of the reference libraries in the sector and 24.69 % less Compared to the AntDesign framework, one of the most powerful and most feature-rich in table representation. It can be concluded that the use of the proposed solution could significantly reduce the complexity of implementation in tables that want to include usability guidelines.

In addition, a study was carried out in which 42 users participated. An attempt was made to obtain an estimate of the usefulness of the guidelines. In the tests carried out, 7 of the 13 managed to significantly reduce the interaction time of the users in carrying out a task. On the

other hand, the application of other 5 / 13 guidelines could end up being negative for the usability of the table, making users take more time to complete tasks. The insights gathered from this evaluation regarding the potential effectiveness of the guidelines can assist developers in considering which usability guidelines to enable or disable when configuring the proposed solution. This will allow the proposal to be adapted more efficiently to each specific case. We think that the correct decision on deactivating or activating the guidelines should rest with the developer and be contrasted with real user activity data from the web page where the solution is being used.

The obtained results demonstrate that the proposed approach can significantly contribute to reducing development costs and improving the usability level of various types of HTML tables. It offers a more efficient and streamlined method for creating tables that adhere to different usability guidelines. In contrast, existing web development frameworks and libraries require manual implementation of these guidelines. By combining their understanding of website usability with user activity logs, developers can make informed decisions about which guidelines to incorporate, resulting in enhanced performance.

As part of our future research lines, we aim to undertake an exhaustive analysis to identify the most advantageous scenarios for implementing these usability guidelines, thereby maximizing user benefits. This endeavour will enable us to refine the heuristics that guides the application of these guidelines, tailoring them to specific user web activities. Moreover, we will explore novel aspects for enhancing tables and incorporate them into our proposal.

CRediT authorship contribution statement

Mario Díaz Ceñera: Conceptualization, Methodology, Writing – review & editing, Software, Data curation. **Julián Grigera:** Conceptualization, Methodology, Writing – review & editing. **Jordán Pascual Espada:** Conceptualization, Methodology, Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Akbar, M., Azizah, F. N., & Putri Saptawati, G. A. (2016). Integration of HTML tables in web pages. In *Proceedings of 2015 International Conference on Data and Software Engineering, ICODSE 2015*, 132–137. doi:10.1109/ICODSE.2015.7436985.

Alarcon, C., Medina, F., & Villarroel, R. (2014). Finding usability and communicability problems for transactional web applications. *IEEE Latin America Transactions, 12*(1), 23–28. https://doi.org/10.1109/TLA.2014.6716488

Alonso-Virgós, L., Espada, J. P., & Crespo, R. G. (2019). Analysing compliance and application of usability guidelines on efficient and understandable controls. *Computer Standards & Interfaces, 66*, Article 103349. https://doi.org/10.1016/J.CSI.2019.04.008

Alonso-Virgós, L., Thomaschewski, J., Espada, J. P., & Crespo, R. G. (2020). Test usability guidelines and follow conventions. Useful recommendations from web developers. *Computer Standards & Interfaces, 70*, Article 103423. https://doi.org/10.1016/J.CSI.2020.103423

Altin Gumussoy, C. (2016). Usability guideline for banking software design. *Computers in Human Behavior, 62*, 277–285. https://doi.org/10.1016/J.CHB.2016.04.001

Aşiroğlu, B., Mete, B. R., Yildiz, E., Nalçakan, Y., Sezen, A., Dağtekin, M., & Ensari, T. (2019). Automatic HTML code generation from mock-up images using machine learning techniques. In *2019 Scientific Meeting on Electrical-Electronics and Biomedical Engineering and Computer Science, EBBT 2019*. doi:10.1109/EBBT.2019.8741736.

Bader, F., Schön, E. M., & Thomaschewski, J. (2017). Heuristics considering UX and quality criteria for heuristics. *International Journal of Interactive Multimedia and Artificial Intelligence, 4*(6), 48. https://doi.org/10.9781/IJIMAI.2017.05.001

Barnum, C., Gillis, K., Dayton, D., & O'Connor, J. (2005). Making connections – Teaming up to connect users, developers, and usability experts. *IEEE International Professional Communication Conference, 108–114*. https://doi.org/10.1109/IPCC.2005.1494166

Borges, J. A., Morales, I., & Rodríguez, N. J. (1996). Guidelines for Designing Usable World Wide Web Pages. *CHI '96*, 277–278.

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., & Zhang, Y. (2008). WebTables: Exploring the power of tables on the web. *Proceedings of the VLDB Endowment, 1*(1), 538–549. https://doi.org/10.14778/1453856.1453916

Cappel, J. J., & Huang, Z. (2007). *A usability analysis of company Web sites*. https://www.researchgate.net/publication/252409801_A_usability_analysis_of_company_Web_sites.

Chen, H.-H., Tsai, S.-C., & Tsai, J.-H. (2000). *Mining tables from large scale HTML texts*. 166–172. doi:10.3115/990820.990845.

Coyle, A. (2017). *Design better data tables. The ingredients of a successful data table UI*. https://coyleandrew.medium.com/design-better-data-tables-4ecc99d23356.

Dibia, V., & Demiralp, Ç. (2019). Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications, 39*(5), 33–46. https://doi.org/10.1109/MCG.2019.2924636

Embley, D. W., Tao, C., & Liddle, S. W. (2005). Automating the extraction of data from HTML tables with unknown structure. *Data & Knowledge Engineering, 54*(1), 3–28. https://doi.org/10.1016/J.DATAK.2004.10.004

Espada, J. P., Díaz, V. G., Crespo, R. G., Martínez, O. S., G-Bustelo, B. C. P., & Lovelle, J. M. C. (2015). Using extended web technologies to develop Bluetooth multi-platform mobile applications for interact with smart things. *Information Fusion, 21*(1), 30–41. https://doi.org/10.1016/J.INFFUS.2013.04.008

Etemadi, V., Bushehrian, O., & Akbari, R. (2018). Association rule mining for finding usability problem patterns: A case study on StackOverflow. In *18th CSI International Symposium on Computer Science and Software Engineering, CSSE 2017, 2018-January*, 24–29. doi:10.1109/CSICSSE.2017.8320144.

Geng, R., Chen, M., & Tian, J. (2014). In-process usability problem classification, analysis and improvement. *Proceedings – International Conference on Quality Software*, 240–245. https://doi.org/10.1109/QSIC.2014.49

Golfarelli, M., & Rizzi, S. (2020). A model-driven approach to automate data visualization in big data analytics. *Information Visualization, 19*(1), 24–47. https://doi.org/10.1177/1473871619858933/ASSET/IMAGES/LARGE/10.1177_1473871619858933-FIG13.JPEG

Grigera, J., Garrido, A., Rivero, J. M., & Rossi, G. (2017). Automatic detection of usability smells in web applications. *International Journal of Human Computer Studies, 97*, 129–148. https://doi.org/10.1016/J.IJHCS.2016.09.009

Halbe, A., & Joshi, A. R. (2015). A Novel Approach to HTML Page Creation Using Neural Network. *Procedia Computer Science, 45*(C), 197–204. https://doi.org/10.1016/J.PROCS.2015.03.122

Hassini, A. (2023). *The Evolution of UI/UX Design: A Look Back and a Look Ahead*. https://bootcamp.uxdesign.cc/the-evolution-of-ui-ux-design-a-look-back-and-a-look-ahead-56b5abced825.

Hellmuth, M. (2019). *The Ultimate Guide to Designing Data Tables*. https://medium.com/design-with-figma/the-ultimate-guide-to-designing-data-tables-7db29713a85a.

Huijing, C. (2019). *Table Design Patterns On The Web*. https://www.smashingmagazine.com/2019/01/table-design-patterns-web/.

Hvannberg, E. T., Law, E. L. C., & Lárusdóttir, M. K. (2007). Heuristic evaluation: Comparing ways of finding and reporting usability problems. *Interacting with Computers, 19*(2), 225–240. https://doi.org/10.1016/J.INTCOM.2006.10.001

ISO 9126. (1991). *The Standard of Reference*. https://www.win.tue.nl/~wstomv/edu/2ip30/references/9126ref.html.

ISO 9241-11:2018. (2018). *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en.

Juristo, N., Moreno, A. M., & Sanchez-Segura, M. I. (2007). Analysing the impact of usability on software design. *Journal of Systems and Software, 80*(9), 1506–1516. https://doi.org/10.1016/J.JSS.2007.01.006

Laubheimer, P. (2022). *Data Tables: Four Major User Tasks*. https://www.nngroup.com/articles/data-tables/.

Mariage, C., Vanderdonckt, J., & Pribeanu, C. (2006). *State of the Art of Web Usability Guidelines*.

Marriott, K., Moulder, P., & Hurst, N. (2013). HTML Automatic Table Layout. *ACM Transactions on the Web (TWEB), 7*(1). https://doi.org/10.1145/2435215.2435219

Mgheder, M. A., & Ridley, M. J. (2008). Automatic generation of web user interfaces in PHP using database metadata. In *Proceedings – 3rd International Conference on Internet and Web Applications and Services, ICIW 2008*, 426–430. doi:10.1109/ICIW.2008.100.

Nielsen, J. (1994). *Enhancing the Explanatory Power of Usability Heuristics*.

Okamoto, S., Dascalu, S., & Egbert, D. (2006). Web Interface Development Environment (WIDE): Software tool for automatic generation of Web application interfaces. In *2006 World Automation Congress, WAC'06*. https://doi.org/10.1109/WAC.2006.376042.

Pawade, D., Sakhapara, A., Parab, S., Raikar, D., Bhojane, R., & Mamania, H. (2018). Automatic HTML code generation from graphical user interface image. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 – Proceedings*, 749–753. doi:10.1109/RTEICT42901.2018.9012284.

Rafalowicz, A., & Nguyen, H. (2017). *Designing Tables for Reusability*. https://uxdesign.cc/designing-tables-for-reusability-490a3760533.

Redfield, K. (2021). *10 Tips for Creating Beautiful Website Tables*. https://blog.duda.co/create-beautiful-website-tables.

Ruiz-Rodríguez, R. (2006). An auxiliary tool for usability and design guidelines validation of web sites. In *Proceedings – 15th International Conference on Computing, CIC 2006*, 304–308. doi:10.1109/CIC.2006.21.

Shepherd, A. (2023). *Top 5 Table UI Design Examples and Templates*. https://mockitt.wondershare.com/ui-ux-design/table-ui-design.html.

Shestopalov, S. (2019). *How to design complex web tables*. https://medium.com/design-bridges/complex-tables-356826d11861.

Spyridakis, J. (2000). Guidelines for authoring comprehensible web pagesand evaluating their success. *Technical Communication (University of Washington), 47*, 359–382.

Sukamto, R. A., Wibisono, Y., & Agitya, D. G. (2020). Enhancing the User Experience of Portal Website using User-Centered Design Method. In *2020 6th International Conference on Science in Information Technology: Embracing Industry 4.0: Towards Innovation in Disaster Management, ICSITech 2020*, 171–175. doi:10.1109/ICSITECH49800.2020.9392044.

Tajima, K., & Ohnishi, K. (2008). Browsing large html tables on small screens. In *UIST 2008 – Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, 259–268. doi:10.1145/1449715.1449758.

Vargas, A., Weffers, H., & Da Rocha, H. V. (2011). Discovering and analyzing patterns of usage to detect usability problems in web applications. *International Conference on Intelligent Systems Design and Applications, ISDA*, 575–580. https://doi.org/10.1109/ISDA.2011.6121717

W3Techs. (2023). *Usage Statistics and Market Share of CSS Frameworks for Websites*. Usage Statistics and Market Share of CSS Frameworks for Websites. https://w3techs.com/technologies/overview/css_framework.