

IL TEAM F.S.C. PRESENTA:

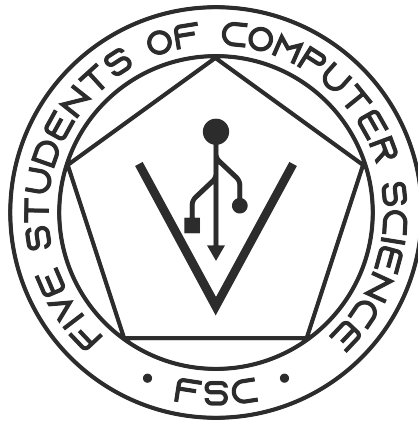


STRUMENTALMENTE

Il team:

Alessandro Annese
Davide De Salvo
Andrea Esposito
Graziano Montanaro
Regina Zaccaria

Informatica e Comunicazione Digitale - (TA)
A.A. 2018/19



F.S.C. — Five Students of Computer Science

Copyright © 2019 F.S.C.

Concesso in licenza secondo i termini della Licenza Apache, versione 2.0 (la "Licenza"); è proibito usare questo file se non in conformità alla Licenza. Una copia della Licenza è disponibile all'indirizzo:

<http://www.apache.org/licenses/LICENSE-2.0>

Se non richiesto dalla legislazione vigente o concordato per iscritto, il software distribuito nei termini della Licenza è distribuito "COSÌ COM'È", SENZA GARANZIE O CONDIZIONI DI ALCUN TIPO, esplicite o implicite. Consultare la Licenza per il testo specifico che regola le autorizzazioni e le limitazioni previste dalla medesima.

Indice

| | |
|--|-----------|
| I Pianificazione | 11 |
| 1 Descrizione del sistema | 13 |
| 1.1 Introduzione | 13 |
| 1.2 Definizione dello scopo | 13 |
| 1.3 Il committente | 13 |
| 1.4 Caratteristiche degli utenti | 13 |
| 1.5 I vincoli | 15 |
| 1.5.1 Requisiti minimi | 15 |
| 1.5.2 Budget | 16 |
| 2 Standard e Manuali di stile | 17 |
| 2.1 Manuale di stile | 17 |
| 2.1.1 I colori | 17 |
| 2.1.2 La navigazione | 17 |
| 2.1.3 Le pagine | 17 |
| 2.2 I contenuti | 17 |
| 2.2.1 Unità 1: la teoria | 17 |
| 2.2.2 Unità 2: la pratica | 18 |
| 2.2.3 I test di autovalutazione | 18 |
| 3 Costi | 19 |
| 3.1 Documento di pianificazione | 20 |
| 3.2 Risorse | 21 |
| 3.2.1 Risorse umane | 21 |
| 3.2.2 Risorse informative | 21 |
| 3.2.3 Risorse applicative | 22 |
| 3.2.4 Risorse post-produzione | 22 |
| 3.3 Stima dei costi | 22 |
| II Progettazione | 23 |
| 4 Introduzione | 25 |
| 4.1 I concetti | 25 |
| 4.1.1 Definizione dei concetti | 25 |
| 4.2 I task | 26 |
| 5 Progettazione | 27 |
| 5.1 Il modello RMM | 27 |
| 5.1.1 Il modello ER | 27 |
| 5.1.2 Progettazione delle slice | 28 |
| 5.1.3 Modello della navigazione | 29 |
| 6 Design | 31 |

| | | |
|------------|--|-----------|
| 6.1 | I colori | 31 |
| 6.2 | Le gabbie logiche | 31 |
| 6.3 | Le icone | 35 |
| 7 | I contenuti | 37 |
| 7.1 | Bibliografia | 37 |
| 7.1.1 | Istruttori ed esperti | 37 |
| 7.1.2 | Libri | 37 |
| 7.1.3 | Materiale online | 37 |
| III | Realizzazione | 39 |
| 8 | Le tecnologie utilizzate | 41 |
| 8.1 | I linguaggi | 41 |
| 8.2 | Il codice | 41 |
| 9 | Avviare dal codice sorgente | 43 |
| 9.1 | Il Makefile | 43 |
| 10 | Documentazione del codice | 45 |
| 10.0.1 | Classes | 45 |
| 10.1 | Funzioni | 45 |
| 10.1.1 | accordo | 46 |
| 10.1.2 | openChildWindow(pageUrl, [windowIcon]) | 46 |
| 10.1.3 | createWindow() | 47 |
| 10.1.4 | promptModal(parentWindow, [options], callback) | 47 |
| 10.1.5 | openMobileNavigation() | 47 |
| 10.1.6 | drop(name, [defaultLinkClass]) | 47 |
| 10.1.7 | setLinks(links) | 48 |
| 10.1.8 | initialize(initial, base) | 48 |
| 10.1.9 | changeTopic(topicName, [base]) | 49 |
| 10.1.10 | initializeQuiz() | 49 |
| 10.1.11 | changeQuizSlide(finalSlide) | 49 |
| 10.1.12 | playStopAudio(audioTagId, buttonRef, stopButtonId) | 50 |
| 10.1.13 | showExitDialog() | 50 |
| 10.1.14 | showExitFromQuizDialog(toOpen) | 50 |
| 10.1.15 | showQuizDialog(nomeQuiz, score, total) | 51 |
| 10.1.16 | openInBrowser(link) | 51 |
| 10.1.17 | openModal(content, [options], [windowIcon]) | 51 |
| 10.1.18 | openOnKeyboardShortcut(shortcut, content, [openAsModal]) | 51 |
| 10.1.19 | script_load() | 52 |
| 10.1.20 | replace_selected() | 52 |
| 10.1.21 | verify_and_store() | 52 |
| 10.1.22 | correct_chord() | 52 |
| 11 | Bug e problemi noti | 53 |
| 11.1 | La velocità | 53 |

| | | |
|-----------|-------------------------------------|-----------|
| IV | Ringraziamenti e riferimenti | 55 |
|-----------|-------------------------------------|-----------|

Elenco delle tabelle

| | | |
|-----|--|----|
| 1.1 | Caratteristiche degli utenti | 14 |
| 3.1 | Costi previsti in ore di lavoro | 19 |
| 3.2 | Costi e percentuali di completamento | 20 |
| 6.1 | Palette dei colori | 31 |

Prefazione

Il team

“Il nostro obiettivo è quello di portare a termine questo progetto e i successivi nel migliore dei modi, nonché quello di creare una squadra forte e duratura che possa sopravvivere al termine del nostro percorso di studi.”

— Il team F.S.C.

Il team di sviluppo del progetto è il team **“F.S.C.”** (*Five Students of Computer Science*). È un team composto da cinque studenti iscritti per l'anno accademico 2018/19 al secondo anno del Corso di Laurea in *Informatica e Comunicazione Digitale (I.C.D.)* dell'Università degli Studi di Bari.

Gli studenti che compongono il gruppo, così come indicati sul frontespizio di questo documento, sono: Alessandro **Annese**, Davide **De Salvo**, *Andrea* **Esposito**, Graziano **Montanaro**, Regina **Zaccaria**. Il referente del gruppo per il presente progetto è A. Esposito.

Il progetto

Il progetto “StrumentalMente” nasce come progetto d’esame per il corso di *Progettazione e Produzione Multimediale* del secondo anno del corso di laurea in I.C.D. dell’Università di Bari.

Le fasi dello sviluppo di questo sistema seguono il **modello di Alessi & Trollip** per la progettazione e lo sviluppo di un’applicazione multimediale.

Il presente testo è un’unione dei vari documenti prodotti durante la progettazione e lo sviluppo di “StrumentalMente”, ovverosia quello di pianificazione, di progettazione e quello di test. Si è scelto di presentarli in un unico documento per favorire il processo di stampa, nonché per mostrare tutte le fasi dello sviluppo in modo organico e completo.

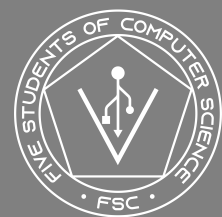
Tutto i file multimediali, il codice sorgente nonché questo documento e quelli sopraccitati sono disponibili per la consultazione su GitHub al seguente link.

<https://github.com/F-S-C/StrumentalMente>



STRUMENTALMENTE - DOCUMENTAZIONE

PIANIFICAZIONE



1 Descrizione del sistema

1.1 Introduzione

A seguito di un *brainstorming* condotto dal gruppo F.S.C., si è deciso di sviluppare una applicazione multimediale che ha l'obiettivo di avvicinare gli utenti alla musica e aiutarli nell'imparare a suonare uno strumento musicale di loro gradimento.

Tale idea è frutto di una difficile scelta fra una serie di idee concepite durante la sopracitata fase di brainstorming. I criteri di scelta sono stati dettati dalle attitudini personali dei componenti del gruppo di lavoro, in composizione con le possibilità fornite dalla musica nel campo della multimedialità.

Il titolo "StrumentalMente", scelto in quanto ritenuto accattivante, deriva dalla condensazione delle due parole "*strumentale*", che rimanda agli strumenti musicali, e "*mente*", che ricorda l'obiettivo finale del sistema: insegnare a suonare uno strumento.

L'applicazione sarà sviluppata in modo da esser rilasciata come applicazione *desktop* e sarà liberamente scaricabile *online*.

1.2 Definizione dello scopo

L'applicazione multimediale StrumentalMente ha l'obiettivo di:

- Avvicinare gli utenti alla musica
- Dare la possibilità agli utenti di imparare a suonare uno o più strumenti
- Dare la possibilità agli utenti di acquisire la capacità di riconoscere la natura degli accordi più comuni
- Fornire degli esercizi per valutare in autonomia l'apprendimento o per migliorare l'utilizzo dello strumento

1.3 Il committente

Il committente dell'applicazione è il docente del corso di Progettazione e Produzione Multimediale dell'anno accademico 2018/19 dell'Università di Bari (CdL¹: Informatica e Comunicazione Digitale), la Prof.ssa Rosa Lanzilotti.

La consegna del sistema multimediale è stimata per il mese di giugno 2019.

1.4 Caratteristiche degli utenti

Da una attenta fase di studio delle possibili tipologie di utente a cui l'applicazione è rivolta, si sono potute individuare tre classi fondamentali di utenti. Tali classi sono, tuttavia, molto flessibili: essendo basate sulla quantità di conoscenze nel campo musicale degli utenti, non è possibile determinare nette linee di separazione, bensì solo delle linee guida.

¹ CdL: abbreviazione di "Corso di Laurea".

Le categorie di utenti individuate sono le seguenti:

- **Neofita:** l'utente conosce nulla o quasi nulla della teoria musicale; non conosce alcuno strumento; potrà utilizzare l'applicazione per avvicinarsi al mondo della musica.
- **Intermedio:** l'utente conosce poco o nulla della teoria musicale; conosce, anche se poco, uno strumento musicale; potrà utilizzare l'applicazione per migliorare le proprie conoscenze sia in campo teorico che pratico.
- **Avanzato:** l'utente conosce almeno le basi della teoria musicale; conosce bene almeno uno strumento musicale; potrà utilizzare l'applicazione per migliorare le proprie conoscenze, conoscere nuovi strumenti e traslare le proprie conoscenze su altri strumenti.

Le classi di utenti sopra elencate sono descritte in tutte le loro caratteristiche nella seguente tabella (Tabella 1.1). Si noti che le età che sono state associate a ogni categoria di utente sono frutto di una stima empirica e sono puramente indicative: le conoscenze in campo musicale non sono facilmente associabili a un'età media.

Tabella 1.1: Caratteristiche degli utenti

| | Neofita | Intermedio | Avanzato |
|--|------------------------------------|-------------------------------------|------------------------------------|
| Età | Dai 13 anni in su | Dai 18 anni in su | Dai 22 anni in su |
| Livello di istruzione | Intermedio | Buono | Ottimo |
| Capacità di lettura | Buona | Buona | Buona |
| Motivazione | Buona | Buona | Buona |
| Conoscenze preliminari | Nessuna conoscenza teorico/pratica | Scarse conoscenze teoriche/pratiche | Buone conoscenze teoriche/pratiche |
| Abilità necessarie | Nessuna | Nessuna | Nessuna |
| Competenze informatiche | Capacità basilari | Capacità basilari | Capacità basilari |
| Familiarità con il Web | Capacità basilari | Capacità basilari | Capacità basilari |
| Capacità di digitazione e scrittura | Capacità basilari | Capacità basilari | Capacità basilari |
| Accesso a un computer | Buona | Buona | Buona |

| | | | |
|------------------------------------|--|---|---|
| Accesso a internet | Nessuno | Nessuno | Nessuno |
| Disponibilità (in tempo) | Almeno tre ore consecutive al giorno | Almeno due ore consecutive al giorno | Disponibilità saltuaria |
| Obiettivo dell'applicazione | Insegnare le basi della musica, avvicinare l'utente all'uso di uno strumento | Migliorare le conoscenze dell'utente, fornirgli un supporto in caso di difficoltà | Assistere l'utente nell'ampliare le proprie conoscenze, fornirgli un supporto per trasporre le stesse su altri strumenti. |

Come già anticipato precedentemente, il criterio con cui si sono individuate le seguenti classi di utenti si basa sulla classificazione dei livelli di conoscenza e competenza nel campo musicale degli stessi. Non è prevista alcuna classificazione basata sulle competenze di campo informatico, in quanto, come già specificato nella precedente tabella, non è richiesta alcuna abilità particolare in tale campo. Si noti che l'accesso a *internet* non è richiesto per utilizzare l'applicazione, se non per scaricare il software e/o i suoi aggiornamenti.

Competenze informatiche necessarie Come già anticipato, non sono necessarie delle competenze informatiche particolari per utilizzare l'applicazione: il sistema sarà progettato in modo da essere fruibile da qualsiasi utente che sappia interfacciarsi con un computer. L'applicazione sarà quindi progettata per essere intuitiva e semplice da utilizzare. Saranno tuttavia fornite, eventualmente, delle scorciatoie da tastiera per i più esperti per velocizzare la loro navigazione all'interno dell'ipermedia.

1.5 I vincoli

In questa sezione del documento saranno presentati i vincoli da rispettare durante lo sviluppo dell'applicazione. Tali vincoli sono frutto di discussioni tra i membri del team e il committente o di previsioni e obiettivi del team su delle fasi dello sviluppo successive.

1.5.1 Requisiti minimi

Il sistema multimediale sarà progettato per diverse piattaforme multimediali con i seguenti requisiti *hardware* minimi:

- Processore da 1 GHz
- RAM: 2 Gb
- Risoluzione dello schermo: 480 × 800

Non è inoltre richiesto alcun *software* aggiuntivo affinché l'applicazione possa essere eseguita. Tuttavia, StrumentalMente è disponibile per i soli sistemi operativi Windows e Linux.

1.5.2 Budget

L'applicazione multimediale è a scopo didattico quindi il committente non ha imposto nessun *budget*.

2 Standard e Manuali di stile

2.1 Manuale di stile

Il sistema avrà un *look* moderno e accattivante che possa enfatizzare la sua semplicità di utilizzo. A tale fine si utilizzeranno dei font *sans serif* (Montserrat e Raleway).

2.1.1 I colori

I colori predominanti saranno una miscela di colori neutri e caldi, ovverosia bianco, grigio scuro e rosso-arancio (il colore associato alla nota Do). Si utilizzeranno le linee guida dettate dal *Material Design* di Google per orchestrarli al meglio e per raggiungere l'obiettivo di un *look* semplice, moderno e accattivante.

2.1.2 La navigazione

Il sistema sarà navigabile utilizzando dei *link* e dei bottoni, che rispetteranno le linee guida del Material Design. In base al colore dello sfondo su cui i bottoni saranno inseriti, tali bottoni possono essere trasparenti con testo rosso-arancio (su sfondi chiari) o viceversa (su sfondi scuri).

2.1.3 Le pagine

Ogni pagina dell'applicazione sarà caratterizzata da un aspetto simile alle altre, con delle piccole differenze in base alla categoria di pagina e alla tipologia di informazioni che conterrà (per maggiori informazioni si veda la sezione 2.2 sui contenuti).

2.2 I contenuti

Il sistema sarà diviso in due unità: teoria e pratica. Per poter comprendere ciò che sarà presentato nell'unità della pratica, è necessario aver compreso tutto ciò che la prima sezione dell'unità della teoria presenta.

Saranno inoltre introdotti dei test di autovalutazione per valutare le competenze acquisite durante l'uso del sistema. Il superamento di tali test non è un requisito per la navigazione delle varie parti dell'applicazione, tuttavia è fortemente consigliato per assicurarsi una migliore comprensione degli argomenti avanzati.

2.2.1 Unità 1: la teoria

Questa unità è suddivisa in due sezioni:

1. Teoria musicale di livello basico
2. Teoria musicale di livello intermedio/avanzato

Teoria musicale di livello basico In questa sezione si presenteranno tutti i concetti basilari senza i quali l'utente non può utilizzare a dovere il sistema. Sarà fornito all'utente un lessico basilare che gli possa permettere di comprendere ciò che StrumentalMente (e gli approfondimenti suggeriti al suo interno) presenta e descrive.

Teoria musicale di livello intermedio/avanzato In questa sezione verranno approfonditi i concetti introdotti nella sezione precedente con uno sguardo meno rivolto alla pratica. Sarà una sezione di approfondimento che mirerà ad arricchire il vocabolario tecnico che l'utente, tramite la precedente sezione, ha iniziato a costruire.

2.2.2 Unità 2: la pratica

Quest'unità presenterà le modalità d'uso, le componenti e una lista di tecniche o accordi per vari strumenti. Gli strumenti che il team ha selezionato dopo un'iniziale fase di brainstorming sono i seguenti:

- Basso
- Batteria
- Chitarra
- Pianoforte/Tastiera

Per comprendere appieno le nozioni presentate in quest'unità è richiesto un vocabolario tecnico minimo, che è possibile acquisire completando la prima sezione della prima unità.

Si vuole sottolineare che l'applicazione StrumentalMente è un'applicazione che può essere arricchita dopo il rilascio, aggiungendo eventualmente altri strumenti alla precedente lista.

2.2.3 I test di autovalutazione

Sono previsti vari test di autovalutazione:

- Uno iniziale per permettere un'autovalutazione delle competenze iniziali
- Uno per la prima sezione dell'unità teorica
- Uno per la seconda sezione dell'unità teorica
- Uno per ogni strumento presentato all'interno dell'unità pratica (in base allo strumento che è scelto dall'utente sarà selezionato quello corrispondente)

Benché il sistema sia un sistema di *e-learning*, non sarà obbligatorio completare i vari test per proseguire nella navigazione tra le sezioni. Questa decisione del team è stata presa in seguito a delle considerazioni sull'usabilità dell'applicazione da parte di utenti che visitano il sistema più volte: poiché il sistema può essere utilizzato come supporto allo studio da altre fonti, deve essere fornita la possibilità all'utente di saltare direttamente alle nozioni a cui è interessato, senza dover necessariamente completare una serie di test autovalutativi.

Approvazione del committente

I contenuti presentati hanno ricevuto l'approvazione da parte del committente il 28 novembre 2018.

3 Costi

La seguente tabella (Tabella 3.1) contiene una lista delle attività da completare per portare al termine il progetto. Tale lista è emersa durante una discussione del *team*.

Oltre a ogni attività, sono elencate il numero di ore che si prevede siano necessarie a portare al termine le stesse.

Tabella 3.1: Costi previsti in ore di lavoro

| FASI DELLA PRODUZIONE | ATTIVITÀ | IMPEGNO ORARIO |
|--|--|----------------|
| Acquisizione del materiale | Acquisizione del materiale audio | 10 |
| | Acquisizione del materiale testuale | 10 |
| | Acquisizione del materiale video e fotografico | 10 |
| | Acquisizione del materiale di supporto (tabelle, schede, ecc.) | 4 |
| | Progettazione dei test di autovalutazione | 6 |
| | TOTALE | 40 |
| Verifica e validazione del materiale acquisito | Stesura di un inventario del materiale d'acquisto | 3 |
| | Revisione e correzione del materiale acquisito | 5 |
| | TOTALE | 8 |
| Definizione dell'interfaccia utente | Sviluppo degli standard comunicativi | 5 |
| | Realizzazione della barra di navigazione | 5 |
| | Realizzazione delle interfacce grafiche | 10 |
| | TOTALE | 20 |
| | Realizzazione delle pagine | 30 |

| | | |
|---------------|--|-----------|
| Sviluppo | Realizzazione delle interazioni tra le pagine | 12 |
| | Realizzazione e ottimizzazione dell'interazione | 8 |
| | Realizzazione dei manuali | 4 |
| | Produzione della versione α | 2 |
| | TOTALE | 56 |
| Test | Alpha test e documento di test | 10 |
| | Revisione del software | 10 |
| | Beta test e documento di test | 10 |
| | TOTALE | 30 |
| Pubblicazione | Realizzazione copia master | 2 |
| | Realizzazione delle copie per sviluppatori e committente | 2 |
| | TOTALE | 4 |

3.1 Documento di pianificazione

Il presente documento è stato modificato dopo circa una settimana di lavoro per poter includere le percentuali di completamento relative delle varie attività previste.

Le percentuali di completamento presenti in questa tabella (Tabella 3.2) sono percentuali empiriche basate su un calcolo approssimativo della mole di lavoro compiuta, che è poi stata paragonata alla mole di lavoro prevista per portare al termine una singola attività.

Tabella 3.2: Costi in ore e percentuali di completamento delle attività previste durante la pianificazione

| ATTIVITÀ | TEMPO STIMATO (ORE) | TEMPO UTILIZZATO (ORE) | COMPLE- TAMENTO PERCENTUALE |
|--------------------------------------|---------------------------|------------------------------|-----------------------------------|
| Acquisizione dei contenuti | 40 | x | x % |
| Verifica e validazione dei contenuti | 8 | x | x % |
| Definizione dell'interfaccia utente | 20 | x | x % |

| | | | |
|----------------------|----|---|-----|
| Sviluppo | 56 | x | x % |
| Test | 30 | x | x % |
| Pubblicazione | 4 | x | x % |

3.2 Risorse

Di seguito, saranno illustrate tutte le risorse utilizzate per la realizzazione del sistema multimediale.

3.2.1 Risorse umane

La distribuzione del lavoro nel team di progettazione di sviluppo del sistema è stato diviso nel seguente modo:

- **Alessandro Annese:** gestione e produzione degli elementi multimediali del sistema; supporto nella creazione delle pagine del sistema e nella gestione della documentazione.
- **Davide De Salvo:** gestione e produzione degli elementi multimediali del sistema; creazione delle pagine del sistema.
- **Andrea Esposito:** gestione della parte “*backend*” dell’applicazione con una speciale attenzione all’utilizzo dei *framework* necessari allo sviluppo dell’applicazione; gestione della documentazione e supporto nella creazione delle pagine del sistema.
- **Graziano Montanaro:** gestione e revisione dei contenuti testuali dell’applicazione; creazione delle pagine del sistema.
- **Regina Zaccaria:** gestione e revisione dei contenuti testuali dell’applicazione; creazione delle pagine del sistema.

Ovviamente, la suddivisione dei lavori precedentemente presentata non esclude la possibilità di variazioni successive o di collaborazioni fra membri del team con compiti differenti nella risoluzione di *task* più complessi di quelli attualmente previsti.

3.2.2 Risorse informative

Tutte le informazioni riguardo gli strumenti musicali e le loro modalità di utilizzo (in senso stretto) sono frutto di studi personali dei singoli componenti del team.

Le informazioni relative alla teoria musicale e strumentale (nonché le modalità di presentazione delle stesse) saranno reperite da libri di testo o da esperti del settore: si prevede una spesa di circa 100€ (cento euro) per poter ricevere supporto nella stesura dei contenuti da parte di esperti del settore (maestri di musica dell’*Accademia musicale Francisco Tàrraga*).



Figura 3.1: Logo dell’Accademia Musicale Francisco Tàrraga

Si provvederà autonomamente, con un eventuale supporto da parte degli esperti, alla creazione di tutto il materiale multimediale di supporto (foto, audio e video).

3.2.3 Risorse applicative

Nello sviluppo dell'applicazione saranno utilizzati i seguenti applicativi:

- *Adobe Photoshop CC* (modifica delle immagini)
- *Adobe Illustrator CC* (creazione dei loghi e degli schemi)
- *Adobe Premiere CC* (modifica e montaggio video)
- *Audacity* (modifica e montaggio audio)

Tutti i programmi precedentemente elencati sono o gratuiti o saranno utilizzati nelle loro versioni di prova.

Inoltre, si utilizzerà Git come sistema di controllo delle versioni, in combinazione con la piattaforma GitHub, che sarà usata per condividere i file sorgenti del sistema.

3.2.4 Risorse post-produzione

Per la pubblicazione saranno necessari:

- 6 CD-Rom per la creazione del master e delle copie per la distribuzione
- Inchiostro e carta per la stampa dei manuali e dei loro prototipi

Saranno distribuite due versioni dell'applicativo: una installabile tramite un *installer* per *Windows* e una eseguibile direttamente da CD-Rom.

3.3 Stima dei costi

Si prevedono costi per:

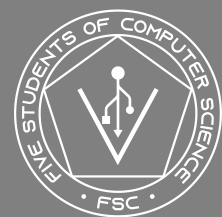
- Consulenza di esperti del settore (si veda la sezione 3.2.2)
- Stampa e rilegatura dei manuali e dei documenti di progetto
- Produzione e decorazione delle copie su CD-Rom

I costi potrebbero variare in base al numero di ore che le componenti del *team* impiegano nella varie fasi dello sviluppo.



STRUMENTALMENTE - DOCUMENTAZIONE

PROGETTAZIONE



4 Introduzione

L'*hypermedia* StrumentalMente ha l'obiettivo di avvicinare gli utenti al mondo della musica e di aiutarli a migliorare le loro conoscenze riguardanti questa arte.

4.1 I concetti

L'applicazione conterrà i seguenti concetti:

- Concetti teorici
- Strumenti
- Accordi
- Quiz

L'applicazione è quindi divisa in sezioni (tante quante i singoli concetti). Tali sezioni sono ulteriormente divise in "unità" all'interno delle quali ogni argomento sarà presentato attraverso una combinazione di testo, immagini, video e audio.

L'applicazione avrà una pagina iniziale che ha lo scopo di introdurre il concetto di "musica", nonché di presentare all'utente delle istruzioni basilari sull'uso del sistema.

StrumentalMente sarà introdotto da una semplice *splash screen* che permetterà all'utente di accedere più gradualmente all'applicazione. Si accompagnerà il tutto con un manuale utente.

Il sistema conterrà una sezione per ogni unità che possa permettere all'utente di mettersi alla prova tramite domande a risposta multipla o "minigiochi".

4.1.1 Definizione dei concetti

Concetti teorici I concetti teorici sono alla base dell'interfaccia con il mondo della musica. Sono suddivisi in due unità (più propriamente dette "livelli"): *teoria di livello base* e *teoria di livello intermedio/avanzato* in cui, rispettivamente, saranno presentati i concetti necessari alla comprensione dei contenuti e saranno presentati dei concetti di approfondimento. Gli argomenti saranno trattati in modo da essere facilmente comprensibili dagli utenti che non hanno conoscenze in questo campo.

Strumenti Come tutte le arti, la musica necessita di una grande abilità pratica. Tramite le varie unità (che si identificano con gli strumenti presenti nell'applicazione, ovvero: batteria, basso, chitarra e pianoforte) in cui è diviso questo concetto l'utente può interfacciarsi sulle tecniche base, intermedie e avanzate dello strumento scelto.

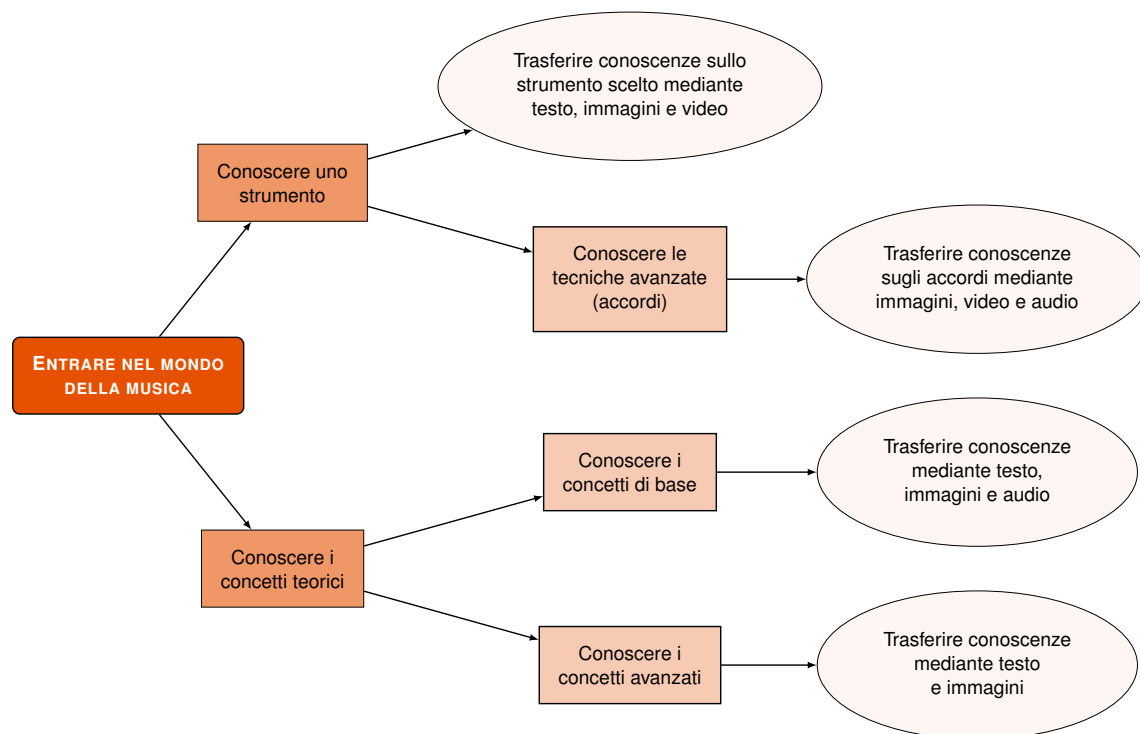
Accordi Una parte degli strumenti prevede la possibilità di suonare degli accordi. Nelle varie unità (che, anche in questo caso, si identificano con gli strumenti) di questo concetto, l'utente potrà imparare a suonare degli accordi con il proprio strumento.

Quiz Per ogni unità dei concetti precedentemente definiti, sarà presente un'unità di *test* che consentirà all'utente di verificare le proprie conoscenze. Non è necessario superare un test per procedere nell'utilizzo dell'applicazione, ma è consigliato.

4.2 I task

1. Conoscere la teoria che c'è dietro l'arte della musica
2. Conoscere le tecniche basilari di almeno uno strumento

Figura 4.1: I task di StrumentalMente. Sono stati disposti orizzontalmente (da sinistra verso destra) affinché potessero facilmente essere inseriti in questo documento.



5 Progettazione

5.1 Il modello RMM

Nella progettazione del sistema si sono individuate quattro entità differenti:

Concetto teorico Classe dei concetti musicali prettamente teorici, necessari a una giusta comprensione del mondo musicale. Sono caratterizzati da un titolo (che li identifica) e una descrizione (la quale è ciò che interessa all'utente).

Strumento Classe degli strumenti musicali. Sono caratterizzati da un nome (che li identifica) e tutte le informazioni annesse (tra cui una descrizione della struttura dello strumento e le tecniche più comuni).

Accordo / Tecnica Classe degli accordi musicali, identificati da un nome, contenenti una descrizione (in termini di note musicali) e degli esempi utili agli utenti. Poiché alcuni strumenti *non* sono polifonici (e non sono quindi in grado di produrre accordi) sono considerate appartenenti a questa classe di oggetti anche le tecniche di utilizzo più avanzate dei suddetti strumenti.

Quiz Classe di oggetti che consentono all'utente di ripetere un argomento. Contengono un numero identificativo, una domanda e fino a quattro risposte differenti. Fanno parte di questa classe anche le domande interattive, ovverosia quelle che richiedono all'utente di *"costruire"* una risposta (si considerano aventi una sola risposta possibile).

5.1.1 Il modello ER

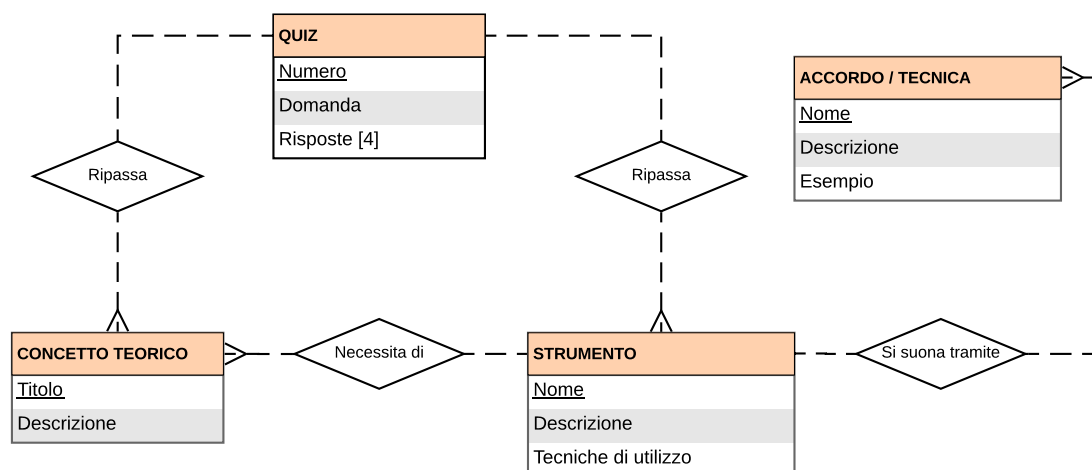


Figura 5.1: Il modello ER di StrumentalMente. Si noti che la notazione utilizzata prevede le *"chiavi primarie"* sottolineate e i campi multipli scritti con la notazione tipica dell'array (con il numero di ripetizioni racchiuso tra parentesi quadre).

5.1.2 Progettazione delle slice

Si sono progettate le *slice* riferite alle quattro entità precedentemente introdotte.

Nei seguenti schemi è indicato con un asterisco (*) la slice iniziale. Inoltre, sono indicate con delle frecce continue i link che permettono lo spostamento tra le varie slice della stessa entità (su tale freccia è posto il significato del link).

Si noti che nella progettazione delle slice dell'entità ACCORDO non si è inserita alcuna informazione in riferimento alla slice *esempio*: questo poiché la tipologia di esempio e i *media* utilizzati possono variare da strumento a strumento.

Non è visibile ne nei seguenti schemi ne nella progettazione del modello della navigazione la possibilità di suddividere le slice in varie “schermate” (a seconda della disponibilità di spaziori in relazione ai contenuti), tuttavia si tiene a precisare che è prevista la suddivisione dei contenuti di una stessa slice in più schermate.

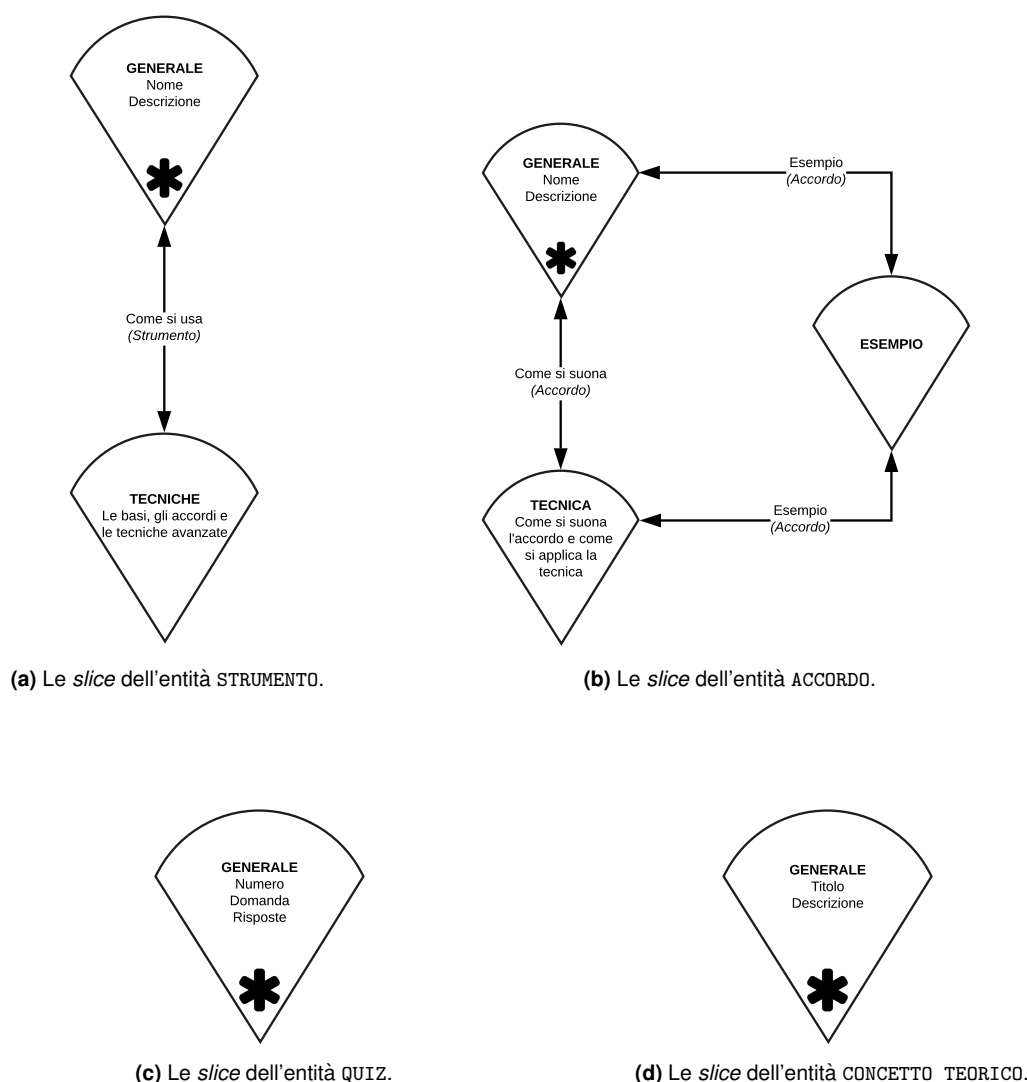


Figura 5.2: Le *slice* del modello RMM di StrumentalMente

5.1.3 Modello della navigazione

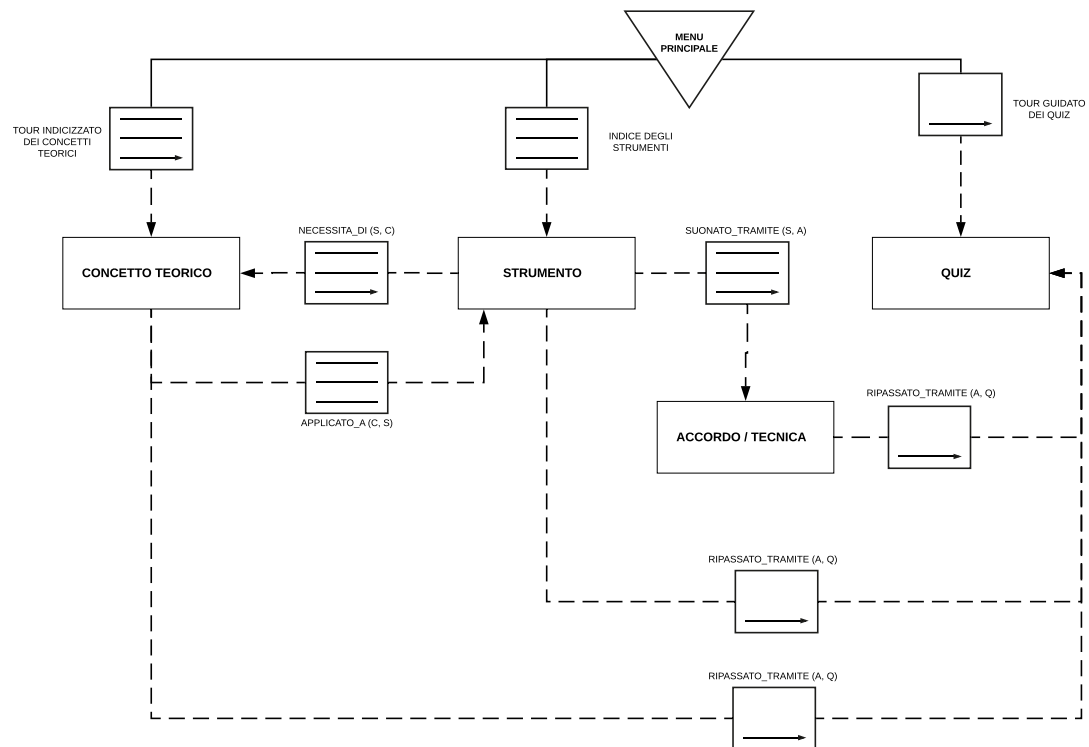


Figura 5.3: Il modello della navigazione di StrumentalMente

6 Design

6.1 I colori

Come stabilito in fase di pianificazione, l'applicazione verterà su un colore rosso-arancio, in quanto solitamente associato alla nota di Do.

Si è, quindi, generata una *palette* di colori partendo da un colore arancio scuro, simile al colore dei cachi. Si sono scelti quattro colori seguendo la “regola” della tetrade cromatica, selezionando dei colori con una distanza di trenta gradi circa (sulla ruota cromatica) dal colore principale.

Si veda la tabella 6.1 per avere dei riferimenti visivi sui colori scelti. Per ogni colore, sono presentate quattro tinte diverse (escludendo il colore “puro”, mostrato in posizione centrale) e sono riportati i vari codici in esadecimale. Inoltre, è possibile osservare la resa sia di un testo bianco che di uno nero sulle varie tinte.

Tabella 6.1: *Palette* dei colori su cui è basato il *design* di StrumentalMente

| | | | | | |
|------------------------|---------|---------|---------|---------|---------|
| Colore primario: | #FF9E6B | #FF8C4F | #E55100 | #802D00 | #571E00 |
| | #FF9E6B | #FF8C4F | #E55100 | #802D00 | #571E00 |
| Colore secondario (1): | #FFC56B | #FFB94F | #E58B00 | #804D00 | #573500 |
| | #FFC56B | #FFB94F | #E58B00 | #804D00 | #573500 |
| Colore complementare: | #6FABEF | #4D8DD5 | #0C4D95 | #012853 | #001B39 |
| | #6FABEF | #4D8DD5 | #0C4D95 | #012853 | #001B39 |
| Colore secondario (2): | #64EFC5 | #42D6A9 | #00976A | #00543B | #003928 |
| | #64EFC5 | #42D6A9 | #00976A | #00543B | #003928 |

Ai precedenti colori, vanno poi aggiunti i colori bianco (#FFFFFF), nero (#000000) e grigio all'80% (#333333), utilizzati per contrastare i colori più accesi e per il testo dell'applicazione.

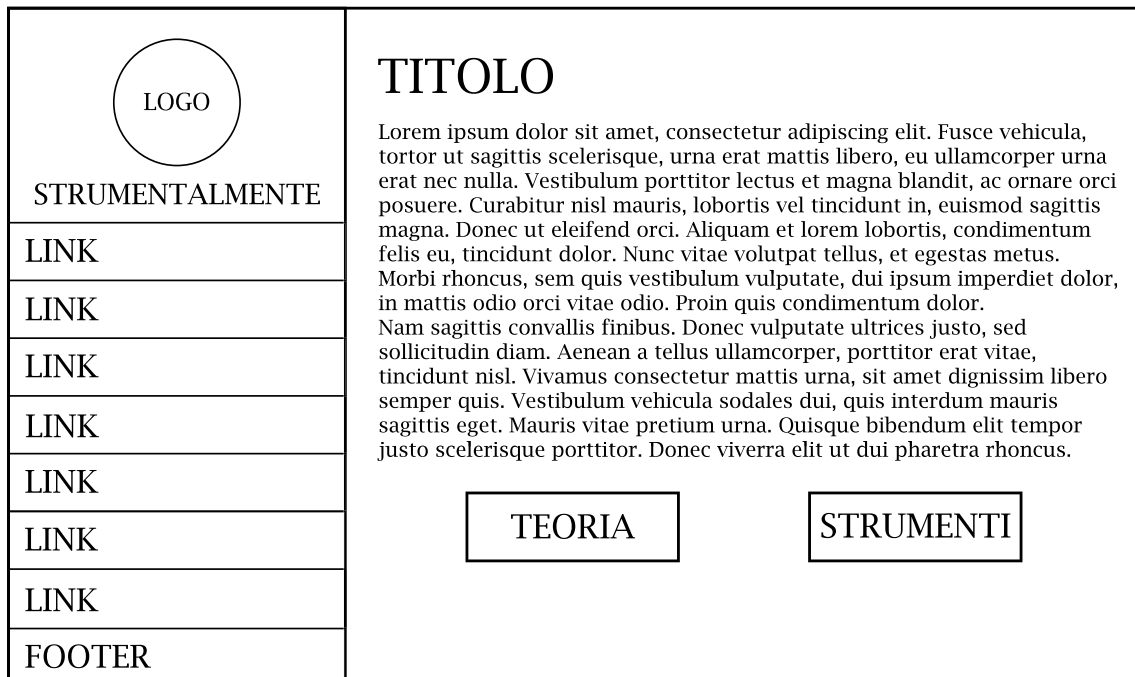
6.2 Le gabbie logiche

Come fase preliminare al design dell'applicazione vera e propria, il team ha condotto una fase di *brainstorming* che aveva come obiettivo la definizione delle varie sezioni (grafiche) dell'applicazione. Frutto di tale processo sono le seguenti gabbie logiche, che definiscono la struttura basilare che è stata scelta per l'applicazione.

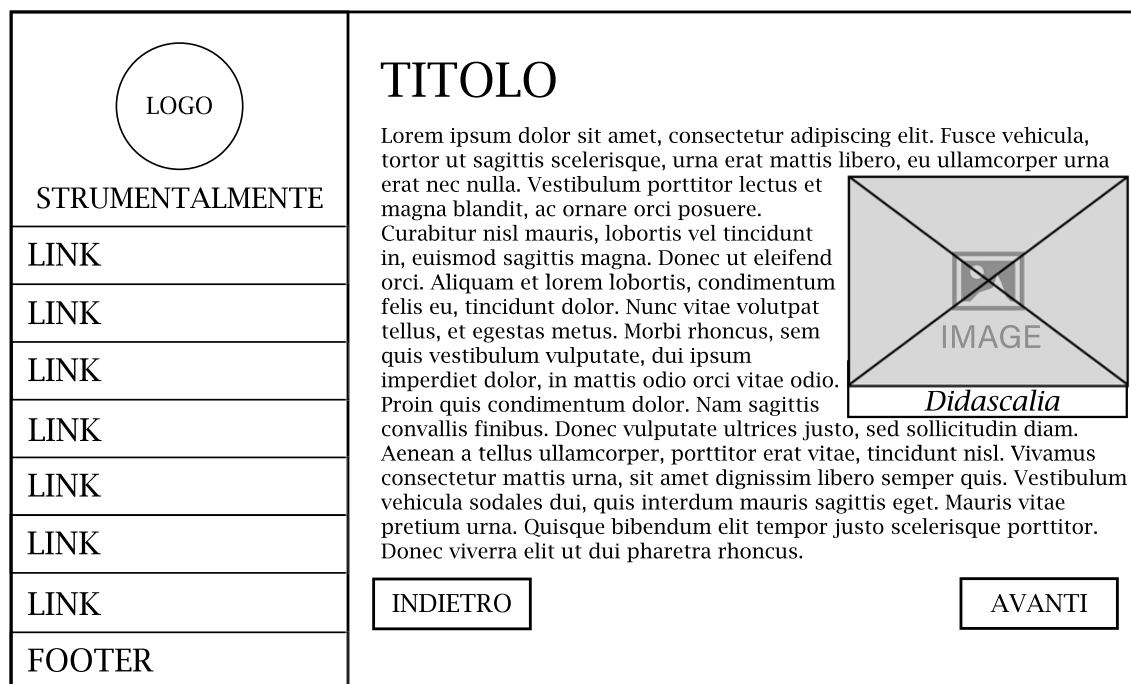
Figura 6.1: Le gabbie logiche di StrumentalMente



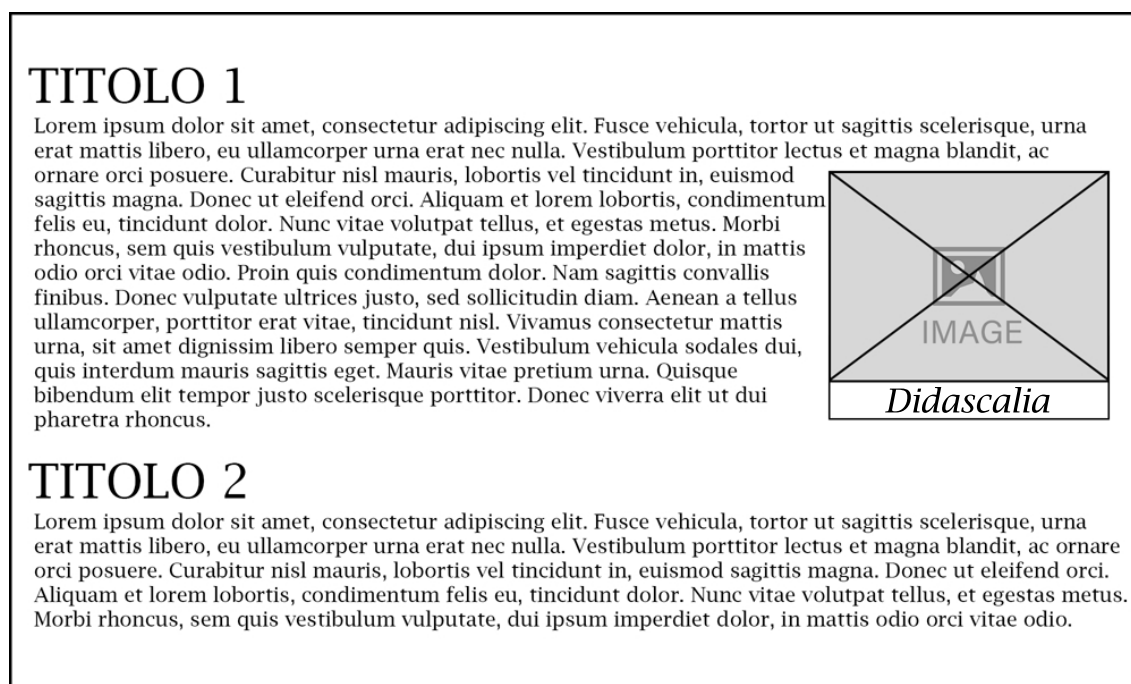
(a) La landing page




(b) La home page



(c) La struttura delle pagine dei contenuti



(d) La pagina (pop-up) di aiuto, di bibliografia e altro

| | |
|---|--|
| <div style="text-align: center;">  <p>STRUMENTALMENTE</p> </div> <div>DOMANDA 1</div> <div>DOMANDA 2</div> <div>DOMANDA 3</div> <div>DOMANDA 4</div> <div>LINK</div> <div>LINK</div> <div>ESCI</div> <div>FOOTER</div> | <h2 style="text-align: center;">TITOLO QUIZ</h2> <p>1. Domanda numero 1 Eventuale spiegazione della domanda.</p> <div style="display: flex; justify-content: space-between;"> <div> <input checked="" type="checkbox"/> Risposta 1 <input type="checkbox"/> Risposta 2 </div> <div> <input type="checkbox"/> Risposta 3 <input type="checkbox"/> Risposta 4 </div> </div> <p>2. Domanda numero 2 Eventuale spiegazione della domanda.</p> <div style="display: flex; justify-content: space-between;"> <div> <input type="checkbox"/> Risposta 1 <input type="checkbox"/> Risposta 2 </div> <div> <input type="checkbox"/> Risposta 3 <input checked="" type="checkbox"/> Risposta 4 </div> </div> <p>3. Domanda numero 3 Eventuale spiegazione della domanda.</p> <div style="display: flex; justify-content: space-between;"> <div> <input checked="" type="checkbox"/> Risposta 1 <input type="checkbox"/> Risposta 2 </div> <div> <input type="checkbox"/> Risposta 3 <input type="checkbox"/> Risposta 4 </div> </div> <div style="display: flex; justify-content: space-between; margin-top: 20px;"> <div>INDIETRO</div> <div>AVANTI</div> </div> |
|---|--|

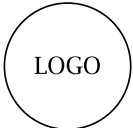
(e) Una pagina del quiz

Hai ottenuto un punteggio di X/X

CONTINUA

VERIFICA

(f) La pagina (pop-up) dei risultati del quiz

| | |
|--|---|
| <div style="text-align: center;">  STRUMENTALMENTE </div> | <h2 style="text-align: center;">TITOLO QUIZ</h2> <p>1. Domanda numero 1</p> <div style="display: flex; justify-content: space-between;"> <input type="checkbox"/> Risposta 1 <input checked="" type="checkbox"/> Risposta 3 </div> <p>2. Domanda numero 2</p> <div style="display: flex; justify-content: space-between;"> <input type="checkbox"/> Risposta 4 <input checked="" type="checkbox"/> Risposta 4 </div> <p>3. Domanda numero 3</p> <div style="display: flex; justify-content: space-between;"> <input type="checkbox"/> Risposta 1 <input checked="" type="checkbox"/> Risposta 2 </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="border: 1px solid black; padding: 5px 10px;">INDIETRO</div> <div style="border: 1px solid black; padding: 5px 10px;">AVANTI</div> </div> |
| DOMANDA 1 | |
| DOMANDA 2 | |
| DOMANDA 3 | |
| DOMANDA 4 | |
| LINK | |
| LINK | |
| ESCI | |
| FOOTER | |

(g) Una pagina di controllo del *quiz*

6.3 Le icone

Come prestabilito, l'applicazione deve avere un *look* moderno e accattivante. A tale scopo si è scelto di seguire alcune linee guida dettate dal *Material Design* di *Google*. A tal fine, si è scelto di utilizzare delle icone semplici simili, per l'appunto, a quelle che *Google* consiglia per creare applicazioni in *Material Design*.

Con uno sguardo teso alla fase di realizzazione del sistema, si sceglie di utilizzare le icone fornite dal font *Font Awesome*¹, in quanto sono disponibili (gratuitamente) diverse icone che rispettano gli standard imposti per la creazione di StrumentalMente.

¹<https://fontawesome.com/>

7 I contenuti

7.1 Bibliografia

In questa sezione è contenuta tutta la bibliografia e la sitografia utilizzata per stendere i contenuti di StrumentalMente.

7.1.1 Istruttori ed esperti

Gli istruttori ed esperti dell'Accademia musicale *Francisco Tàrraga* sono:

- Andrea **Manco**, istruttore teorico
- William **Marino**, istruttore di basso
- Giovanni **Pagliaro**, istruttore di chitarra
- Marcello **Nisi**, istruttore di batteria
- Marco **Amati**, istruttore di pianoforte

7.1.2 Libri

- [3] Matteo Carcassi. *25 studi melodici progressivi*. Curci, 1998.
- [4] Mauro Giuliani. *Centoveventi arpeggi*. Edizioni Suvini Zerboni, 1976.
- [6] Stefano Pantaleoni. *Teoria, analisi e composizione per i licei musicali*. Vol. I, II e III. Liceo Attilio Bertolucci Editore, 2015.
- [7] Luigi Rossi. *Teoria Musicale*. Edizioni Carrara, gen. 1977.
- [8] Julio Sagreras. *Le prime lezioni di chitarra*. Edizioni BERBEN, 2010.

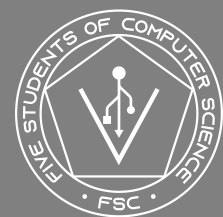
7.1.3 Materiale online

- [1] Anonimo. *Le caratteristiche del suono*. URL: [http : / / eventi . centrostudicampostrini . it / media / archive / 161107 - 1031 - il _ suono . pdf](http://eventi.centrostudicampostrini.it/media/archive/161107-1031-il_suono.pdf) (visitato il 07/02/2019).
- [2] Stefano Busonero. *Il valore delle note e delle pause*. URL: <https://www.busonero.it/2015/04-valore-delle-note-e-pause/> (visitato il 07/02/2019).
- [5] Marco “Pikkolo” Loiodice. *Corso di musica per tutti... quelli che la amano!* URL: [https : / / www . inventati . org / rebirth / pikko _ landia / corso _ di _ musica _ by _ pikkolo . pdf](https://www.inventati.org/rebirth/pikko_landia/corso_di_musica_by_pikkolo.pdf) (visitato il 07/02/2019).
- [9] Alessandro Toschi. *Il significato della musica*. Mar. 2012. URL: [https : / / alessandrotoschiblog . wordpress . com / 2012 / 03 / 25 / il - significato - della - musica /](https://alessandrotoschiblog.wordpress.com/2012/03/25/il-significato-della-musica/) (visitato il 07/02/2019).



STRUMENTALMENTE - DOCUMENTAZIONE

REALIZZAZIONE



8 Le tecnologie utilizzate

8.1 I linguaggi

Il sistema StrumentalMente è scritto utilizzando l'**HTML5** (il che porta all'utilizzo del **CSS3** per definire le varie regole di stile) e il **JavaScript**. L'intera applicazione è basata su *Node.js*, principalmente per il pacchetto **Electron**, che consente di utilizzare i precedenti linguaggi, in combinazione con il motore di rendering *Chromium*, per creare delle applicazioni *desktop*.

8.2 Il codice

Il codice sorgente di StrumentalMente è fondamentalmente suddiviso in tre file principali (escludendo i file HTML e CSS utilizzati per creare l'aspetto grafico dell'applicazione). Tali file sono:

app.js È il file principale dell'applicazione. Contiene il processo principale ("*main process*") di Electron e gestisce, quindi, tutti gli eventi principali di StrumentalMente, tra cui:

- L'apertura dell'applicazione e il *rendering* della prima finestra
- La chiusura dell'applicazione e le relative peculiarità di alcuni sistemi operativi (si pensi alla possibilità di ricreare la finestra appena chiusa su MacOS)
- L'apertura di finestre di dialogo
- L'apertura di finestra secondarie

render.js È il file principale del *rendering* dell'applicazione. Contiene tutte le funzioni da eseguire nel "*rendering process*" (processo di *rendering*) di Electron.

main.js Contiene tutte le funzioni che servono a dare dinamicità alle pagine HTML dell'applicazione.

9 Avviare dal codice sorgente

Affinchè si possa avviare StrumentalMente dal codice sorgente è necessario che siano installati `npm` e `Node.js`. Successivamente, è necessario installare le dipendenze del progetto, eseguendo i seguenti comandi nella *directory* in cui è situato il codice sorgente (cartella `src/` della *repository*):

```
npm install --save-dev electron
npm install --save-dev mousetrap
npm install --save-dev node-localstorage
```

È possibile installare *Electron Packager* per poter compilare un eseguibile (non è necessario per avviare l'applicazione) utilizzando il comando:

```
npm install -g electron-packager
```

Dopo aver installato tutte le dipendenze, è possibile avviare l'applicazione utilizzando il comando seguente all'interno della *directory* del codice sorgente:

```
npm start
```

9.1 Il Makefile

Per comodità, si è creato un `Makefile` per eseguire facilmente tutte le operazioni sul codice. Il comando `make` è da eseguire all'interno della *directory* principale della *repository*.

Il file `Makefile` contiene le seguenti regole:

(all) Avvia StrumentalMente.

install Installa le dipendenze del progetto ed *Electron Packager*.

start Avvia StrumentalMente.

deploy Compila StrumentalMente in un eseguibile per varie piattaforme. Al momento le piattaforme per cui la generazione è inclusa in questa regola sono:

- Windows (x64)

docs Ricompila tutta la documentazione partendo dai sorgenti \LaTeX .

jsdoc Ricompila la documentazione del codice in un file \LaTeX .

bib2html Converte la bibliografia in formato \BibTeX in HTML e la importa automaticamente nell'apposita sezione della pagina "about".

10 Documentazione del codice

La presente documentazione è stata generata in modo semi-automatico da *JSDoc*, *JSDoc to Markdown* e *Pandoc*.

10.0.1 Classes

accordo Classe accordo.

10.1 Funzioni

openChildWindow(pageUrl, [windowIcon]) Apre una finestra “figlia” e modale.

createWindow() Crea la finestra principale.

promptModal(parentWindow, [options], callback) Creazione della finestra di dialogo.

openMobileNavigation() Apre la navbar in modalità “mobile”. Questa funzione è mantenuta solo per consentire un eventuale eccessivo ridimensionamento della finestra.

drop(name, [defaultLinkClass]) Permette, alla pressione di un bottone, di aprire una sottolista della navbar.

setLinks(links) Cambia i link e i nomi dell’argomento precedente e quello successivo a quello attuale

initialize(initial, base) Funzione che, al caricamento della pagina, si occupa di impostare il numero di tag section presenti all’interno della pagina nella memoria locale del browser, di impostare come sezione visibile corrente la prima (sempre all’interno della memoria locale) e di nascondere tutti i tag section successivi al primo.

changeTopic(topicName, [base]) Cambia l’argomento correntemente mostrato.

initializeQuiz() Inizializza la pagina del quiz.

changeQuizSlide(finalSlide) Cambia la slide del quiz attualmente mostrata.

playStopAudio(audioTagId, buttonRef, stopButtonId) Permette di avviare, mettere in pausa o stoppare un audio.

showExitDialog() Mostra il dialogo di richiesta di conferma di uscita.

showExitFromQuizDialog(toOpen) Mostra il dialogo di richiesta di conferma di uscita dal quiz.

showQuizDialog(nomeQuiz, score, total) Mostra il dialogo con il punteggio dei quiz.

openInBrowser(link) Apre un link nel browser predefinito.

openModal(content, [options], [windowIcon]) Apre una finestra modale mostrante il contenuto richiesto.

openOnKeyboardShortcut(shortcut, content, [openAsModal]) Apre, tramite una shortcut da tastiera, una finestra mostrante il contenuto richiesto.

script_load() Seleziona un numero casuale compreso tra 1 e 7 e ne imposta l'accordo da richiedere all'utente.

replace_selected() Ripristina le checkbox selezionate dall'utente e il nome dell'accordo richiesto durante il quiz.

verify_and_store() Verifica che le selezioni effettuate dall'utente siano corrette in base all'accordo presentatogli e memorizza: se la selezione è corretta (1) o non corretta (0), le checkbox selezionate (e non) e l'accordo che l'utente doveva riprodurre.

correct_chord() In base al numero di accordo che l'utente doveva riprodurre, ripristina la sequenza di selezioni corretta nello schema.

10.1.1 accordo

Classe accordo.

Kind: global class

new accordo(nome, dita, tasto_iniziale)

| Param | Type | Description |
|----------------|---------|---|
| nome | String | Stringa che indica il nome dell'accordo |
| dita | Boolean | Sequenza di valori logici che indicano se la checkbox corrispondente è stata selezionata o meno |
| tasto_iniziale | number | Indica il numero del capotasto iniziale dell'accordo |

10.1.2 openChildWindow(pageUrl, [windowIcon])

Apre una finestra "figlia" e modale.

Kind: global function

| Param | Type | Default | Description |
|--------------|--------|-------------------|--|
| pageUrl | String | | L'URL della pagina da aprire (assoluto o relativo) |
| [windowIcon] | String | ./assets/icon.ico | L'icona della finestra. |

10.1.3 createWindow()

Crea la finestra principale.

Kind: global function

10.1.4 promptModal(parentWindow, [options], callback)

Creazione della finestra di dialogo.

Kind: global function

| Param | Type | Description |
|--------------|---------------|--|
| parentWindow | BrowserWindow | La finestra "genitore" |
| [options] | Object | Le opzioni della nuova finestra |
| callback | * | La funzione da richiamare alla chiusura della finestra |

10.1.5 openMobileNavigation()

Apri la navbar in modalità "mobile". Questa funzione è mantenuta solo per consentire un eventuale eccessivo ridimensionamento della finestra.

Kind: global function

10.1.6 drop(name, [defaultLinkClass])

Permette, alla pressione di un bottone, di aprire una sottolista della navbar.

Kind: global function

| Param | Type | Description |
|--------------------|--------|---|
| name | String | L'ID della lista che si vuole controllare |
| [defaultLinkClass] | String | La classe iniziale del bottone |

10.1.7 setLinks(links)

Cambia i link e i nomi dell'argomento precedente e quello successivo a quello attuale

Kind: global function

| Param | Type | Description |
|--------------------|--------|--|
| links | Object | Le nuove impostazioni e link |
| links.previous | String | Il nome della pagina precedente |
| links.previousLink | String | Il link della pagina precedente (il nome del file <i>senza</i> l'estensione) |
| links.next | String | Il nome della pagina successiva |
| links.nextLink | String | Il link della pagina successiva (il nome del file <i>senza</i> l'estensione) |

10.1.8 initialize(initial, base)

Funzione che, al caricamento della pagina, si occupa di impostare il numero di tag section presenti all'interno della pagina nella memoria locale del browser, di impostare come sezione visibile corrente la prima (sempre all'interno della memoria locale) e di nascondere tutti i tag section successivi al primo.

Kind: global function

| Param | Type | Default | Description |
|---------|--------|---------|--------------------|
| initial | String | | Il primo argomento |

| Param | Type | Default | Description |
|-------|--------|---------|--|
| base | String | ./ | La cartella in cui sono situati i file degli argomenti (default: ./) |

initialize~changeSlide(slide)

La funzione, in base al valore assunto da slide (true/false) cambia la sezione corrente in quella precedente (in caso di slide = false) o in quella successiva (in caso di slide = true). Inoltre si occupa di aggiornare il numero della slide corrente nella memoria temporanea del browser. Inoltre, in base al numero di slide, si occupa di rendere visibili (o nascondere) i relativi pulsanti di spostamento (avanti con id next, indietro con id back e quiz con id quiz).

Kind: inner method of initialize

| Param | Type | Description |
|-------|-------|----------------------------------|
| slide | numer | Il numero della slide da aprire. |

10.1.9 changeTopic(topicName, [base])

Cambia l'argomento correntemente mostrato.

Kind: global function

| Param | Type | Default | Description |
|-----------|--------|---------|---|
| topicName | String | | Il prossimo argomento |
| [base] | String | ./ | La cartella in cui è situato il file dell'argomento |

10.1.10 initializeQuiz()

Inizializza la pagina del quiz.

Kind: global function

10.1.11 changeQuizSlide(finalSlide)

Cambia la slide del quiz attualmente mostrata.

Kind: global function

| Param | Type | Description |
|------------|--------|--|
| finalSlide | number | La slide da aprire in seguito alla richiesta di variazione della slide. Tale valore deve essere compreso nell'intervallo $[0, n]$, dove n è il numero di slide presenti nella pagina. |

10.1.12 playStopAudio(audioTagId, buttonRef, stopButtonId)

Permette di avviare, mettere in pausa o stoppare un audio.

Kind: global function

| Param | Type | Description |
|--------------|-------------|--|
| audioTagId | String | L'ID dell'elemento <code><audio></code> da controllare |
| buttonRef | HTMLElement | Un riferimento al bottone che richiama questa funzione |
| stopButtonId | String | L'ID del bottone di Stop. |

10.1.13 showExitDialog()

Mostra il dialogo di richiesta di conferma di uscita.

Kind: global function

10.1.14 showExitFromQuizDialog(toOpen)

Mostra il dialogo di richiesta di conferma di uscita dal quiz.

Kind: global function

| Param | Type | Description |
|--------|--------|--|
| toOpen | String | Il file da aprire se è cliccato il tasto 'Sì'. Il percorso è relativo rispetto alla cartella principale. |

10.1.15 showQuizDialog(nomeQuiz, score, total)

Mostra il dialogo con il punteggio dei quiz.

Kind: global function

| Param | Type | Description |
|----------|--------|--------------------------------|
| nomeQuiz | String | Il nome del quiz. |
| score | number | Il punteggio ottenuto. |
| total | number | Il punteggio totale possibile. |

10.1.16 openInBrowser(link)

Apri un link nel browser predefinito.

Kind: global function

| Param | Type | Description |
|-------|--------|-------------------|
| link | String | Il link da aprire |

10.1.17 openModal(content, [options], [windowIcon])

Apri una finestra modale mostrando il contenuto richiesto.

Kind: global function

| Param | Type | Default | Description |
|--------------|--------|-------------------|---|
| content | String | | Il link (assoluto o relativo) da aprire |
| [options] | Object | | Le opzioni della nuova finestra |
| [windowIcon] | String | ./assets/icon.ico | L'icona della finestra modale |

10.1.18 openOnKeyboardShortcut(shortcut, content, [openAsModal])

Apri, tramite una shortcut da tastiera, una finestra mostrando il contenuto richiesto.

Kind: global function

| Param | Type | Default | Description |
|---------------|---------|---------|---|
| shortcut | String | | La shortcut da utilizzare |
| content | String | | Il link (assoluto o relativo) da aprire |
| [openAsModal] | boolean | false | Se è true, la finestra sarà aperta come modale, altrimenti sarà aperta nella stessa finestra. |

10.1.19 `script_load()`

Seleziona un numero casuale compreso tra 1 e 7 e ne imposta l'accordo da richiedere all'utente.

Kind: global function

10.1.20 `replace_selected()`

Ripristina le checkbox selezionate dall'utente e il nome dell'accordo richiesto durante il quiz.

Kind: global function

10.1.21 `verify_and_store()`

Verifica che le selezioni effettuate dall'utente siano corrette in base all'accordo presentato e memorizza: se la selezione è corretta (1) o non corretta (0), le checkbox selezionate (e non) e l'accordo che l'utente doveva riprodurre.

Kind: global function

10.1.22 `correct_chord()`

In base al numero di accordo che l'utente doveva riprodurre, ripristina la sequenza di selezione corretta nello schema.

Kind: global function

11 Bug e problemi noti

11.1 La velocità

L'applicazione non è scritta come applicazione nativa, quindi la sua velocità e responsività non è lontanamente paragonabile a quella di un'applicazione nativa.

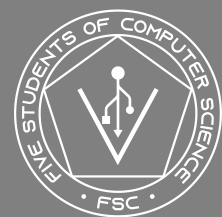
Questa limitazione è dovuta principalmente a tre fattori: il linguaggio di programmazione utilizzato (JavaScript), il *framework* Electron e il motore di *rendering* su cui quest'ultimo si basa (Chromium).

Nonostante gli sforzi compiuti dal team di sviluppo per limitare questo problema, per esempio sacrificando la leggibilità del codice a favore di tipologie di istruzione più rapide (si pensi al `forEach` in confronto al classico ciclo `for`: il primo è mediamente più lento del secondo ma è più leggibile), l'applicazione resta poco rapida.



STRUMENTALMENTE - DOCUMENTAZIONE

RINGRAZIAMENTI E RIFERIMENTI



Ringraziamenti

Si ringraziano i beta tester M. *Chiarofonte*, P. *Caracciolo*, B. *De Salvo*, R. *Manno*, C. *Marino*, senza cui non si sarebbe potuto completare lo sviluppo di questo sistema.

Si ringrazia inoltre l'Accademia Musicale *Francisco Tàrraga* di Taranto che ha messo a disposizione del team degli istruttori di musica. Si ringrazia, in particolare:

- Andrea Manco, istruttore teorico
- William Marino, istruttore di basso
- Giovanni Pagliaro, istruttore di chitarra
- Marcello Nisi, istruttore di batteria
- Marco Amati, istruttore di pianoforte