LIVE LAB v1.0

A Smarter Approach to Musical Practice

Live Lab

A Smarter Approach to Musical Practice

Authorship and Ownership:
This document has been prepared by Imanol Arreseygor.

Document Version:
Version: 1.0
Creation Date: October 27, 2025
Language: English

Intended for technical documentation and internal development use.

Scope of Application:
This module is intended for developers, instructional designers, and collaborators involved in the musical analysis system.

Contact:
Email: imanol.arreseygor@gmail.com

Table of Contents

Introduction

In recent years, following the expansion of applications, digital tools have emerged that enable a new way to develop skills when playing a musical instrument. There has been a shift from classical and academic learning to self-taught learning, especially in more modern styles, culminating in the use of these programs.

Advantages and challenges of digital tools

While the classical method can feel monotonous due to the rigidity of its structure and the limited repertoire, self-taught learning can lead to frustration—especially in the early stages if no tangible results are achieved. New digital tools, however, offer the following advantages:

- ✔ Personalize learning according to the user's level,
- ✔ Gamify practice to make it more enjoyable,
- ✔ Provide progress tracking, which boosts motivation,
- ✔ Often include modern repertoires,
- ✔ Are more affordable and accessible to a wide audience.

Despite all these new possibilities, there are disadvantages.

- ✘ **Limited internal structure**. Lessons are often presented as isolated units or with little connection between them.
- ✘ **Limited personalization**. Although they adapt to the user's level, they do not allow for freely combining theory, technique, and practice according to individual interests.
- ✘ **Rigid progression**. Learning paths are usually linear and not easily adaptable, which reduces long-term motivation.

Personal needs and holistic approach

In my personal case, my interest in musical practice branches out in several directions: from live pad performance in fun and engaging styles like hip hop, break, jazz — in true finger drumming fashion — to the development of skills in music creation and production, including harmonic and rhythmic theory, expressive sensitivity, knowledge of instruments, and improvisational ability. The goal is to address all aspects of musical creation and performance, which presents the challenge of planning and combining lessons to achieve positive synergy and maximize the effectiveness of practice.

A tool for synergy in musical learning

This is where the tool described in this document comes in—a program whose main functions are to organize the elements involved in music and interrelate them, creating a more complex and intelligent base system than working with disconnected elements. This approach offers:

STRUCTURED ORGANIZATION	Classify items into categories and subcategories: • Provides a clear and accessible conceptual framework. • Facilitates understanding and selection of areas to practice.
PANORAMIC VIEW	• Allows you to identify at a glance the set of skills and concepts available • Helps identify areas of interest or improvement.
SYNERGY BETWEEN LESSONS	By establishing relationships between elements, it's possible to combine lessons that reinforce each other. This: • Optimizes the learning process • Balances skill development.

Ultimately, this tool allows users to build personalized and effective practice routines, working on different aspects of musical practice in a balanced way.

Final considerations

It is important to note that this program a draft or prototype, not a functional application. It is presented as a conceptual proposal to illustrate how practice routines can be organized more intelligently, without ignoring their limitations. It is not intended as the starting point for a fully developed application, but rather as a demonstration of the potential for integrating this approach into more advanced educational solutions. While I am not a professional developer, I have the ability to conceptualize and design systems that address real needs. This prototype aims to highlight a gap that, with proper development, could be effectively addressed in the future.

Objectives of the dossier

1. **Present the program's conceptual framework:** inspiration, objectives, and pedagogical foundations.
2. **Describe how it works:** Elements, interface, internal logic, and usage modes.
3. **Present conclusions, identify areas for improvement, and propose future directions.**


MPC LIVE II
Hip-hop-Kit-CAPSUN Pumps Kit
Track 01
Sequence 04
unfolded
COPY
DELETE
UNDO
REDO
ON/OFF
MAIN

Conceptual Framework

2.1. Ideas Behind the Creation of the Program

The development of this program stems from a personal need to understand what effective practice truly involves, how to plan a routine accordingly, and a reflection on how a tool might support that process. Although the program is still in its prototypical development phase, a tool of this kind could help other instrumentalists—especially self-taught musicians or those still in training—to better plan and optimize their practice. Whether it's to showcase the diversity of practice options to those who only know one method, or to bring structure to more scattered approaches (as is often my own case), the tool aims to be useful.

Conceptual foundations and philosophy

The philosophy behind the program is based on the need for a system that facilitates musical practice from a holistic perspective, combining personal skills, technical abilities, theoretical knowledge, and a deeper understanding of the components that make up a lesson. The aim is to help users make informed, conscious and efficient decisions about what and how to practice.

DIVERSITY

The multiplicity of styles, techniques, and learning methods reflects the richness of music and can be overwhelming. An effective tool must consider this diversity and help navigate it.

TOTALITY AND BALANCE

Instrumental skill is a system of interdependent elements. Musical practice is the result of an interconnected system of technical, expressive, cognitive, and sensory elements. It is not just about improving an isolated skill, but about developing a balanced set of competencies. The tool aims to help seek balanced practice without falling into dispersion.

AWARENESS OF DEVELOPMENT

Being aware of the elements involved facilitates the identification of goals, the understanding of practice, and the monitoring of progress, which is essential for maintaining motivation.

Inspiration and pedagogical model

The program is inspired by educational apps like Melodics or Synthesia, which promote instrument practice in an enjoyable and motivating way. However, it's not intended to replace them, but rather to complement them, helping to organize the practice routine:

✔ By presenting **musical elements in an organized and hierarchical** way.

✔ By **linking** these elements to form a more intelligent system.

✔ By making the **internal structure of lessons** visible (what elements they include).

Effective practice requires a **comprehensive, systemic, balanced, and customizable approach**.
This program aims to guide without limiting and to accompany without directing.

✔ By allowing for **customized combinations** of elements based on specific goals.

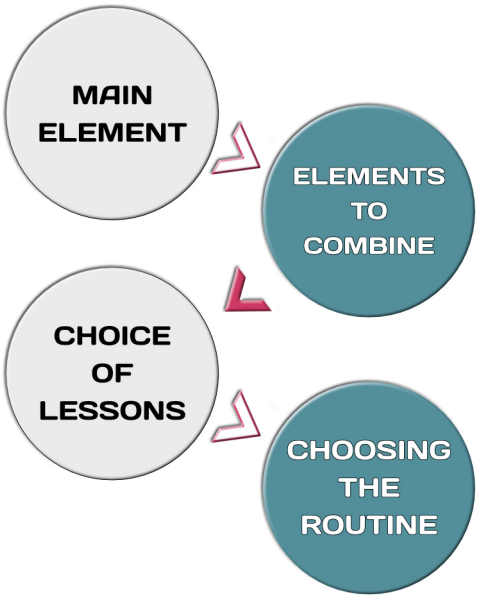
✔ By incorporating features that foster a **more conscious, reflective, and strategic practice**.

Identified issues and added value

Current instrumental learning apps have revolutionized musical practice, but they often lack advanced systems for creating personalized routines. The proposed program:

Solve problems like	It provides a series of values
• Difficulty in gaining a **global view** of the elements that make up each lesson. • Inability to select lessons based on **specific combinations of elements**. • Lack of tools to **organize, filter, and adapt** practice in an advanced and personalized way.	• An **intelligent organization** of content by musical and educational elements. • The ability to **choose a core element** and find lessons that address it along with complementary aspects. • A focus on building **customized and strategic practice routines**. • A tool that promotes **awareness, clarity, and autonomy** in the practice process.

An approach based on element combination



The core of the program lies in its **combinatorial capability**: the user selects a main element they wish to practice, the system shows related elements and, from there, the user chooses meaningful combinations. The program then displays lessons grouped according to these combinations. This strategy enables:

- Enhancing the practice of the desired element without losing sight of the **context** in which it appears.
- Supporting a **rich progression** that develops multiple skills simultaneously.
- **Saving time** by avoiding fragmented and unstructured practice through lessons that create synergy between elements.

2.2. Fundamentals of Effective Musical Practice

This section presents various methodologies supported by research and pedagogical experience that facilitate the effective development of musical practice. These strategies have been considered in the design of the program, adapting them to its current and potential capabilities.

Theoretical knowledge

Knowledge of the underlying musical theory is a fundamental foundation that facilitates the understanding of what is being practiced. Understanding theoretical concepts allows the user to better contextualize and assimilate each exercise or lesson. This means not just practicing, but knowing **what** is being practiced and **why**.

Although the current version of the program does not incorporate the theory associated with each musical element, the way the elements are organized and their selection allows the user to intuitively approach the underlying theory through the program's own structure.

Goal planning

Setting clear goals before practice is key to musical development. Just like theoretical knowledge helps to understand *what* is being worked on, goal planning helps define **where one is headed**. This provides clarity about the purpose of practice and helps focus efforts. The program implements this function in two ways:

Element Selection	Goal List
The program requires the user to choose a primary element as the basis for the session. This acts as a main objective. The selection of additional, complementary elements helps refine and define more specific goals.	One of the available lists is dedicated to practice goals. Although only a few are implemented at this stage, it opens the door for integrating a wide variety of goals associated with different elements of the program in the future.

Time Division

Although this feature is not yet implemented in the current version, the way routines are displayed in blocks helps estimate and allocate time for each practice section. This allows for better time management during practice sessions.

Structure of the practice

Not all lessons serve the same purpose in a practice routine. Here, three main types of lesson functions are introduced, ranging from simple to more complex:

WARM UP

Aimed at physically preparing the body for practice, improving hand mobility, and preventing injuries, these exercises are typically simple and require less effort, helping to gradually warm up. The program includes a dedicated section for this purpose.

TECHNICAL AND FUNDAMENTAL PRACTICE

Before moving on to more comprehensive exercises, it is important to dedicate time to mastering the fundamentals and techniques to be integrated into one's musical skillset. This type of practice is implemented through the following program sections:

Musical Dimensions	Techniques	Fundamentals
Determines which fundamental aspects of the practice will be worked on.	It refers to the musical techniques themselves.	`Listbox` for lessons that serve to practice specific fundamentals or techniques.

REPERTOIRE REHEARSAL

Dedicated to practicing complete pieces or complex exercises. Currently, this feature is included in the *Development* section, which covers both advanced exercises and complete arrangements. It may be further subdivided in future versions for greater specialization.

Routines: Chunking and Blocking

This is the core of the program. Choosing lessons is important, but combining them effectively is what maximizes progress. Two complementary types of practice routines are considered:

CHUNKING	BLOCKING
• It consists of breaking down tasks into simpler elements and practicing from the basics to the complex. • The program allows you to create routines of this type by selecting lessons with common elements for the warm-up, fundamentals, and rehearsal phases, facilitating logical progression.	• It involves practicing in separate blocks, each focusing on a different aspect, to achieve a balance between various skills. • The user can create blocks with various combinations of elements, thus working on different areas in a single session.

While **chunking** focuses on depth—using different types of lessons to develop a single element—**blocking** emphasizes breadth—using each block to work on different aspects. These approaches are not exclusive; rather, they complement each other and can be used together.

2.3. Key Concepts and Program Categories

This section describes the main musical, learning, and objective components that make up the program's structure. The organization and selection of these elements allows the user to design personalized practice routines, facilitating the comprehensive understanding and development of their musical skills.

Musical elements

This section compiles various musical aspects considered in the program, grouped into the categories of *skills*, *musical dimensions and sub-dimensions*, *techniques*, and *modes*, along with a basic overview of the relationships established between them. While not all possible elements are included, this set is sufficient to form a foundational structure that gives the program its meaning.

SKILLS

Performer's physical and mental capabilities when playing the instrument.

Agility
The ability to move hands and fingers quickly and accurately in order to play complex sections with precision.

Independence
Ability to execute different patterns with each hand or finger.

Endurance (or Stamina)
The musician's ability to sustain technical and expressive performance over extended periods, both physically and mentally.

Timing
Precision in rhythmic execution and coordination, achieving a sense of unity and groove in the performance.

Sensitivity
The ability to control the intensity and articulation of each note enhances expressive capability. This includes both the interpretation of emotional content and the flexibility to shift between different expressive styles.

Note: Although this skill is included, it currently has no associated lessons.

MUSICAL DIMENSIONS

They encompass the main characteristics of a musical piece:

Rhythm
The horizontal component of music, related to the timing between sounds. Rhythm is responsible for the sensation of movement through a basic pulse.

- **Signature**: Structure of the measure.
- **Note Values**: Duration of notes.
- **Tuplets**: Modify the length of a given note.

Harmony
The vertical component that reflects the relationship between pitches or tonal distances in a piece, resulting in chords.

- **Scales**: The base set of notes used to create harmonies and melodies.
- **Chords**: Groupings of notes played simultaneously.
- **Chord Progressions**: Sequences of chords that establish harmonic movement.

Structure
Way a musical piece is organized over time.

- **Sections**: Each of the blocks that make up a musical arrangement.
- **Forms**: Organization of the sections of a musical piece.

TECHNIQUES

Specific physical or interpretive methods used to produce sound or enhance musical expression.

Rudiments
Basic percussion patterns.

Fills & Rolls
Techniques used to create transitions and continuous, rapid note sequences, especially in percussion.

Ornaments
Musical ornaments that enrich expression.

Arpeggios
Successive execution of the notes of a chord.

Articulations
Different ways of playing and tying notes.

Tremolo y Vibrato
Periodic variations in amplitude or frequency in a note.

Polyrhythm
Superposition of rhythms with different durations.

Swing-Shuffle
Rhythmic technique that generates a characteristic groove.

MODES

Figura 1: Interfaz gráfica (GUI) del programa

They influence how the content is approached. Some of these modes of performance are compiled here:

Free Mode
As indicated, a way of playing without restrictions.

Linear Mode
A mode in which only one pad is pressed at a time, sometimes using only the index finger of each hand.

Layering
It consists of superimposing several sound layers to create a fuller, richer sound with greater depth as the instrument is played.

Cue Point Drumming
Triggering loops or musical sections from the pads.

One Handed Drumming
One-handed rhythmic playing, allowing for combination with other instruments.

Layout
Custom arrangement of sounds on the pads.

Relationships between elements

The program's system not only considers these elements in isolation, but also establishes relationships between them, allowing for a comprehensive and coherent vision of musical practice. For example, all the skills are interconnected and, except for timing (mainly related to rhythm), the rest are linked to the three main musical dimensions.

Lesson elements

This section describes the elements associated with the lessons. It is important to note that the term *lesson* is used in three different senses:

THE LESSON ITSELF

That is, each specific practice.

LESSONS CHARACTERISTICS

Refers to the inherent elements of the lessons; that is, they describe the features of the lessons rather than what is developed by practicing them. These are only used for selecting the main lesson element:

- **Interface**: Indicates the instrument used and displays the lessons specifically developed for its practice. Currently only _keys_ and _pads_ are considered.
- **Musical Genre**: The musical style of the lesson. In the current version, the genres of the lessons are analyzed and shown in a list.
- **Level/Grade**: Specifies the difficulty of the lesson. Divided into 3 categories, based on the *Melodics* system: easy (levels 1–5), medium (levels 6–10), hard (11–15).

LESSON FUNCTION IN THE ROUTINE

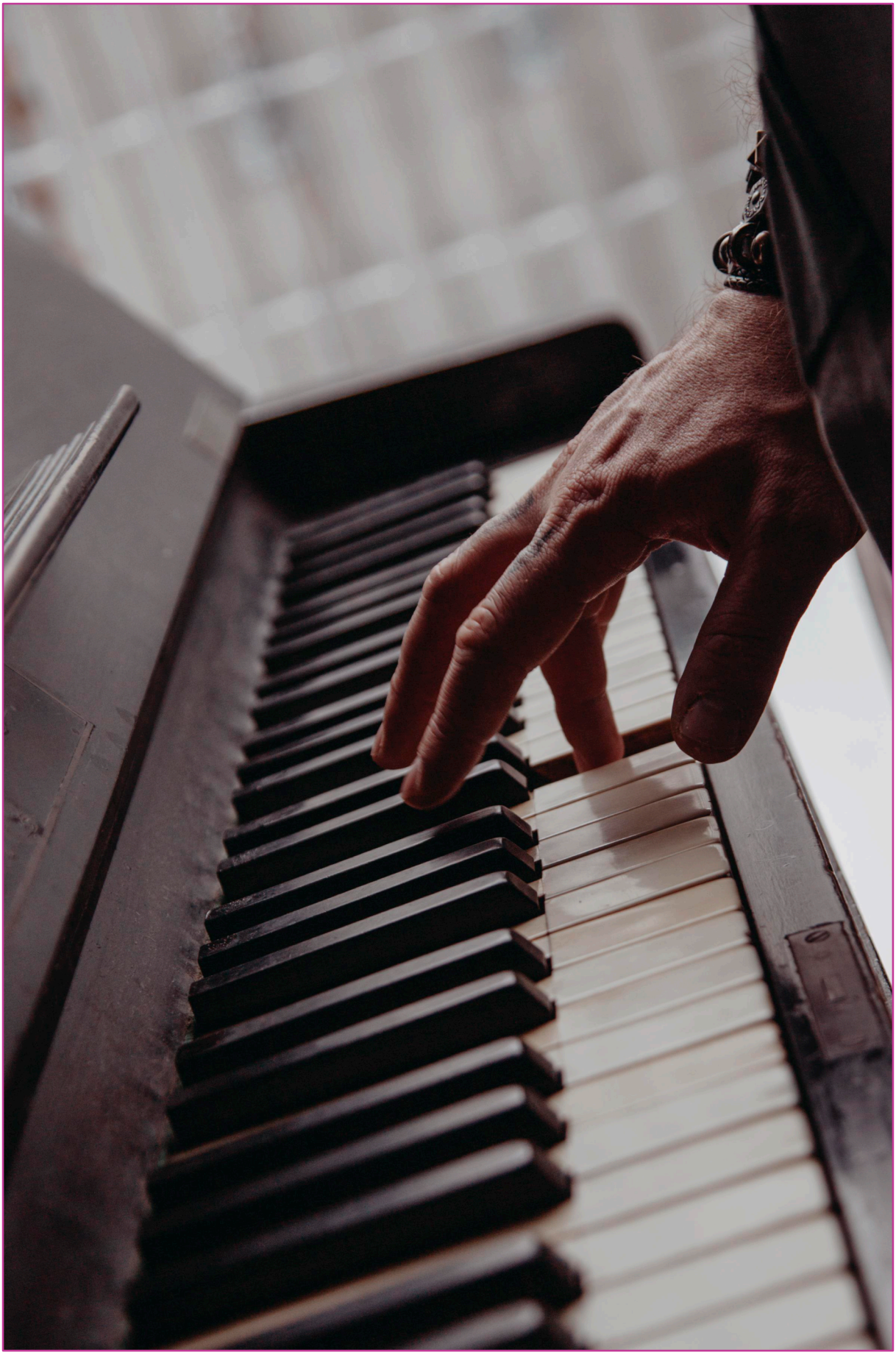
Reflects the use explained in section 2.2.3 on warm-up, technical work, and repertoire practice.

- **Warm-up**: Section for including lessons that help warm up the hands before moving on to more complex lessons.
- **Fundamentals**: Lessons that cover basic and technical aspects are shown in this section. Useful for internalizing them before progressing to more advanced lessons.
- **Development**: For deeper practice, showing more complex lessons, whether they deal with fragments that combine several elements or full arrangements.
- **Improvisation**: Although the section is present in the program, no improvisation lessons have been included yet.

Each of these sections that reflect the function of the lesson in the routine is duplicated—one dedicated to showing all lessons that contain the selected elements, and the other showing only the lessons selected for the routine.

Goal elements

The program includes a dedicated section for goal management, allowing users to define and visualize the objectives to be achieved during practice. Currently, only a few basic goals have been included as examples, but the structure is ready to be expanded and adapted to the user's needs. This section is also presented in two sections: one for all available goals and another for those selected in the routine although the latter is not in operation.



Program

3.1 Functional Explanation

This section describes the general functioning of the program. For a detailed explanation of the internal workings of the application, please refer to the appendix I-III.

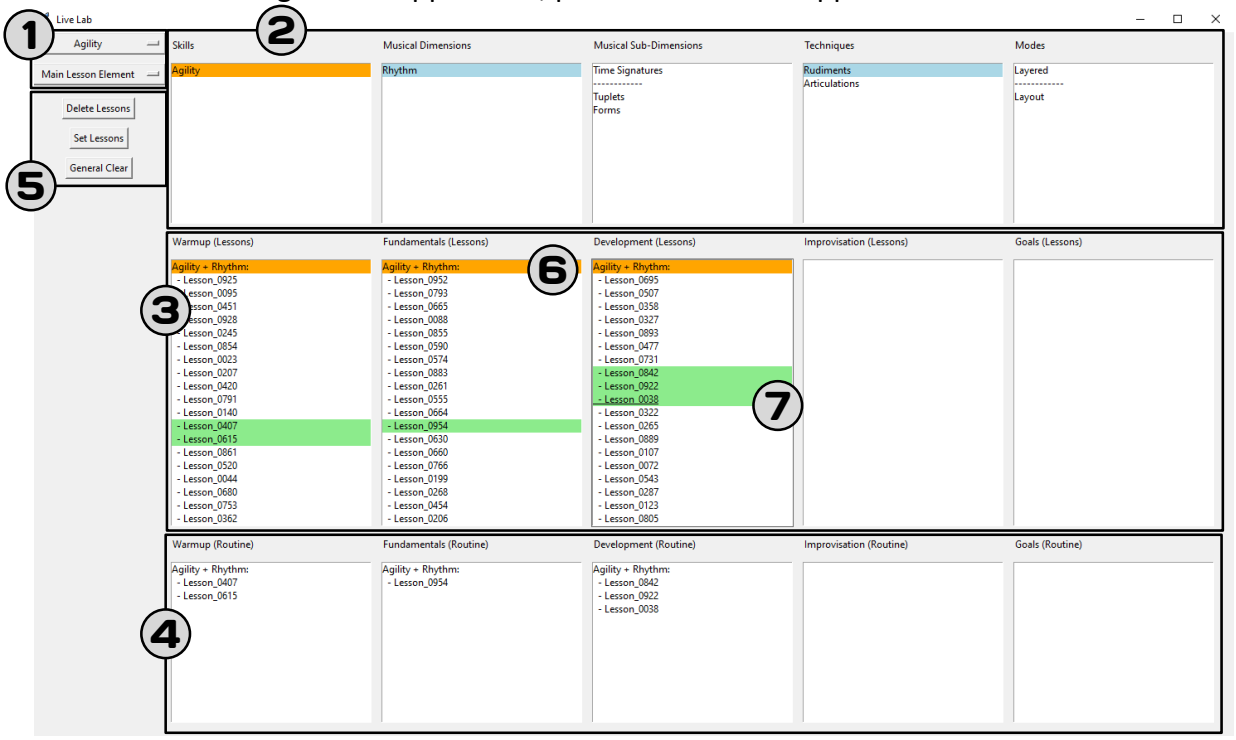


Figure 1: Graphical interface (GUI) of the program

1. **Main element menubuttons**
2. **Musical element listboxes**
3. **Lesson and objective listboxes**
4. **Routine listboxes**
5. **Buttons**
6. **Lessons groups**
7. **Colors**

General interface

The program is structured into three main sections [Fig. 1: 1-5], complemented by visual references that facilitate the work [Fig. 1: 6 and 7].

MAIN ELEMENT MENUBUTTON [1]

Starting point of the program. The element on which the practice will be based is selected.

Main Musical Element	Main Lesson Element
Displays the related musical elements, along with associated lessons and objectives.	First identifies the lessons that contain the selected characteristic and then shows the musical elements present in that set of lessons.

LISTBOXES [2-4]

Drop-down lists that allow the selection and combination of musical elements and lessons, including the initial main element. Each row serves a specific function:

- **First row (Musical Elements)**: Displays the available musical elements, organized as described in section 2.3.1.
- **Second row (Lessons and Objectives)**: Shows all lessons that include any of the selected elements, grouped by combinations of those elements. This makes it easy to identify which lessons work on the desired elements. This row also includes a specific listbox for objectives.
- **Tercera fila (Rutina)**: Similar to the second row, but only shows the selected lessons, allowing quick and easy visualization of the personalized practice routine.

BUTTONS [5]

- **Clear Routine**: Resets non-set lessons, making it easier to try different combinations and find the most suitable routine.
- **Setting Routine**: Once the desired lesson combination is selected, this button allows setting them in the routine, ensuring the selection is not lost when using the *clear routine* button. This enables the creation of complex routines through successive layers of selection.
- **General Reset**: Deletes all selections made, including fixed lessons, allowing the user to start again from scratch.

LESSON GROUPS [6]

As items are selected, lessons are dynamically grouped into distinct sets of combinations of the selected items and for which at least one lesson is available.

COLORS [7]

The interface uses a color-coding system to facilitate the identification of selected elements in the listboxes:

- **Orange**: Highlights the selected main element, considered the most important for the routine. In the first musical row, it highlights the main element if it's musical; in the second and third rows, it highlights the combination of elements that includes this main element.
- **Blue**: Indicates the musical elements selected after the main one, as well as lesson categories whose element combinations do not include the main one. This distinction helps quickly identify which lessons include the main element.
- **Green**: Marks the lessons selected in the second listbox row. This selection is retained even after using the routine clear button, facilitating the exploration of different combinations.

Functioning Flow

The program flow is structured around two categories, each with two elements:

1. TYPE OF SELECTED ELEMENT: Musical y Lesson-Based

- **Musical**: Includes the selectable elements from the *main musical element* menubutton and the musical listboxes in the first row, as described in section [2.3.1].
- **Lesson-Based**: Includes elements selectable from the main lesson element menubutton and the available lessons in the listboxes for warm-up, foundations, and development in the second row. In this case, the options vary: in the menubutton, they refer to lesson characteristics (see [2.3.2]), while in the listboxes, they refer to the lessons themselves.

2. TYPE OF PROCESS: Search and Combination

- **Search**: The process by which elements matching or related to the selected one are located.
- **Combinación**: Involves comparing the results obtained in the previous search with those of the last selected element. The combination varies depending on the listbox type: in the musical ones (first row), the intersection of both results is sought (or they are shown separately if there's no match); in the lesson and objectives listboxes (second row), the results are summed, regrouping the lessons into the resulting categories.

Given the above, the general processes that occur are the following:

1. Selecting a main musical element

When a musical element is selected as the starting point, the program establishes connections with other related musical elements (see [2.3.1]) and displays the lessons that contain the selected element.

2. Selecting a main lesson element

It searches for lessons with the selected element and then analyzes the musical elements contained within that set of lessons.

3. Selecting an item from the musical listboxes (first row)

A process similar to the one described in the first point is carried out, combining the results with those previously obtained.

4. Selecting an item from the lessons listboxes (second row)

The program analyzes the information of the selected lesson, identifies the corresponding musical elements, searches for lessons that include any of those elements, and combines them with the previous results.

3.2 Uses

Once the functioning of the program has been described, it is important to mention some possible uses. Regardless of the user's goals and objectives, the program is designed to be used following specific steps.

Choosing the main element

This is the starting point of the program. The choice is made based on what one wishes to practice. One might want to work on a specific skill, delve into a particular style, or apply a specific theoretical aspect. The program allows any of the elements in the *menubuttons* to be set as the axis.

Strategic combination of elements

Once the first element is chosen, the combination process begins. As many musical elements as desired can be selected, although as combinations are made, the possibilities may narrow to only show those with greater affinity among them, which helps to make more synergistic selections.

A specific lesson to work on can also be selected. This choice will modify the musical results, narrowing them to those contained in the selected lesson. This makes it possible to choose musical items shared with the selected lesson for a more specific result, with a list of lessons closer to the one chosen.

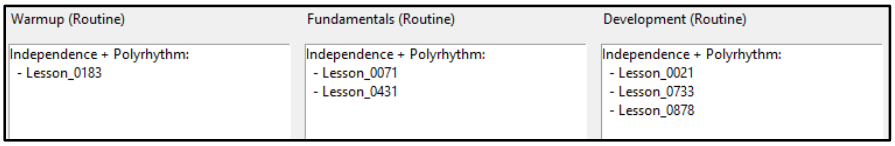
Smart prioritization

As musical elements are selected, lessons will be grouped into clusters of selected elements, allowing for a greater variety of options: Should one choose those that contain the most desired combination of elements, or those that include the highest number of selected elements? This makes it possible to either select lessons that most closely match the intended practice, or maximize the number of practiced elements by choosing those that contain a greater number of selected items.

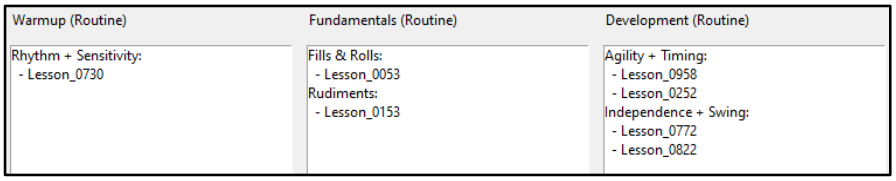
Personalized routines

If lessons ensuring the desired practice have been previously selected, here they are combined into the desired routine.

By selecting two or more lessons from the same group—especially if they are of different types—you create a *chunking*-style routine. To do this, simply select a core element and combine it with others, choosing lessons that belong to the same group of elements or at least share some of them.



The *routine set* button can be used to apply a different combinations of the main element plus items from *listboxes*. This allows for a block-based approach, with each block focusing on a different aspect. This refers to a *blocking*-style practice aimed for a more general balance.



For a routine that blends *chunking* and *blocking*, lessons from different categories can be selected, especially if they share some element. This allows continued focus on a specific practice while varying different facets, aiming for balance and versatility at the same time.

Any combination of the different uses is possible through the routine set button.

Experimentation

Sometimes it is unclear what one wants to practice. In these cases, trying random selections may provide clues about what could be interesting to work on. It may also be useful to go beyond usual patterns and try new things or detect deficiencies (undeveloped areas) and see how they can be addressed.

Conclusions and Future Developments

Proposal Summary

This dossier presents a program designed to facilitate the creation of routines that maximize instrumental practice. The main objective is to optimize musical development, whether through a succession of different lessons focused on a specific aspect, the integration of multiple elements in a single session, or the search for a balance between various facets of learning. To achieve this, the system relies on an organized structure where each element belongs to a functional category, allowing for a visualization of the whole and conscious selection of which aspects to work on. The program offers:

✔ A comprehensive overview of the factors involved in musical practice.

✔ Tools for deciding which core element will be the focus of the routine.

✔ Options for combining elements and generating specific, personalized practices.

✔ The ability to create routines tailored to individual needs.

In short, it helps achieve a smarter organization of practice in a simple way, thanks to a system that relates each of its components.

Current Limitations

Despite the value of the program and the fact that it fills a gap in the practice of musical instruments, it has notable limitations and it is evident that the current version is incomplete:

✘ Limited database

The system relies on a sufficiently large and varied lesson database to be truly useful. Currently, the database contains only generic example lessons, which limits its applicability. Manually creating new lesson cards can be a laborious task for many users.

✘ Unable to save routines

In the current version, there is no save function, which means that all information is lost when the program is closed.

✖ Underdeveloped sections

Some sections, such as improvisation, modes, and lack associated elements, which limits their usefulness and the overall scope of the program.

✖ Lack of contextual information

Lessons are displayed in the lists without additional relevant information, such as difficulty level, instrument, style, or specific features. This makes it difficult to make informed decisions and forces users to select lessons without knowing their real context, which goes against the goal of conscious and meaningful practice.

Improvement Proposals

To overcome these limitations and enhance the program, the following lines of development are proposed:

Implementation of advanced filters

Incorporate a filtering system that allows for a more sophisticated combination of musical and instructional elements, making it easier to define and diversify practice.

Expansion and improvement of the database

Include more lessons and elements, as well as establish more explicit relationships between them, to enrich the combination possibilities and cover a broader spectrum of styles, instruments, and levels.

Enriched contextual information

Associate each lesson and musical element with relevant information (theory, level, achievements, etc.), enabling a more informed selection tailored to the user's needs.

Intelligent relationships between lessons

Implement a system that displays the dependencies and progressions between lessons of different difficulty levels, making it easier to advance progressively or select directly based on the user's experience.

Saving routines

Create a routine saving system for reuse or comparison between different configurations.

Conclusions

In short, this program represents a tool for organizing and optimizing instrumental practice, providing a flexible and customizable structure that adapts to different styles, levels, and goals. While the current version is functional and offers a solid framework, its true potential will be realized with the development of the proposed improvements, which will allow for a richer, more informed, and more effective experience. We invite users and collaborators to participate in this evolutionary process, convinced that together we can transform the way musical practice is approached and contribute to the growth of the educational and artistic community.

Appendix I: Modules

Information

This module serves as an essential auxiliary note for the rest of the program. It does not import any other modules, but it is used by most of them to centralize information about the structure and relationships of the elements. Its main functions are:

- **To gather information** about the composition and organization of variables.
- **To provide this information** to the other modules.

Module Sections

1. Core Categories
2. Relationships
3. Matrix of Matrices
4. Subcategories
5. Interface
6. Styles

1. Core Categories

Defines the main musical categories used throughout the program via the Categorias dictionary. These categories are employed across all main modules:

- **Skills**: Agility, Independence, Endurance, Sensitivity, Timing, Improvisation/Creativity
- **Musical Dimensions**: Rhythm, Harmony, Structure.
- **Musical Sub-Dimensions**: Time Signature, Musical Figures, Tuplets, Scales, Chords, Chord Progressions, Sections, Forms.
- **Techniques**: Articulations, Fills & Rolls, Polyrhythm, Rudiments, Swing, Arpeggios, Ornaments, Tremolo and Vibrato.
- **Modes**: Free, Linear, Layered, Cue Point Drumming, One-Handed Drumming, Layout.

There is an unused dictionary called Categorias_Mejorado intended for future expansion and better organization of musical elements.

2. Relationships

This section defines the existing relationships between different items within the main categories. A relationship matrix is used for each relevant category pair.

- *Skill ↔ Ability*
- *Skill ↔ Musical Dims*
- *Skill ↔ Musical Sub-Dims*
- *Skill ↔ Techniques*
- *Skill ↔ Modes*
- *Musical Dims ↔ Sub-Musical Dims*
- *Musical Dims ↔ Techniques*
- *Musical Dims ↔ Modes*
- *Musical Sub-Dims ↔ Techniques*
- *Musical Sub-Dims ↔ Modes*

Note: With the future implementation of Categorias_Mejorado, the structure of these matrices might change to accommodate more complex sublists.

3. Matrix of Matrices

All relationship matrices are grouped into a master variable called Matrices_de_Relaciones. This structure is mainly used by the search module to analyze and manage relationships between elements.

4. Interface

Defines the main instruments for the user interface, represented as a list of options. In the current version, the only two considered interfaces are pads and keys.

5. Styles

A list of musical styles considered in the program, obtained through the analysis of all lessons.

6. *A*dditional Variables and Dictionaries

Several additional dictionaries used throughout the program are included here:

- **`listboxes_selections`**: An accumulative list of elements selected by the user.
- **`lessons`**: A dictionary that records the lessons (as keys) along with the matching selected elements (as values).
- **`selected_lessons`**: A dictionary that stores the selected lessons, where the keys are the listboxes they belong to, and the values are the lessons themselves.

GUI (Graphical User Interface)

The graphical user interface (GUI) acts as the functional core of the program, serving as a bridge between the user and the internal engine. It manages both the display of results and the dynamic interaction with the various processing modules, enabling exploration, selection, and combination of musical content.

The interface allows the user to execute the program's main functionalities through a series of interconnected processes:

1. **Displaying the interface** on screen in a column layout
2. **Automatically presenting the result** related to the selected central musical or educational element
3. **Selecting menu options** that dynamically modify the output

Module Sections

1. Auxiliary Functions
2. Main Functions
3. GUI: Organization of Frames and Elements

1. Auxiliary Functions

This section groups several helper functions designed to facilitate structure, reusability, and clarity in the main code. These functions serve specific roles in the visualization, organization, and writing of results to the interface, with a modular focus and an emphasis on efficient handling of selection lists, thematic grouping, and visual representation.

a. Interface Element Creation

`create_label(text, row, column, frame)`

Auxiliary function to create label elements within a frame. Accepts the label text, its position in the grid (row and column), and the container frame. It improves code readability and avoids repetition when building the GUI.

`create_listbox(row, column, frame, width=30, height=12)`

Enables creation of listbox elements with customizable size and position, also within a frame. Facilitates uniform and controlled placement of these visual containers.

b. List Content Management

`delete_result(List)`

Removes all items from a list (usually a Listbox), leaving it empty. Used to reset visual states or prepare spaces before inserting new results.

c. Semantic Reorganization by Groupings

`rearrange_matching(lesson_coincidence_lists)`

This function plays a key role: it restructures matching items (associated with "Warm-up", "Fundamentals", and "Development") based on thematic groups defined in `lessons`.

The procedure is as follows:

- Groups keys based on their associated values in `lessons`.
- Sorts these groups according to the number of elements they contain.
- Iterates through each thematic section and assigns items to the corresponding group within `lesson_coincidence_lists`, explicitly adding that grouping category.

This process is essential to enable a hierarchical and categorized visualization of lessons, avoiding flat lists and promoting a clearer, more meaningful structure for the user.

d. Writing and Final Display of Results

`write_results()`

This is the core function for writing to the interface, responsible for:

1. **Reorganizing** the previously identified matches using `rearrange_matching()`.
2. **Transforming the resulting groupings** into a clean visual structure with thematic headers.
3. **Writing that information into the visual Listboxes** (`result_lists`), ensuring items are properly formatted (with indentation and clean labels).
4. **Applying distinctive colors** based on the relevance of elements:
   - **Orange** for the `main_element`.
   - **Light blue or green** for items selected by the user, depending on the context.
5. **Identifying which selected items should be part of the final structure** (`selected_lessons`) and writing them to the corresponding listboxes (`listboxes_3`), maintaining the hierarchy by subgroups.

2. Main Functions

This code block defines the main functions of an interactive system that allows searching, selecting, and organizing elements from various Listboxes within a graphical interface. The functions are organized into three groups:

a. Main Search Function: `search_function()`

`search_function(value_inside, search_type="element")`

Purpose: To search and display results for a central element based on the selected type of search (`musical` or `lesson`).

Parameters:

- `value_inside`: Input text field (e.g., an `Entry`) from which the value to be searched is taken.
- `search_type`: Indicates the type of search: `musical` or `lesson`.

How it works:

1. Clears previous results (`delete_result()`), resets auxiliary lists and global variables.
2. If `search_type` is `musical`:
   i. Locates the entered musical element, determines which category it belongs to and the corresponding results are searched.
   ii. Records the musical element in the `listboxes_selections` dictionary.
3. If `search_type` is `lesson`: Searches in the lessons and displays the results.
4. Finally, it updates the visual results using `write_results()`.

b. Selection Function from Listboxes: `on_select()`

`on_select(event)`

Purpose: To register which items the user has selected in a Listbox and process them.

How it works:

- Detects which Listbox triggered the event (`event.widget`).
- Adds the selected items to the `listboxes_selections` dictionary, organizing them by `listbox`.
- Call to the `combine()` function (not defined here) with the first selected item to process it.
- Then updates the results with `write_results()`.

c. Lesson Management Functions

These functions allow manipulation of the selected lessons (e.g., locking them in place, clearing them, displaying them).

`action_1()`

Purpose: To visually clear the lesson-related Listboxes and filter out any lessons that haven't been marked as "locked" (`state=True`).

Main Steps:

1. Clears the contents of the final Listboxes (`listboxes_3`).
2. Clears the selected items in the result-related Listboxes (`listboxes_selections`).
3. Reconstructs a new `selected_lessons` dictionary, keeping only the lessons marked as locked (`state=True`).
4. Fills the final `listboxes` with the locked lessons, including formatted headers for each subset of lessons.

`action_2()`

Purpose: To mark all selected lessons as definitive (state `True`).

How it works: Iterates through the `selected_lessons` dictionary and updates the state of all items to `True`.

`general_clear()`

Purpose: To restore the system to its initial state.

Actions performed:

- Clears results shown in both the result Listboxes and the final lesson Listboxes.
- Resets all auxiliary structures: `listboxes_selections`, `lessons`, and `selected_lessons`.

3. GUI: Organization of Frames and Elements

This block of code aims to visually structure the graphical user interface of an application built with the `Tkinter` library. It is organized into frames, dropdown menus (menubuttons), buttons, and listboxes that allow the user to explore and select musical elements and structured lessons.

a. Main Window Structure

Two main zones are defined within the interface:

`options_frame`: A left-hand sidebar that contains selection menus and buttons.

`results_frame`: The right-hand zone divided into several rows and columns, where results are displayed in listboxes.

b. Dropdown Selection Menus (Menubuttons)

"Central Musical Element" Menu

A `Menubutton` associated with a `StringVar` allows the user to select a central musical element from `categories` dictionary. Each selection triggers the `search_function()` function with the `musical` type.

"Central Lesson Element" Menu

This second menu allows the filtering of lessons by:

- **Interface** (e.g., type of instrument)
- **Level** (hierarchically divided into easy, intermediate, hard)
- **Genre** (musical genres)

Each option also calls `search_function()`, but this time with the "lesson" type.

c. Action Buttons for Lessons

Three buttons are placed in the sidebar:

Clear Lessons: Executes `action_1()`, which clears all non-fixed selections.

Fix Lessons: Executes `action_2()`, which marks the current selections as fixed.

General Clear: Calls `general_clear()`, resetting the entire system (visual and logical)..

d. Result Organization in 3 Visual Rows

The right-hand side of the interface (`results_frame`) is structured into 3 horizontal rows with 5 columns each:

Row 1: Musical Elements

Contains `Listboxes` displaying musical elements organized by categories.

- Skills
- Musical Dimensions
- Musical Sub-Dimensions
- Techniques
- Modes

Each listbox is linked to a selection event (`on_select`), allowing interaction with the items.

Row 2: Lessons as Results

Displays the filtered lessons based on the selected central element or lesson. The columns represent lesson categories.

Interactive listboxes are also used here.

- Warm-Up
- Fundamentals
- Development
- Improvisation
- Goals

Row 3: Routine Selection

This row reflects the personalized routine based on the selected lessons. The categories match those of Row 2.

These columns are populated in a processed manner after executing `action_1()` and `action_2()`.

e. Final Initialization

`main.mainloop``()`: This command launches the main loop of the program and keeps the interface open, awaiting user interaction.

Main_Functions

This module acts as an intermediary between the graphical user interface (GUI) and the functional modules for searching and combining musical and lesson-based content. Its purpose is to prevent overloading the main GUI module by separating search and processing logic into a cleaner structure.

Main Functions:

- **`search_musical_results(parameter)`**: Performs a search based on a central musical element.
- **`search_lesson_results(parameter)`**: Explores musical elements present in a specific lesson.
- **`combine(selection)`**: Manages the full flow of data analysis and combination, integrating musical and lesson elements to generate refined results.

This module uses shared structures (lists, dictionaries) defined in other files such as `Information.py`, and delegates analytical tasks to functions in `Search.py` and `Combination.py`.

Module Sections

1. Global variables
2. Helper functions
3. Main functions

1. Global Variables

`index`: A list used to store the category and index of the selected central element.

2. Helper Functions

`delete_lists(lst)`: Clears the contents of a list if valid. Used to reset the state of result lists.

3. Main Functions

`search_musical_results``(parameter)`

Purpose: Search for and display results related to a central musical element.

Steps:

1. Locates the parameter in the musical categories (`categories`) and saves its position.
2. Calls `show_category_options()` to display sub-options depending on the category.
3. Calls `call_lessons_goals(parameter, True)` to find associated lessons.
4. Returns the result lists (`matching_results`).

`search_lesson_results``(parameter)`

Purpose: Search for musical elements present in a selected lesson.

Steps:

1. Clears the lists related to musical elements.
2. Finds lessons related to the parameter.
3. Analyzes those lessons using `analyze_lessons2(file)` to detect which musical elements they contain.

`combine``(selection)`

Purpose: Process and combine both musical and lesson-based results.

Steps:

1. Clears internal structures (`matches`, `originals`).
2. Saves a copy of previous results in `originals`.
3. Determines whether the selection is a musical element or a lesson:
   - If musical: updates category and index.
   - If lesson: analyzes its content with `analyze_lessons2(selection)`.
4. Searches for lessons related to the parameter via `call_lessons_goals(selection, False)`.
5. Removes duplicates.
6. Calculates matches between original and new lists using `match_lists(originals, matched_lists, matches)`.
7. Updates `matched_lists` with the new results.

Search

This module's main function is to act as a bridge between the data files (lessons and objectives in `.md` and `.json` formats) and the modules responsible for search, combination, or visualization of information. It functions as an engine for analysis and match extraction, structuring and classifying educational content into lists organized by key categories.

Module Sections

1. Auxiliary functions
2. Main functions

1. Auxiliary Functions

`clear_lists(Lists)`: Clears multiple match lists, preventing data accumulation across consecutive searches.

`call_lessons_goals(parameter, clear)`: Launches the search functions together, including `Muestra_Opciones_Ar`.

2. Main Functionalities

Content Analysis

`analyze_lessons(file)`: Analyzes `.md` files to find textual matches with predefined musical categories.

`analyze_lessons2(file)`: Performs an equivalent analysis on `.json` files based on metadata (tags, grade, genre, etc.).

Match Extraction from Files

`show_options_ar(path, main_element, included_lists, terms, clear=True)`:

Searches for matches in `.md` file headers based on system-defined keywords.

`show_options_ar2`

`(path, main_element, included_lists, terms, clear=True)`:

Performs the same operation on `.json` files, evaluating the presence of the central element in associated metadata.

Relational Classification

`show_category_options(index)`:

Given a central element and its index, determines which other categories are related based on a matrix of relationships among musical dimensions (skills, techniques, modes, etc.).

Combination

This module is designed to manage and process matches between categorized elements, facilitating comparison and consolidation of data previously retrieved from different sources through searches and relational structures. It acts as a bridge between search results and the user interface, avoiding direct overload of the graphical module (GUI).

Module Sections

1. Global variables
2. Functions

1. Global Variables

`matches`: A dynamic dictionary that stores the combined results by category after analysis.

`originals`: Auxiliary copy used to restore the original state or for later comparison.

2. Functions

`clear_lists(*lists)`: Utility function to empty multiple lists passed as arguments. Prevents unwanted accumulation in successive comparisons.

`match_lists(list1, list2, matches)`

Purpose: Detect and record matches between two categorized dictionaries of elements.

Functionality:

- For the categories "Warm-up", "Fundamentals", and "Development", elements are combined directly, as they are commonly used as complementary labels rather than hierarchies.
- For all other categories:
  - Direct matches (intersections) are recorded.
  - If there are no matches but both lists contain content, all elements from both are inserted with a visual separator ("----------").
  - If one list is empty, the content of the other is added without separation.

Parameters:

- `list1`, `list2`: Categorized dictionaries containing lists of elements.
- `matches`: External dictionary where results are stored.

Return Value: The updated `matches` dictionary with category-based results (updated in-place).

Appendix II: Variables y Dictionaries

This section documents the main variables, lists and dictionaries used in the code, detailing their function within the system to facilitate understanding and maintenance.

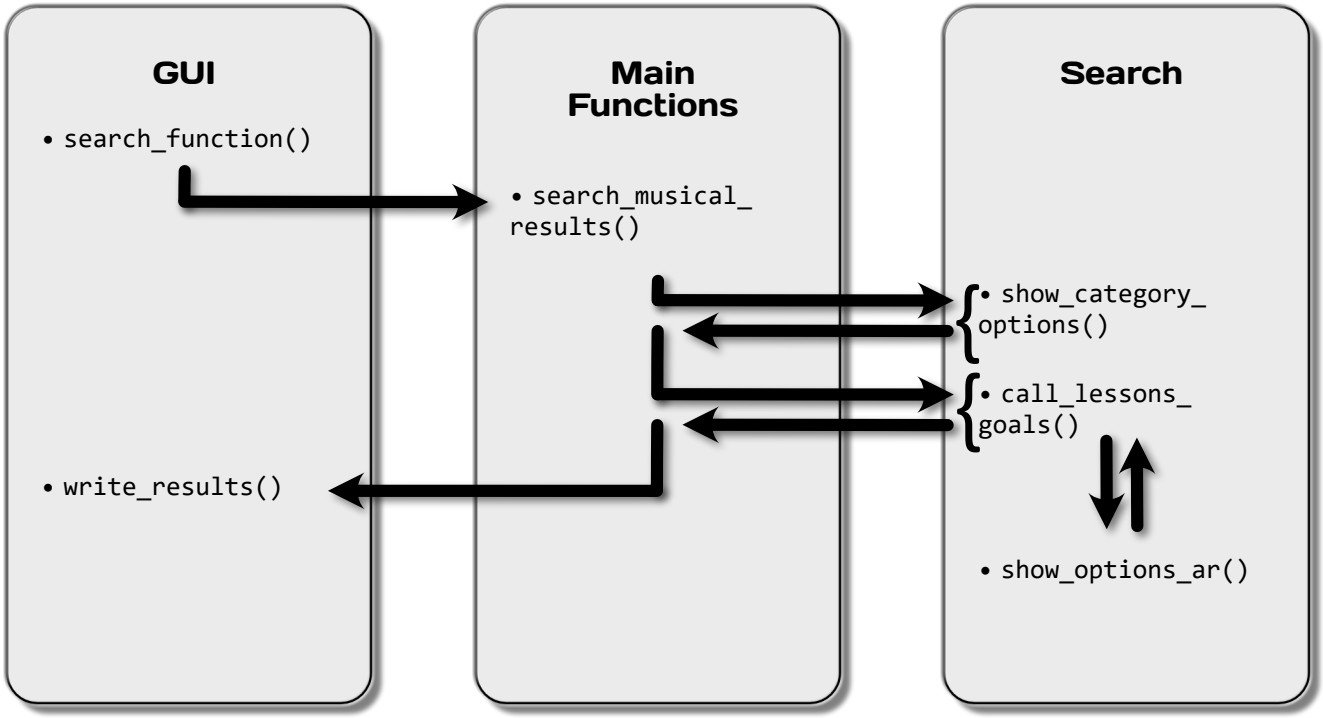
Name	Type	Description	Used modules
`categories`	dict[str, list[str]]	Groups musical elements.	Information, GUI, Search
`Relationship_Matrices`	dict[str, dict[str, matriz]]	Organizes relationship matrices between musical elements.	Information, Search
`matched_lists`	dict[str, list]	Stores lists of matches for each musical and lesson element.	Information, GUI, Main_Functions, Search, Combination
`listboxes_selections`	dict	Records the elements selected by each listbox.	Information, GUI

Name	Type	Description	Used modules
`lessons`	dict	Stores lessons as keys, associated with the selected elements.	GUI, Search
`selected_lessons`	dict	Stores the lessons explicitly selected by the user.	Information, GUI
`main_element`	var(str)	Stores the central selected value in the interface, used as a focus or reference.	GUI
`matches`	dict[str, list]	Stores the matching elements for each category.	Combination
`originals`	dict[str, list]	Saves the original (unfiltered) lists for each category.	Combination

Appendix III: Flows

Selection of a Central Musical Element

This section describes the process by which the user selects a central musical element from the interface and how that selection triggers a series of functions that generate related results and update the visible information in the application.



Detailed Process

1. The user selects a musical element through a `Menubutton` that contains the various musical categories defined in the Categorias dictionary.

2. Upon selection, **the function `search_function()` (located in the GUI module) is executed**, where the argument `value_inside` represents the selected value.

The following operations are then executed:

• The function `delete_result()` is called, which clears the contents of the `result_lists` dictionary. This dictionary contains the current results displayed on screen, obtained from interaction with the various listbox elements.

• The dictionary `listboxes_selections`, which stores the user's active selections, is cleared.

• The contents of the `lessons` dictionary are deleted; this dictionary uses lesson names as keys and associates them with the selected central element.

• The current value of `main_element` is retrieved from the `Menubutton`.

3. Next, **the function `search_musical_results()`** (located in `Main_Functions`) is executed. Its purpose is to **group the process of analysis and filtering of musical results without overloading the GUI interface.**

This process includes:

• Clearing the variable `index`, which will be used to record the location of the element within the `categories` dictionary.

• Searching for the index of the `main_element` within its respective category and storing it in `index` as a pair (`category_name`, `position_in_list`).

4. Based on this, **the function `show_category_options()`** (located in the `Search` module) **is called**. This function identifies the musical elements related to the `main_element` through the following steps:

a. The dictionary `matched_lists`, which will store the results, is cleared.

b. The `relations` dictionary, which defines associations between categories, is declared.

c. Using the matrices defined in `Relationship_Matrices`, the system calculates which elements from each category are related to the selected element and stores them in `matched_lists`.

d. If the `main_element` does not belong to the "Skills" category, it is also explicitly included in its corresponding category within `matched_lists`.

e. The function returns `matched_lists`.

5. Then, **`call_lessons_goals()` is executed**. This function searches the database for lessons and objectives containing the parameter (the `main_element`) and performs the following:

- Calls `show_options_ar2()` to display matching lessons (of types "Warm-up", "Fundamentals", and "Development").
- Executes `show_options_ar()` again to identify associated pedagogical goals.

6. A copy of the `matched_lists` dictionary is made into the variable `matching_results`, which will be returned as the final result of the process.

7. The system checks which category the `main_element` belongs to.

8. Finally, **`write_results()`** (located in the GUI module) **is executed** to display the generated results on screen.

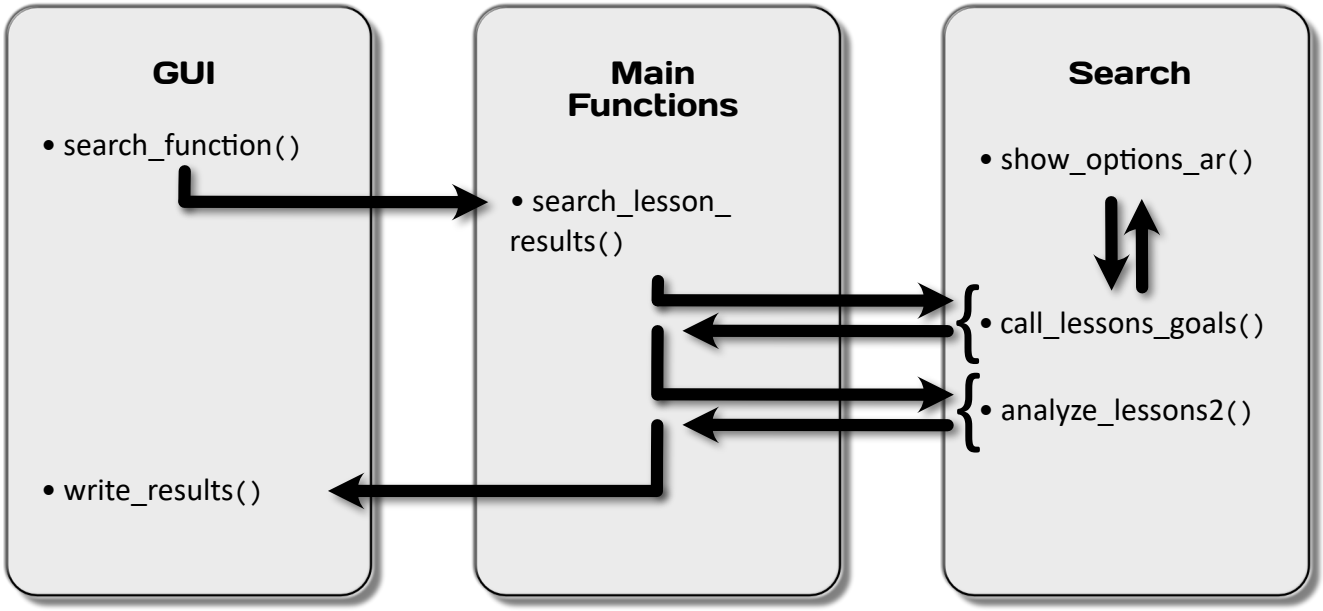
Functioning Summary

Based on the selection of the central musical element, the system generates a list of related elements using the relationship matrix, managed by the `show_category_options()` function.

Simultaneously, a search is performed for lessons and objectives that include the selected main element. This search is handled by the `call_lessons_goals()` function, which delegates the display and filtering of relevant results to `show_options_ar()`.

Selection of a Central Educational Element

This section describes the process by which the user selects a central lesson element, and how this selection triggers a series of functions to generate related results. The system uses this element as a starting point to detect relevant lessons and, from them, identify the associated musical elements.



Detailed Process

1. The user selects an educational element through the corresponding `Menubutton`, which displays the available categories.

2. This action triggers the `search_function()` function (located in the `GUI` module), where the selected value (`value_inside`) corresponds to an educational element.

The following actions are then executed:

- `delete_result()` is called, which clears the contents of the `result_lists` dictionary, holding the elements shown in the different `listbox` components.
- The `listboxes_selections` dictionary is cleared, which stores the user's active selections.
- The `lessons` dictionary is cleared, which stores lesson names as keys and associates them with the central educational element.
- The `main_element` value is retrieved, which corresponds to the element selected in the `Menubutton`.

3. The `search_lesson_results()` function (located in the `Main_Functions` module) **is executed**. This function groups the process to avoid overloading the GUI logic. Inside this function:

- The lists within `matched_lists` are cleared for the following categories: "Skills", "Musical Dimensions", "Musical Sub-Dimensions", "Techniques", and "Modes".

4. Then, the `call_lessons_goals()` function (from the `Search` module) **is called**, where `parameter` refers to the `main_element`. This function analyzes the database to identify lessons and objectives that include this parameter.

Inside this function:

- `show_options_ar2()` is executed to filter and present matching lessons (of type "Warm-up", "Fundamentals", and "Development").
- `show_options_ar` is executed again to identify the related pedagogical goals.

5. The `analyze_lessons()` function is then called for each item found under the keys "Warm-up", "Fundamentals", "Development", and "Improvisation" within `matched_lists`.

This function analyzes the associated lessons to extract the musical elements they contain, generating a result list consistent with the central lesson element. The process unfolds as follows:

a. Different paths are constructed to locate each lesson's corresponding file.

b. The file is opened and read, with the content stored in `files`.

c. The function iterates through all categories and elements in the `categories` dictionary, searching for matches in the file text.

d. It checks whether each category already exists in `matched_lists` (to avoid key errors).

e. It ensures that no duplicate entries are added to the respective lists.

f. If a match is found, it is added to the appropriate category in the `matched_lists` dictionary.

6. Finally, the `write_results()` function (in the `GUI` module) **is executed** to update the interface with the obtained results.

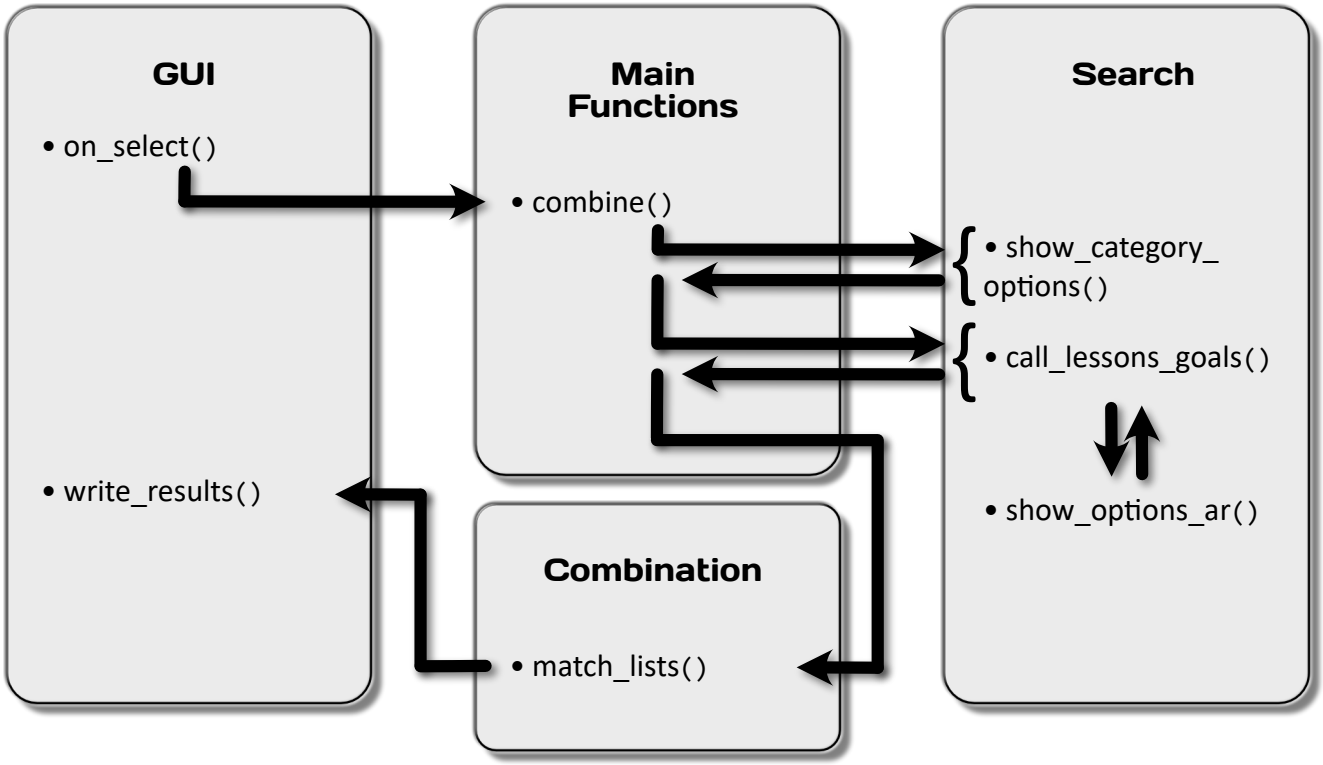
Functioning Summary

Based on a central educational element, the system identifies all lessons and objectives in which that element appears. This analysis is handled by the `call_lessons_goals()` function, and within it, by `show_options_ar()`.

Afterward, the system analyzes the files corresponding to those lessons to extract the musical elements they contain. This task is performed by the `analyze_lessons()` function, which generates a precise and duplicate-free list, organized by musical categories.

Selection of an Item from a Musical Listbox

This section describes the process by which the user selects an item from one of the active musical `Listbox` components. Such selection triggers a chain of functions that identify related musical elements, lessons, and goals. The system uses the selected item as an entry point to generate a combination of results consistent with previous selections.



Detailed Process

1. **The user selects a musical item in one of the active musical `Listbox`** of the interface.

2. **This action triggers the `on_select`** event (in the `GUI` module), a function that acts as an intermediary between the interface and the combination system. The following actions are then performed:

   - The `Listbox` that triggered the event is stored in the variable `current_listbox`.
   - The corresponding key for the event is added to the `listboxes_selections` dictionary if it hasn't been registered yet.
   - The selected indices from the active `Listbox` are retrieved and stored in the variable `selected_index`.
   - The text associated with the selected index (i.e., the name of the musical item) is retrieved.

3. The function `combine()` (from the `Main_Functions` module) **is executed**. This function is the core of the combination system. Inside it:

- The `matches` and `originals` dictionaries, which store previous results and newly combined results, are cleared.
- The contents of the `matched_lists` dictionary are copied into `originals`, preserving the previous selection.

4. Then, the function `show_category_options()` (in the `Search` module) **is called**. It searches for related musical items through category relationships. Inside this function:

- The `matched_lists` dictionary is cleared.
- The `relations` dictionary is defined, containing relationships between musical categories.
- Based on `relations` and `Relationship_Matrices`, a list of related elements is determined and stored in `matched_lists`.
- If the item's category is not "Skills", the `main_element` is stored in its corresponding category.
- The updated contents of `matched_lists` are returned.

5. The function `call_lessons_goals()` (from the `Search` module) **is then called** to query the database of lessons and objectives. Inside this function:

- `show_options_ar2()` is executed to retrieve matching lessons (of types "Warm-up", "Fundamentals", and "Development").
- `show_options_ar()` is executed again to retrieve pedagogical objectives related to the selected item.
- Duplicate entries in each of the resulting lists are then removed.

6. The function `match_lists()` (from the `Combination` module) **is executed**. This function compares previous results with the new ones:

- It detects matches between lists of lists.
- If there are matches, they are directly included in the results.
- If there are no matches, the lists are combined and separated by "----------"
- For the keys "Warm-up", "Fundamentals", and "Development", lists are combined directly without comparison.
- Once completed, the `matched_lists` dictionary is cleared again, as the current results are stored in `matches` .
- The contents of `matches` are transferred to `matched_lists`, which becomes the updated base of related items.

7. Finally, the `write_results()` function (in the `GUI` module) **is executed**, visually updating the interface with the current results derived from the selected item.

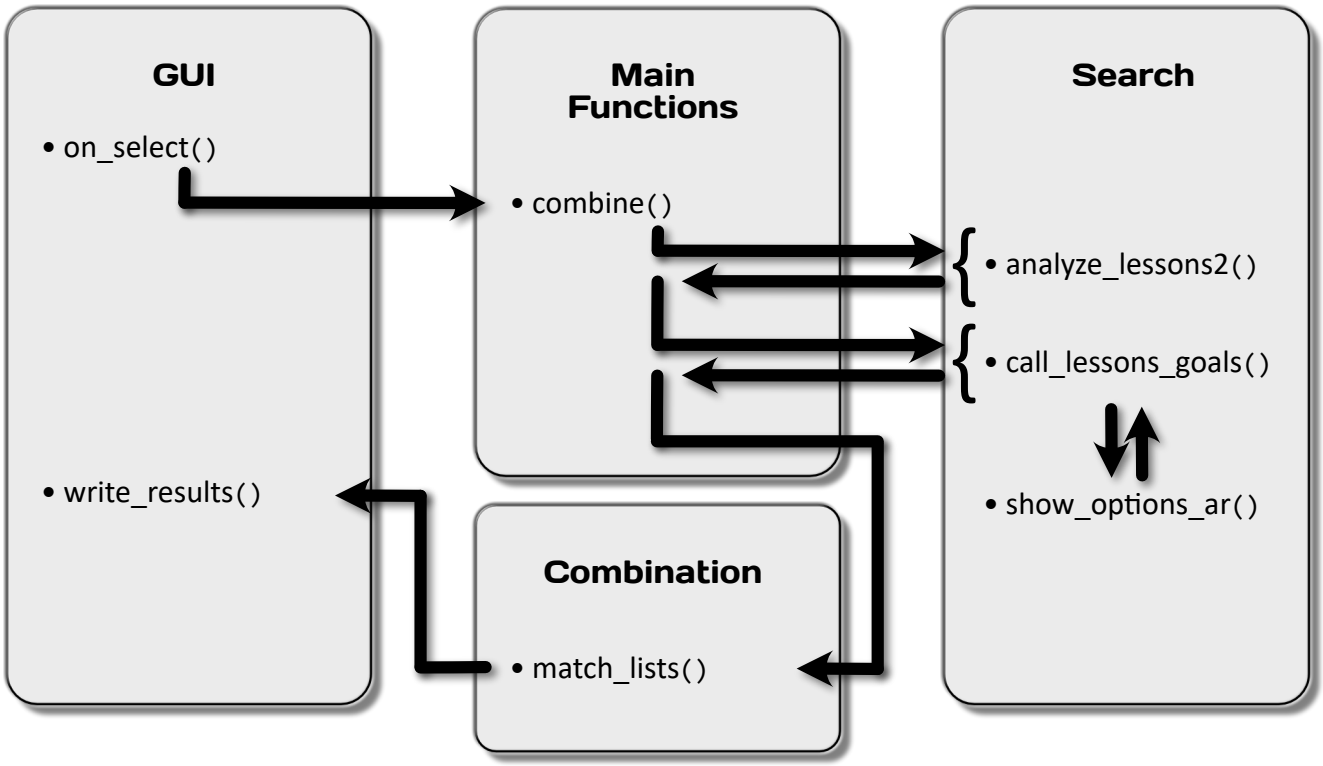
Functioning Summary

When the user selects a musical item from a Listbox in the interface, the system activates a series of functions that process this selection as a new reference point. First, the item is identified, and internal records are updated while preserving previous selections. Then, a combination process is triggered, which clears previous results and searches for musically related items through predefined categories, thus updating the set of relevant items.

Based on this new selection, the system queries the database to identify associated lessons and pedagogical objectives. The new results are compared with the previous ones and integrated coherently—either through direct matches or visual combination. Finally, the interface is automatically updated with the new results, ensuring a dynamic and context-aware user experience.

Selection of an Item from a Lesson Listbox

This module enables the selection of an item from one of the musical Listboxes, triggering an automated process that analyzes and combines information linked to the chosen item. The involved functions detect musical elements associated with the selected lesson and generate a network of related lessons and pedagogical objectives.



Detailed Process

1. **Item selection**
   - The user selects an item from a Listbox.
   - The `on_select(event)` function is triggered.
2. **`on_select()` Function**
   - Acts as a bridge between the selection and the analysis/combination process.
   - Internally:
     a. Registers the originating `Listbox` in `current_listbox`.
     b. Adds the selection key to `listboxes_selections` if absent.
     c. Extracts the selected index.
     d. Retrieves the name of the selected item.
3. **Function `combine()`**
   - Core of the analysis-combination process, contrasting new vs. previous results.
   - Steps:
     a. Clears `matches` and `originals`.
     b. Copies previous data from `matched_lists` to `originals`.
     c. Clears `matched_lists`.
4. **Function `analyze_lessons2()`**
   - Searches for musical elements within the selected lesson.
   - Steps:
     a. Locates the corresponding .json in `/Lessons` folder.
     b. Reads file content.
     c. Iterates through predefined `categories`.
     d. Verifies that the category exists in `matched_lists`.
     e. Adds any matching tags to the correct category.
5. **Function `call_lessons_goals()`**
   - Executed for each of: {"Skills", "Musical Dimensions", "Musical Sub-Dimensions", "Techniques", "Modes"} (if present).
   - Finds lessons and objectives containing the given element.
   - Calls `show_options_ar()` to display lesson and goals matches.

6. **Duplicate Removal**
   - Cleans duplicate items from all lists in `matched_lists`.
7. **`match_lists()` function**
   - Compares previous and new lists by category.
   - If matches exist, stores them. Otherwise, combines with "----------" separator.
   - Keys "Warm-Up", "Fundamentals", and "Development" are merged directly.
8. **Cleanup**
   - `matched_lists` is cleared post-comparison.
9. **Result Transfer**
   - Results from `matches` are copied back to `matched_lists`.
10. **GUI Update**
    - `write_results()` function updates the interface with current results.

Functioning Summary

Information Extraction

- Using `analyze_lessons2()`, the system detects musical elements in the selected lesson.
- With `call_lessons_goals()`, it identifies related lessons and objectives.

Result Combination

- The `match_lists()` function compares and merges previous and new data to generate a coherent, updated network of relevant content.

Appendix IV: Bibliography

Below is a selection of relevant bibliographic references used to support and inform the development of this dossier. While not all existing documents on the subject have been included, the sources listed address key aspects related to musical practice, self-regulated learning, performance excellence, and the psychology of music. This bibliography provides a solid and up-to-date theoretical framework that underpins the contents and proposals of this work.

Araújo, M. V. (2016). Measuring self-regulated practice behaviours in highly skilled musicians. Psychology of Music, 44(2), 278–292. https://doi.org/10.1177/0305735615573563

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. Psychological Review, 100(3), 363–406.

Hallam, S. (1995). Professional musicians' approaches to the learning and interpretation of music. Psychology of Music, 23(2), 111–128.

Lehmann, A. C., & Gruber, H. (2006). Deliberate practice in music: Development and psychometric validation of a standardized measurement instrument. Psychology of Music, 34(3), 339–356.

Lehmann, A. C., Sloboda, J. A., & Woody, R. H. (2007). Psychology for musicians: Understanding and acquiring the skills (2nd ed.). Oxford University Press.

McPherson, G. E. (2001). A longitudinal study of self-regulation in children's musical practice. Psychology of Music, 29(1), 111–138. https://doi.org/10.1080/14613800120089232

Miksza, P. (2009). Relationships among achievement goal motivation, impulsivity, and the music practice of collegiate brass and woodwind players. Bulletin of the Council for Research in Music Education, 188, 7–22.

Miksza, P., Prichard, S., & Sorbo, D. (2018). Using a music microanalysis protocol to enhance instrumental practice. Music Education Research, 20(2), 194–207.

Parncutt, R., & McPherson, G. E. (Eds.). (2002). The science & psychology of music performance: Creative strategies for teaching and learning. Oxford University Press.

Williamon, A. (Ed.). (2004). Musical excellence: Strategies and techniques to enhance performance. Oxford University Press.