Taboo: Chord Sheet Editor and Manager Web Application

Ervin Erőss*, Adorján-Ferenc Péter*, Balázs Sebestyén†, Tibor Fazakas†, Zoltán Szabó† and Csaba Sulyok*
*Faculty of Mathematics and Computer Science, Babeș-Bolyai University
RO-400084 Cluj-Napoca, Romania
†Codespring
RO-400664 Cluj-Napoca, Romania
erosservin@gmail.com; peter.ador@yahoo.com; sebestyen.balazs@codespring.ro;
fazakas.tibor@codespring.ro; szabozoltanbors@gmail.com; csaba.sulyok@gmail.com

***Abstract*—The aim of the Taboo project is to provide a user-friendly interface for creating and managing guitar chord sheets. The target audience is set as musicians with an interest in browsing guitar tablatures and creating their own collection of chord sheets.**

The essential innovation of the application lies in the interactive interface for creating and editing these sheets, which are described by a Domain Specific Language (DSL). This markup language is interpreted by a rendering engine, facilitating the responsive appearance of the chord sheets on devices with different resolutions.

This article presents the Taboo project, including the requirements of the application, its architecture and the details of the implementation. It describes the technologies and tools used during the development process and it demonstrates the usage of the software.

I. Introduction

There are several guitar tablature management systems[1] available for musicians, however one crucial common issue of these websites is the storage of data in plain text format. Therefore chords are usually positioned above the lyrics with whitespace characters, which makes the sheets harder to read on smaller screens because the lines are incorrectly broken. Furthermore, they may present other deficiencies, such as the lack of password recovery, the difficulty of editing sheets or the insecure user data processing.

Chord sheets are lyrics with guitar chords positioned over particular words, indicating where and when a chord should be strummed. The application defines a domain specific language (DSL) (see Section III-D), which helps keep the accuracy of chord sheets.

The motivation of the Taboo project is to improve the shortcomings of already existing similar systems. Its purpose is to provide an interactive interface to create chord sheets having a responsive appearance. Thus the application provides a good user experience when only mobile devices are available. Further auxiliary features are also taken into consideration, such as downloading chord sheets in PDF format (see Section III-B), transposing chords or creating collections of chord sheets. Transposing chord sheets involves changing the key of the song, shifting all notes/chords with the same interval as the difference in keys. This preserves the harmonic nature of the song, while involving different, potentially easier hand positions.

The requirements and the architecture of the project, along with the data models are presented in Section II. Section III explains the implementation details of the application. The used technologies and tools are presented in Section IV. Finally, Section V and VI present the usage of the application with plans for further development.

[1]Example website: http://gitartab.hu

II. The Taboo project

The following section presents the main functionalities, the architecture and the domain entities of the application.

A. Requirements

Users of the application may take on different roles, such as guest, authenticated user, moderator and administrator.

The guest user can access the least amount of functionalities, some of the more important ones include:

- listing the uploaded chord sheets per page;
- transposing and downloading chord sheets in PDF format;
- searching for chord sheets.

Compared to the user with guest privileges, additional functionalities are available for the registered and authenticated user, such as:

- creating chord sheets;
- modifying and deleting their own sheets;
- reporting chord sheets created by another user;
- organizing chord sheets into collections;
- creating, deleting and downloading collections in PDF format;
- sharing collections via public link;
- viewing and modifying profile data.

Besides the previously enumerated functionalities, the following are available for the moderator:

- deleting any chord sheet;
- deleting a flag from a reported chord sheet, which was created by another user.

All the enumerated functionalities are available for the user with administrator privilege, along with the following:

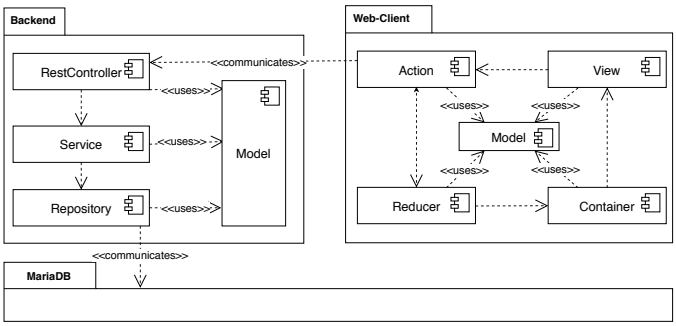

Fig. 1. Architecture of the system with the layers and the relationships between them

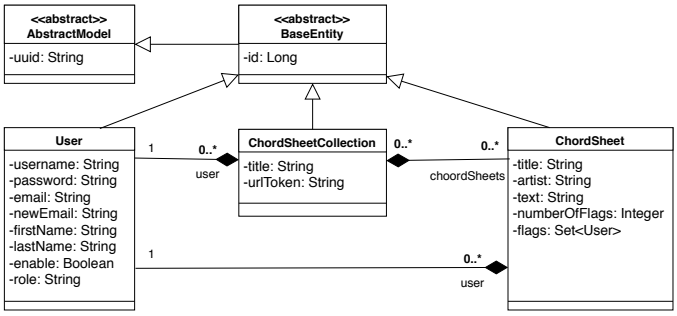

Fig. 2. Class diagram representing server side entities and the relationships between them

- upgrading a simple user to moderator;
- downgrade a moderator to simple user.

B. Architecture

The Taboo software system contains three major components: a Java server built on the Spring framework, a web client using React, and a MariaDB database management system (see Figure 1). The application follows the conventions set forth by multilayer architectures [1]. In the following, the dedicated server-side components are detailed.

The `Model` contains the main entities of the software, which are mapped to the database by using the Hibernate object relational mapping framework. The information sharing between different layers is also realized using these models. The `Repository` component is responsible for the different database manipulation operations, such as creating, reading, updating and deleting entities. The application uses the MariaDB [2] relational database management system for storing data. The business logic of the application can be reached through the `Service` component. It uses the Repository layer to build data transmitted to the higher layers. The `RestController` performs the routing of HTTP requests obeying a RESTful specification, thus opening up the server to any potential clients.

The responsibilities of the web client module are building and managing the de facto presentation layer of the application. Its `View` component stores the structure of the displayed graphical elements. The `Action` component facilitates the HTTP communication with the server; it uses an event-based mechanism to send requests and react to incoming responses. The responsibilities of the `Reducer` are to catch events and the received data to forward. Finally, the `Container` component updates the application state with data transmitted by `Action`, thereby displaying the actual content.

C. Data models

The classes representing server-side entities are Plain Old Java Objects (POJO) boasting only private fields, a public no-argument constructor and getter/setter methods. They are shared across the different components of the server. The relationships between the data models can be seen in Figure 2.

The `AbstractModel` class has a single property, which provides a universally unique identifier for the entities. The `BaseEntity` extends this, which also contains a single property, namely the `id`, which is equal to the primary key in the relational database. These are extended by the `User`, the `ChordSheet` and the `Collection` classes.

The entities are all mapped to the relational database by the implementation of the JPA (Java Persistence API) specification provided by the Spring framework. The entity classes are decorated with the appropriate annotations dictated by the specification; restrictions and relationship schemes are specified here for the underlying tables and their columns.

III. Implementation Details

This section illustrates the operating principles of major features of the project using figures. The server side component of the Taboo application is responsible, among others, for PDF generation for downloading chord sheets and collections; rendering and automatic sending of e-mails. Among significant front-end features are the chord sheet editor or the communication with the server. In the following, the implementation details of such scenarios are presented.

A. RESTFul web services

Taboo is a Single Page Application, which implies a single initial HTML resource is served to the browser, handing over further content change rights to the client-side code blocks. When a user event occurs, these blocks send asynchronous requests to the server to fetch/push any relevant data. With any fetched data being processed and inserted into the right template, it is ensured that up-to-date data is presented without refreshing the page. This architecture loosens the dependency between client and server, allowing either side to changed without a significant impact on the other. It also decreases network traffic, since only necessary data is refreshed.

The Taboo project contains an API layer, which receives and processes HTTP requests sent by the clients, strictly abiding to RESTful conventions. It uses the Spring Web extension framework to map incoming HTTP requests to controller methods using the `@RequestMapping` annotation.

For example, a `POST` request sent to the `/api/chordSheets` endpoint is routed to the

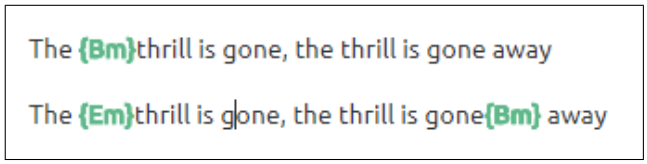

Fig. 3. The format of the DSL chord sheet for storage and handling.

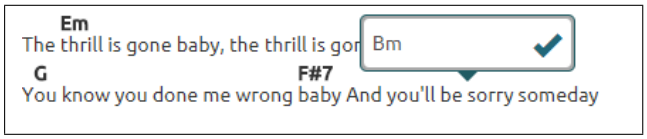

Fig. 4. Pop-up for interactive chord operations

`addNewSheet()` method, which expects a `ChordSheet` object as parameter; the `@RequestBody` annotation signals that the request body should be deserialized into the parameter. The information is propagated forward through the service and repository layers, finally begin inserted into the database.

The requests received on the `/api/chordSheets/{id}` endpoint perform different tasks according to the request type: `GET` returns, `PUT` updates and `DELETE` deletes the chord sheet with the corresponding ID.

All request bodies, parameters and payloads use the JSON format, with the serialization and deserialization being resolved by the Spring Web framework. Responses encapsulate HTTP status codes set according to REST conventions, reflecting the success/failure status of execution.

B. PDF generation

To make the favorite songs of a user available even in an offline setting, the Taboo application provides the opportunity of downloading chord sheets and collections in a generated printer-friendly PDF format. Users may also create hand-picked music booklets by printing collections.

The PDF generation is implemented on the server side with the help of the Flying Saucer library, which expects an input string with XHTML content for rendering. To produce the mentioned parameter, the chord sheet described by a DSL needs to be processed. In this DSL, the chords are represented by a chord name placed between curly brackets (see Figure 3), marking their right position in the lyrics. The processing of the lyrics entails placing the chords over the text using the appropriate CSS classes. After the content generation, the PDF is output by using the iText library. Since the server does not need the generated files it stores the rendered content in the memory only while they are used.

C. E-mail sending

One of the missing functionalities of similar web applications is resetting forgotten passwords. In order to offer this feature, it needs to introduce a registration involving two steps. This means that after providing the personal data, the server generates a unique identifier, which is included as a link parameter in the confirmation message sent to the e-mail address given by the user. By clicking the link, the user proves that the e-mail address belongs to them, so they get access to the system.

When the user starts the password reset process, the server sends a link to the given e-mail address, which redirects to a form where the user can set a new password.

Similarly to the registration and the forgotten password resetting procedures, the registered user has the opportunity to change the e-mail address which was given at the registration. With the help of a verification e-mail it is ensured that the address is valid.

For the functionalities described above, e-mail sending is necessary. The Taboo application uses the Mailgun services in order to realize this functionality. The generation of the mail content includes filling the FTL (FreeMarker Template Language) templates with data (username, link, date). After generating the e-mail, it is transmitted to the Mailgun server, which delivers the it to the user.

D. Chord sheet editor

One of the main components of the Taboo project is the chord sheet editor, which gives the possibility to create and maintain the content of the website. The editor interface is implemented with a simple and intuitive editing process in mind for the user. The editor supports two ways of entering chords: either in an interactive (using a mouse) or in a raw format.

The editor uses the DSL to separate the chords from the lyrics, and the chord sheets are also stored in the database in this format. As can be seen in Figure 3, the interpreter of the editor recognizes the characters inside the curly brackets as a chord. Therefore, displaying and editing the sheet on the website is also possible thanks to the DSL. Displaying the chord sheet on the user interface, the rendering unit parses the text row by row, replacing chord syntaxes with specific HTML tags. These elements are formatted and placed above the lyrics using CSS. Since the chords have a specific location relative to the lyrics, they remain well positioned even on smaller screens where word wrapping occurs.

The chord sheet editor is implemented using Draft.js [3], which is an open source rich text editor framework developed by Facebook. It allows annotating ranges of text with metadata. As can be seen on the Figure 3 chords are highlighted, since the editor interprets the attached metadata and adds a special React component (Decorator) to the corresponding range.

The application includes an interactive chord input GUI, in order to avoid forcing the user to learn the syntax of the DSL and to follow its evolution. When the user clicks on any position in the lyrics, a chord input window appears. The window (see Figure 4) contains only an input field to enter the chord, and a button to finalize the operation. Finding the right method for positioning and moving the window presents a challenge, because the accurate position of the window over the cursor has to be determined for various screen resolutions.

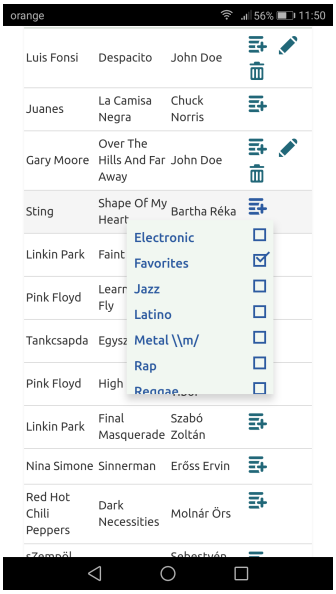

Fig. 5. The list of chord sheets and the drop-down menu to add them to collections

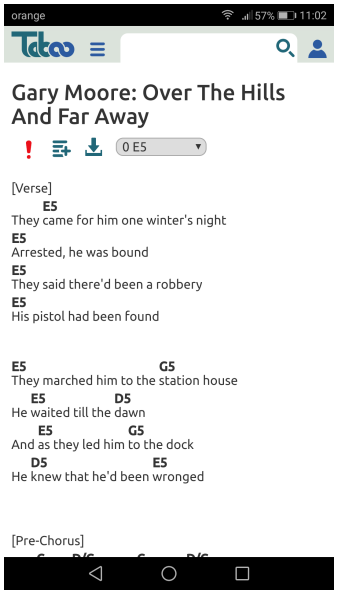

Fig. 6. The details of a chord sheet with the action buttons

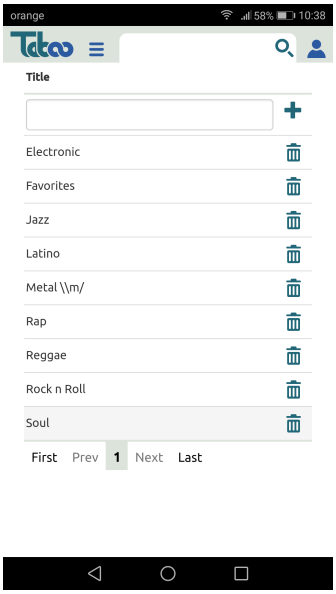

Fig. 7. Component to manage chord sheet collections

The editor supports the interactive insertion, modification as well as deletion of the chords. The editor verifies the content of the input field before each insertion. It validates the leading characters and ensures they represent prefixes for possible chords. The chords are easily editable with the interactive mode, because the position of the cursor is checked after every click, and if an entity is found, its content is loaded into the input field of the window. When editing an existing chord sheet, it is loaded into the editor, which parses and builds the `ContentState` in order to become editable again.

E. Communication with the server

The client communicates with the server through a REST API with JSON formatted requests and responses. The client side of the application is implemented using the React JavaScript framework. This is extended with the Redux[4] state manager in order to make the application easier to maintain and to facilitate further development. Thus the data flow of the application is strictly unidirectional, so the state becomes predictable.

The application has a store which contains its entire state. The objects in this store are read-only, and the only way to modify them is by dispatching actions. These are executed by `Reducers`, which are pure functions; they do not depend on any external state, such as a database, and always return the same result if the same arguments are given. The requests to the server are executed using the Redux Thunk middleware, which enables asynchronous execution.

IV. Technologies & Tools

This section describes the technologies and tools used to build, run, deploy and ensure quality of the Taboo project.

The server side of the project is developed in the Java programming language, with the architectural layers implemented using the Spring [5] framework. Besides Spring modules, other employed third-party libraries include the Apache FreeMarker[6] template engine for e-mail content generation and the Flying Saucer[7] and iText[8] libraries for PDF file generation. The build process is executed with the Gradle[9] build and dependency management tool.

The web client side uses the TypeScript[10] programming language, the React [11] framework and the Redux [4] state management library. Additionally, the Draft.js[3] framework is used to implement the chord sheet editor and the responsive design is built using Bootstrap[12]. It uses the Yarn package manager to control the dependencies and Gulp[13] to transpile TypeScript code into browser-compatible JavaScript.

Quality assurance of the application in its development is guaranteed by static code analyzers, which enforce agreed upon code conventions being respected in the source code. The project uses Checkstyle[14] and FindBugs[15] for Java[] code as well as TSLint[16] for TypeScript.

Git[17] is used for version control, while GitLab takes care of centralized repository management[18], project management as well as continuous integration[19]. The latter involves pipelines begin automatically executes at every push to the repository. The jobs in the pipeline include the build process, running the code analyzers and the tests. Additionally, in case of a push or merge to the main development branch, a continuous deployment pipeline is triggered: the application is automatically deployed to a staging server with the help of Docker[20] and docker-compose.

V. Usage of the Taboo Application

This section shows the main functionalities of the Taboo web application using screenshots and descriptions.

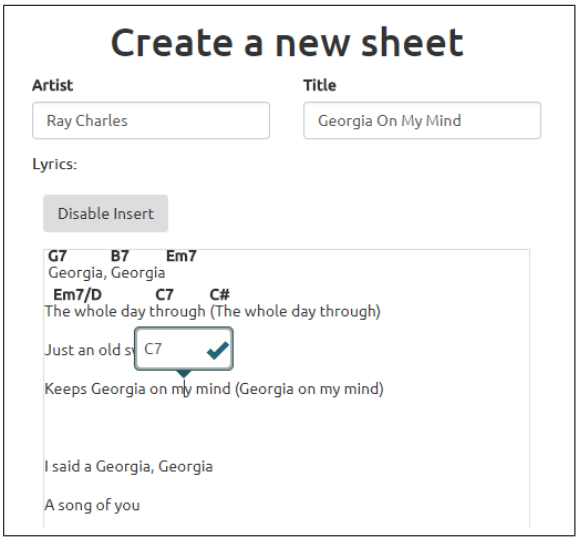

Fig. 8. The chord sheet editor

The user is greeted by a landing page with the navigation bar on top, providing the search field and an icon, which opens a drop-down menu with user related functionalities. For a guest user, the drop-down menu contains both the standard login form as well as the Facebook API assisted login; links are also available to reset a forgotten password and to register for a new account. An authenticated user is instead presented with navigation options to their own chord sheets and collections, to change their profile data or to log out of the application.

The main body area of the application may show a list of chord sheets in many different scenarios: as a result of viewing the proprietary sheets of an authenticated user, listing the content of a collection or presenting search results. Figure 5 shows a sheet listing example; each row represents a chord sheet by showing its artist, title and uploader. The authenticated user is allowed more functionalities, such as sorting chord sheets into collections or editing and deleting own chord sheets. The administrators and moderators have permission to delete chord sheets belonging to other users as well.

Clicking on a row prompts the respective chord sheet to be rendered (see Figure 6). The usability on smaller devices is a main concern of the application, therefore the interface design layout shows simplicity and transparency. The chord sheets details view offers further functionalities: transposing chords, downloading, editing, deleting, reporting or sorting into collections.

The chord sheet collection view (see Figure 7) allows users to create, view or delete collections. Using the drop-down menu seen in Figure 5, the users can order chord sheets into collections.

Figure 8 shows the chord sheet editor view. While typing lyrics into the editor, the user can add chords which appear immediately above the text, and are managed in the background using the DSL.

VI. Conclusion and Future Work

The Taboo web application reaches its aim by implementing a user-friendly and intuitive interface for creating and managing chord sheets in a precise manner. It successfully involves an interactive chord sheet editor and eliminates the problem of broken lines with the help of a simple domain specific language. Thanks to the minimalistic responsive design, the application remains enjoyable on smaller screens as well.

Additional features successfully encapsulated within the application include: collection management and sharing by URL, transposing and reporting chord sheets, downloading collections and chord sheets in PDF format, user management, etc.

The planning, design and development phase of the project has brought further improvement ideas to light, which include:

- internationalization and multi-language support for the application;
- providing an interface for the users to be able to make suggestions referring to chord sheets;
- graphically displaying chord fingering above the chords in the sheets;
- integrating a chord sheet generator capable of recognizing chords from an audio stream or YouTube video (such as Chordify) into the system.

References

[1] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002.
[2] "MariaDB Official website." [Online]. Available: https://mariadb.org/
[3] "Draft.js Official website." [Online]. Available: https://draftjs.org/
[4] "Redux Official website." [Online]. Available: https://redux.js.org/
[5] J. Rod, H. Juergen, D. Keith, S. Colin, H. Rob, R. Thomas, A. Alef, D. Darren, K. Dmitriy, P. Mark, T. Thierry, V. Erwin, T. Portia, H. Ben, C. Adrian, L. John, L. Costin, F. Mark, B. Sam, L. Ramnivas, P. Arjen, B. Chris, A. Tareq, C. Andy, S. Dave, G. Oliver, S. Rossen, W. Phillip, W. Rob, C. Brian, N. Stephane, and D. Sebastien, "Spring Framework Reference Documentation." [Online]. Available: https://docs.spring.io/spring/docs/4.3.9.RELEASE/spring-framework-reference/htmlsingle
[6] "FreeMarker Java Template Engine Official website." [Online]. Available: https://freemarker.apache.org/
[7] "The Flying Saucer Official website." [Online]. Available: https://flyingsaucerproject.github.io/flyingsaucer/r8/guide/users-guide-R8.html
[8] "iText Official website." [Online]. Available: https://developers.itextpdf.com/apis
[9] "Gradle Official website." [Online]. Available: https://gradle.org/
[10] "TypeScript Official website." [Online]. Available: https://www.typescriptlang.org/
[11] React Official website. [Online]. Available: http://www.reactjs.org/
[12] "Bootstrap Official website." [Online]. Available: https://getbootstrap.com/
[13] "Gulp.js Official website." [Online]. Available: https://gulpjs.com/
[14] "Checkstyle Official website." [Online]. Available: http://checkstyle.sourceforge.net/
[15] "FindBugs™ Official website." [Online]. Available: http://findbugs.sourceforge.net/
[16] "TSLint Official website." [Online]. Available: https://palantir.github.io/tslint/
[17] "Git Official website." [Online]. Available: https://git-scm.com/
[18] V. Driessen, "A successful Git branching model," 2010. [Online]. Available: http://nvie.com/posts/a-successful-git-branching-model/
[19] "GitLab Continuous Integration." [Online]. Available: https://docs.gitlab.com/ee/ci/
[20] "Docker Official website." [Online]. Available: https://www.docker.com/